Novel Memetic Computing Structures for Continuous Optimisation

Fabio Caraffini, M. Sc.

Submitted in partial fulfilment
of the requirements for the degree of Doctor of Philosophy.

Faculty of Technology

De Montfort University

May 2014

Acknowledgements

I could not have imagined having a better advisor for mentoring my doctoral studies, on both the professional and human level. A special thank you is owed to Prof. Ferrante Neri, without whom this piece of work would not have been possible.

I would also like to thank Prof. Robert John for welcoming me to the Centre for Computational Intelligence at De Montfort University and awarding me with a Postgraduate Research Studentship that provided the necessary financial support for me to carry on my research. Dr. Simon Coupland and Dr. Lorenzo Picinali for their constant help and availability. A sincere thank you goes to Lorenzo and Ferrante for supporting me in several side projects that, despite the issues and challenges with the technical infrastructure, led to the creation of a powerful computational cluster and the generation of numerical data worth several publications.

Thanks to Hannah May who accepted to proofread this thesis, despite the short notice, and did a meticulous job.

Moreover, I want to thank all the people I have met all over the world, who kept me company during this experience and helped me when in "trouble". In particular my friends and colleagues in Jyväskylä (thanks to Giovanni and Ulpukka for taking care of me after my accident and Adriano for company during the long-cold-dark Finnish winter), in Leicester (my personal trainer Valeria and personal GP Nikol) and my old friends in Italy Zoso and Mito who still find the time to contact me.

Last but not least, I am grateful to my family for encouragement and support, despite the distance. I would like to express my gratitude to my parents Mauro and Santina, my brother Diego but also my uncle Italo and my aunt Mirella who always care about me, as well as "i citti" Perseo, Antea and Paride. A particular acknowledgement goes to my mum for prompt overseas shipments of home-made delicacies, and to my granny Rosa for knitting every year piles of scarves and hats that have warmed up the freezing Finnish weather and the rainy English winters.

Fabio

Abstract

This thesis studies a class of optimisation algorithms, namely Memetic Computing Structures, and proposes a novel set of promising algorithms that move the first step towards an implementation for the automatic generation of optimisation algorithms for continuous domains. This thesis after a thorough review of local search algorithms and popular meta-heuristics, focuses on Memetic Computing in terms of algorithm structures and design philosophy. In particular, most of the design carried out during my doctoral studies is inspired by the lex parsimoniae, aka Ockham's Razor. It has been shown how simple algorithms, when well implemented can outperform complex implementations. In order to achieve this aim, the design is always carried out by attempting to identify the role of each algorithmic component/operator. In this thesis, on the basis of this logic, a set of variants of a recently proposed algorithms are presented. Subsequently a novel memetic structure, namely Parallel Memetic Structure is proposed and tested against modern algorithms representing the state of the art in optimisation. Furthermore, an initial prototype of an automatic design platform is also included. This prototype performs an analysis on separability of the optimisation problem and, on the basis of the analysis results, designs some parts of the parallel structure. Promising results are included. Finally, an investigation of the correlation among the variables and problem dimensionality has been performed. An extremely interesting finding of this thesis work is that the degree of correlation among the variables decreases when the dimensionality increases. As a direct consequence of this fact, large scale problems are to some extent easier to handle than problems in low dimensionality since, due to the lack of correlation among the variables, they can effectively be tackled by an algorithm that performs moves along the axes.

Concerning tables, pseudo-code and notation

This section clarifies the notation used throughout this thesis. Regarding the use of boldface, vectors are always indicated in boldface and represented in Cartesian coordinate. The operations of *cross* and *scalar product* are indicated with $\times$ and $\cdot$, respectively. Sets and matrices are in boldface too. *Matrix product* has been left blank while the *Hadamard* (element-wise) *product* is represented by the following symbol "$\circ$". For example, given two n-by-m matrices $\mathbf{A}$ and $\mathbf{B}$ ($n, m, k \in \mathbb{N}^+$):

$$\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,m} \\ A_{1,2} & A_{2,2} & \dots & A_{2,m} \\ \vdots & \vdots & \ddots & \dots \\ A_{j,1} & \dots & \dots & A_{j,k} \end{bmatrix} \circ \begin{bmatrix} B_{1,1} & B_{1,2} & \dots & B_{1,m} \\ B_{1,2} & B_{2,2} & \dots & B_{2,m} \\ \vdots & \dots & \ddots & \dots \\ B_{j,1} & \dots & \dots & B_{j,m} \end{bmatrix} = \begin{bmatrix} A_{1,1}B_{1,1} & A_{1,2}B_{1,2} & \dots & A_{1,k}B_{1,m} \\ A_{1,2}B_{1,2} & A_{2,2}B_{2,2} & \dots & A_{2,k}B_{2,m} \\ \vdots & \dots & \ddots & \dots \\ A_{j,1}B_{j,1} & \dots & \dots & A_{j,k}B_{j,k} \end{bmatrix}$$

while if $\mathbf{A}$ is n-by-m and $\mathbf{B}$ m-by-k:

$$\mathbf{AB} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,m} \\ A_{1,2} & A_{2,2} & \dots & A_{2,m} \\ \vdots & \vdots & \ddots & \dots \\ A_{j,1} & \dots & \dots & A_{j,k} \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} & \dots & B_{1,m} \\ B_{1,2} & B_{2,2} & \dots & B_{2,m} \\ \vdots & \dots & \ddots & \dots \\ B_{j,1} & \dots & \dots & B_{j,m} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} A_{1,i}B_{i,1} & \sum_{i=1}^{m} A_{1,i}B_{i,2} & \dots & \sum_{i=1}^{m} A_{1,i}B_{i,k} \\ \sum_{i=1}^{m} A_{2,i}B_{i,1} & \sum_{i=1}^{m} A_{2,i}B_{i,2} & \dots & \sum_{i=1}^{m} A_{2,i}B_{i,k} \\ \vdots & \dots & \ddots & \dots \\ \sum_{i=1}^{m} A_{n,i}B_{i,1} & \dots & \dots & \sum_{i=1}^{m} A_{n,i}B_{i,k} \end{bmatrix}$$

Regarding pseudo-code, matrices are initialised by means of the following methods:

- $ones\,(i, j)$: returns a i-by-j all-ones matrix
- $eye\,(i, j)$: returns a i-by-j identity matrix
- $randomSampling\,(i, j, \mathcal{D})$: samples i uniformly distributed j-dimensional vectors within the hyper-space $\mathcal{D}$

The following distributions have also been used throughout this thesis:

- $\mathcal{N}\,(\boldsymbol{\mu}, \mathbf{C})$: (multivariate) Normal distribution with mean value $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$ (returns a column matrix of Gaussian distributed real-valued numbers, having the same size of $\boldsymbol{\mu}$, e.g. if μ is a scalar value then $\mathbf{C}$ is the variance σ^2 and the distribution degenerates into a univariate Gauss function)

- $\mathcal{B}^{\pm 1}(p)$: modified Bernoulli distribution (returns 1 with a given probability p and -1 with probability $1-p$)

- $\mathcal{C}(l,\gamma)$ Cauchy distribution with location parameter l and scale parameter γ (returns a single Cauchy distributed value)

- $\mathcal{U}(a,b)$: Uniform distribution (returns a uniform distributed real number $\in [a,b]\,,\ a,b,\in \mathbb{R}$

- $\mathcal{I}(a,b)$: returns an integer number uniformly distributed in $[a,b] \subset \mathbb{N}$

Pseudo-code refers to the implementation that has been coded and employed for my PhD. Despite the language being mostly formal, e.g. the mathematical symbolism $f(\mathbf{x})$ is meant to represent the value of f in $\mathbf{x}$ rather than the evaluation of the functional call, they are fully detailed in order to allow the replication of code and results.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

μDE-3SOME Micro-population Differential Evolution-3SOME. 80, 82, 83, 121, 147, 153–155

μDEA Micro-Differential Evolution with extra moves along the Axes. 60, 63, 66, 68, 97, 100, 124

(1+1)-CMA-ES $(1+1)$ Covariance Matrix Adaptation Evolution Strategy. 38, 81–84, 153–155

(1+1)-CMA-ES-3SOME 3SOME with $(1+1)$ Covariance Matrix Adaptation Evolution Strategy. 81–83, 121, 147

3SOME Three Stage Optimal Memetic Exploration. 1, 2, 6, 68–70, 77–90, 95, 99, 101, 103–106, 121, 124, 147, 149, 151–157

3SOME-Powell 3SOME equipped with Powell local search. 84–86, 121, 148, 151–153

3SOME-Rosenbrock 3SOME equipped with Rosenbrock local search. 84–86, 121, 148, 151–153

AI Artificial Intelligence. 10, 40, 59, 108

BBOB Black-Box Optimisation Benchmark. 72, 73, 75, 76, 79, 82–86, 88–91, 93, 94, 102, 104, 105, 114, 116, 117, 130–134, 149, 151–156, 158, 160, 161, 163, 167

BGA Breeder Genetic Algorithm. 33, 34, 59

BLX BLend X-over. 33, 34

CA Cultural Algorithm. 58

CA-ILS Cultural Algorithms with Iterated Local Search. 117, 118, 148

cBFO compact Bacterial Foraging Optimisation. 50

CCPSO2 Cooperatively Coevolving Particle Swarms for Large Scale Optimisation. 42, 43, 57, 62, 89, 104–106, 118, 119, 124, 146, 150, 163–168

cDE compact Differential Evolution. 50, 53, 56, 57, 70, 89, 90, 147, 150

cDE-light compact Differential Evolution light. 53, 55–57, 147

CDF Cumulative Distribution Function. 50, 52

CEC Congress on Evolutionary Computation. 62, 72, 73, 75–80, 84–91, 93, 94, 102, 104–106, 110, 112, 114–118, 129–133, 135, 150, 153, 155–168

cGA compact Genetic Algorithm. 50, 146, 150

CI Computational Intelligence. 4, 10

CIO Computational Intelligence Optimisation. 5, 10, 11, 13, 27, 58, 59, 66, 68, 143

CLPSO Comprehensive Learning Particle Swarm Optimiser. 41, 42, 57, 70, 89, 93, 94, 104–106, 117, 118, 146

CMA-ES Covariance Matrix Adaptation Evolution Strategy. 11, 35–38, 57, 59, 62, 63, 67, 82, 90, 109, 111, 112, 117, 118, 120, 121, 124, 126, 128, 129, 137, 146, 147, 150

CMA-ES-RIS CMA-ES super-fit scheme for the re-sampled inheritance search. 90, 100, 120, 121

cPSO compact Particle Swarm Optimisation. 50, 117, 118, 147

DE Differential Evolution. 3, 27, 30, 43, 44, 46–48, 50, 56, 57, 60, 62–65, 80, 81, 100, 118, 121, 123, 124, 147

EA Evolutionary Algorithm. 27, 30–32, 40, 44, 58–60, 117, 143

EC Evolutionary Computing. 31, 39

EDA Estimation of Distribution Algorithm. 40, 50, 100, 101

EMS Elementary Memetic Structure. 97, 98, 100

EP Evolutionary Programming. 39, 57

EPSDE Ensemble of Parameters and Mutation Strategies Differential Evolution. 49, 57, 62–64, 97

EPSDE-LS Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search. 62, 63, 68, 99

ES Evolution Strategy. 34–36, 38, 41, 57, 97

FPS Fitness Proportional Selection. 33

FWER Familywise Error Rate. 141

GA Genetic Algorithm. 32, 34, 35, 39, 44, 47, 57–60, 63, 97

GLSA Genetic Local Search Algorithms. 58

GP Genetic Programming. 5, 39, 57

GS Global Search. 26, 60

GSS Golden Section Search. 21, 84, 121, 148

IRQ Intermediate Research Questions. 5–8, 77, 95, 96, 120, 122, 125, 136, 137

ISPO Intelligent Single Particle Optimisation. 42, 57, 90, 92–94, 146, 150

JADE Adaptive Differential Evolution with Optional External Archive. 48, 49, 60, 70, 89, 93, 94, 104–106, 117, 118, 147, 161

jDE Self-Adapting Control Parameters in Differential Evolution. 48, 68, 147

KISS Keep it Simple, Stupid. 68

LJP Lennard-Jones Potential. 72, 73, 118, 119, 132

LS Local Search. 1, 3, 6, 10–15, 25, 26, 32, 34, 47, 58–60, 62, 63, 66, 67, 80, 85, 87, 88, 95, 97, 99, 101, 113, 114, 121, 138

LSOP Large Scale Optimisation Problems. 7, 8, 18, 24, 36, 42, 43, 59, 60, 68, 79, 87, 90, 93, 95, 118, 121–125, 129, 132

MA Memetic Algorithm. 1, 3, 6, 27, 58–60, 62, 63, 66, 96, 118

MA-LSCh Memetic Algorithm with Local Search Chain. 59, 60, 63, 68, 70, 89, 90, 100, 104–106, 118, 119, 147, 163–168

MADE Multicriteria Adaptive Differential Evolution. 68

MC Memetic Computing. 1–4, 6, 7, 13, 27, 58, 59, 63, 66–69, 77, 84, 87, 89, 90, 93, 96, 97, 99–102, 107, 112, 120, 136

MDE-pBX Modified Differential Evolution with p-Best Crossover. 50, 57, 60, 62, 70, 89, 97, 104–106, 118, 119, 147, 163–168

ML-3SOME Meta-Lamarckian-3SOME. 77, 78, 121, 149

MN Memetic Node. 98, 99, 101, 103, 107, 112, 113

MS-CAP Multi-Strategy Coevolving Aging Particle. 63, 65, 68, 100, 121

NFL No Free Lunch. 2, 5, 66, 76, 99, 143

nuSA Non uniform Simulated Annealing. 28, 29, 57

PDF Probability Density Function. 50, 51, 53, 57

PID Proportional Integral Derivative. 90

PMS Parallel Memetic Structure. 2, 7, 89, 90, 96, 99–107, 112, 117, 118, 121, 124, 136, 148, 161–164

PSO Particle Swarm Optimisation. 3, 40–42, 44, 57, 60, 63, 64, 93, 100, 117, 118, 121, 123–125

rcGA Real-Coded Compact Genetic Algorithm. 50

RI-3SOME Rotation Invariant-3SOME. 80, 121

RIS Re-sampled Inheritance Search. 87–91, 95, 97, 99–101, 113, 121, 124, 136, 147, 155–159

RIS-3SOME Rotation Invariant Shrinking-3SOME. 80–84, 121, 147, 153–155

RS Re-sampled Search. 87–89, 97, 121, 136, 147

RWS Roulette Wheel Selection. 33, 34

S-3SOME Shrinking-3SOME. 78–80, 95, 121, 150

SA Simulated Annealing. 5, 28, 29, 57

SADE Self-adaptive Differential Evolution Algorithm for Numerical Optimisation. 47, 57, 60, 118, 147, 150

SI Swarm Intelligence. 27, 30, 40, 44

SISC Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimisation. 73, 128–131, 133

SMADE Super-fit Multicriteria Adaptive Differential Evolution. 68

SPAM Separability Prototype for Automatic Memes. 2, 7, 107, 108, 112–120, 126, 132, 148, 165–168

SPSA Simultaneous Perturbation Stochastic Approximation. 12, 13, 21, 23, 26, 148

SUS Stochastic Universal Sampling. 33

VISPO Very Intelligent Single Solution Optimisation. 92–95, 100, 121, 148, 159, 160

Chapter 1

Introduction

This piece of work is the outcome of a deep investigation about the algorithmic structure of a class of modern optimisers referred as Memetic Computing (MC) algorithms, in order to understand their working principles and be able to identify the fundamental building blocks forming them. The goal is to produce a grammar whose syntactic elements are basic operators, which can be combined together for tackling a given problem. The aim is to use such basic components in order to design novel memetic structures that being simple, still display high performances, and to propose a software framework for selecting and automatically assembling them together, so coding a first prototype for the automatic generation of optimisation algorithms for continuous domains. In few words, the very final goal motivating this piece of research can be formulated with the following research question:

IS IT POSSIBLE TO DETECT THE "ATOMIC UNITS" FORMING A MEMETIC STRUCTURE AND CODE A PROTOTYPE OF A SOFTWARE THAT, GIVEN AN OPTIMISATION PROBLEM DEFINED IN A CONTINUOUS DOMAIN, EXPLOIT THEM FOR AUTOMATICALLY GENERATING A TAILORED SOLVER?

The automatic generation of optimisation algorithms in continuous domains, is an ambitious goal that will be fully achieved in many years of research, but still possible, by passing through a number of intermediate objectives. This thesis proves the feasibility of this long-term research project, by partially achieving the big picture of automatic algorithms generation. In particular, a general prototype for the automatic design of MC algorithms has been proposed and used to tackle optimisation problems according to two main features: separability and dimensionality. Other important features characterising an optimisation problem, such as ill-conditioning and multi-modality, will be added in the future. The most difficult step will be to consider these properties at the same time, rather than separately one at time.

In order to achieve this first result, my research has initially gone through an extensive literature review (and implementation and testing) of the most widely used Local Search (LS) routines and population-based algorithms for global optimisation, them being the foundations for building every Memetic Algorithm (MA), and subsequently, MC structures have been taken into consideration. In particular, the Three Stage Optimal

Memetic Exploration (3SOME) algorithm, see (Iacca, Neri, Mininno, Ong & Lim 2012), has been thoroughly studied and altered as a starting point for understanding MC structures and proposing novel algorithms based on the same principle, i.e. algorithmic simplicity. As an evolution of the 3SOME algorithm, a novel memetic structure, that is the Parallel Memetic Structure (PMS) has been presented after defining a new notation for representing memetic structures. The same topology used in PMS has then been exploited in the Separability Prototype for Automatic Memes (SPAM). The approach proposed for SPAM can build an optimisation algorithm from scratch, by studying the problem and extracting information regarding its "degree of separability", which has been approximated via an index measuring the correlation amongst the design variables. Finally, a study on the effect of increasing dimensionality values on the correlation index, has been presented. A detailed breakdown, chapter by chapter, of the described research activity is provided at the end of this introductory section.

1.1 Motivation and overview of related previous work

For about three decades (1960-1990) computer scientists investigated optimisation methods with the aim of detecting the ultimate algorithm capable of outperforming all the others. During those years, important computational paradigms, see for example (Eiben & Smith 2007), were defined. The search for the best optimiser has been stopped by the publication of the No Free Lunch (NFL) theorem (Wolpert & Macready 1997). The NFL theorem mathematically proves that the average performance of any pair of algorithms A and B across all possible problems is the same. This fact implies that if an algorithm performs well on a certain class of problems, then it necessarily pays for that with degraded performance on the set of all remaining problems (as this would be the only way that all algorithms can have the same performance averaged over all the optimisation problems). Even though the NFL theorem is not generally valid, as discussed in (Auger & Teytaud 2010) and (Poli & Graff 2009), see Appendix B for details, the concept that there is no universal optimiser had a significant impact on the scientific community. More specifically, since the performance of each algorithm over all the possible problems has theoretically been proven to be the same, it was clear that there was no longer a reason to discuss which algorithm is, in general, better or worse. On the contrary, as the most direct consequence of NFL theorem, computer scientists started to propose algorithms which were tailored to specific problems in order to design an optimiser that could display its highest performance for the specific problem of interest, e.g. see (Chabuk, Reggia, Lohn & Linden 2012), (Salvatore, Caponio, Neri, Stasi & Cascella 2010) and (Tao, Zain, Ahmed, Abdalla & Jing 2012). In real-world problems, the use of a tailored algorithm (with its parameter setting) for each problem could be extremely impractical as it would likely require the constant action of an optimisation expert to adjust the design to the industrial needs. In addition, the implicit non-stationary nature of optimisation processes imposes that there is no unique most suitable algorithm and parameter setting during run-time. In order to tackle these issues, modern optimisation algorithms are composed of multiple search operators that are coordinated with the aim of promptly reacting to the necessities of the optimisation processes and thus successfully address reasonable ranges of problem features. Although a proper taxonomy cannot be easily formulated, due the overlap of the various classes, a

possible classification of modern optimisation algorithms is given in the following subsections.

1.1.1 Adaptive and self-adaptive algorithms based on a paradigm

These algorithms are based on one single paradigm/optimisation framework, but include different variants of it. The search operators corresponding to each variant are coordinated by means of a feedback-based criterion. For example, in the context of Differential Evolution (DE), the algorithm proposed in (Qin, Huang & Suganthan 2009) combines multiple mutation strategies coordinated on the basis of a learning period and a randomised success based logic. In (Mallipeddi, Mallipeddi & Suganthan 2010) a so-called ensemble of mutation and crossover strategies, as well as the related parameters are encoded within the solutions and evolve with them. Other harmonic self-adaptive combinations of components within the DE framework are proposed in (Brest & Maucec 2011) and (Brest, Korosec, Silc, Zamuda, Boskovic & Maucec 2013). In (Zhu, Zhou, Ji & Shi 2011) an adaptive memetic approach is designed in the context of Particle Swarm Optimisation (PSO) with reference to DNA sequence compression.

1.1.2 Global optimisation frameworks that use multiple local search algorithms

These approaches, often labelled as MAs, see (Moscato & Norman 1989) and (Moscato 1989), are composed of a global optimisation framework and multiple diverse local search LS components coordinated within this framework, see for example (Neri, Cotta & Moscato 2011), (Ong, Lim & Chen 2010), and (Neri & Cotta 2012). An adaptive logic is supposed to select the most appropriate local search. Some famous examples of adaptive schemes for memetic algorithms are the Meta-Lamarckian Learning (Ong & Keane 2004) that varies the selection probability of the local search on the basis of its previous success, the self-adaptation and coevolution, see (Krasnogor & Smith 2005), (Smith 2007), and (Smith 2012) that encode within the solution or in a parallel population the local search activation, and the fitness diversity adaptation, see (Caponio, Cascella, Neri, Salvatore & Sumner 2007) and (Neri, Toivanen, Cascella & Ong 2007), that balances exploration and exploitation needs on the basis of the estimated diversity of the population. Recently, a generalisation of the concept of MA to the broader concept of MC has been given; see (Ong et al. 2010) and (Iacca, Neri, Mininno, Ong & Lim 2012). According to the MC notation, an algorithm is a structure without a prefixed structure and composed of heterogeneous operators.

1.1.3 A list of multiple algorithms coordinated by a supervisor

In these frameworks, a list of multiple algorithms is coordinated by means of a heuristic rule or supervisory/adaptive scheme. Usually these approaches are successfully used for addressing domain specific optimisation problems, especially (but not always) combinatorial. A family of algorithms belonging to this

class is often referred to as optimisation portfolios. Some examples of portfolios of heuristically coordinated algorithms have been proposed in (Vrugt, Robinson & Hyman 2009) and (Peng, Tang, Chen & Yao 2010). Another popular portfolio approach for the propositional satisfiability problem is the so-called SATzilla platform, see (Xu, Hutter, Hoos & Leyton-Brown 2008) and (Hoos 2012). Within this platform, a study to assess the trade-off of the search algorithms with the aim at determining an automatic coordination system is presented in (Hutter, Hoos & Leyton-Brown 2010). A recent study that models the behaviour of optimisers and predicts their run time is given in (Hutter, Xu, Hoos & Leyton-Brown 2014). Another famous type of algorithmic structure composed of a list of solvers (optimisation algorithms) intelligently coordinated is the hyper-heuristic. In hyper-heuristics the coordination amongst multiple algorithms is performed by means of a machine learning technique. The latter acts as a supervisor structure that learns which algorithms are the most suitable for a given problem. A famous example of hyper-heuristic implementation in the field of timetabling and rostering are proposed in (Cowling, Kendall & Soubeiga 2000) and (Burke, Kendall & Soubeiga 2003) while two extensive literature reviews on the topic are given in (Özcan, Bilgin & Korkmaz 2008) and (Burke, Hyde, Kendall, Ochoa, Ozcan & Woodward 2010), respectively. In (Burke, McCollum, Meisels, Petrovic & Qu 2007) graph colouring heuristics are coupled with a random ordering heuristic. Several attempts to perform the coordination of the heuristics are given in the literature. A classic approach consists of assigning a score and rewarding the most promising heuristics: the so-called choice function, see (Cowling et al. 2000). A combination of the choice function with a randomized criterion, is also very broadly used, see e.g. (Kendall, Soubeiga & Cowling 2002). Among many coordination schemes present in the literature, more sophisticated approaches make use of reinforcement learning in a stand-alone or combined fashion, see e.g. (Burke et al. 2003) and (Dowsland, Soubeiga & Burke 2007), and memory-based mechanisms, see (Burke & Bykov 2008). In (Acampora, Gaeta & Loia 2011) and (Acampora, Cadenas, Loia & Ballester 2011) elegant learning schemes are coupled with multiple operators (multi-agents) for addressing complex optimisation problems.

1.2 Discussion

In some cases this classification is mainly a philosophical one and cannot be easily spotted within the implementation details. For example, some MC structures can also be adaptive, see (Caponio et al. 2007), (Krasnogor & Smith 2005), (Nguyen & Yao 2008), (Smith 2007), and (Smith 2012). Nonetheless, this philosophical distinction has a major impact when we consider the future trend of research in Computational Intelligence (CI). More specifically, the first category attempts to stretch to a range of various problems single algorithmic paradigms. The second category attempts to tackle the problem by balancing global and local search. The third category relies on the idea that a list of algorithms is likely to contain efficient solvers. These solvers, by means of an adaptation mechanism, become at run time those that are most likely chosen. The topic of automatic design of optimisation algorithm is currently intensively discussed within the computational science community, see (Meuth, M. Lim, Ong & Wunsch-II 2009) and (Zhu, Jia & Ji 2010), as well as in related fields from slightly different perspectives, see e.g. (Hoos 2012), (Wu, McCall, Corne & Regnier-Coudert 2012) and (Ren, Jiang, Xuan & Luo 2012). Automatic design of optimisation algorithms is thoroughly studied in

(Hamadi, Monfroy & Saubion 2012) from multiple perspectives. For example, in (Hoos 2012) a survey on automatic parameter setting is given while in (Epstein & Petrovic 2012) a learning system for the coordination of heuristics in the context of constrained optimisation is presented. In (Battiti & Campigotto 2012) the use of reinforcement learning for online parameter tuning of stochastic local search algorithms is shown. In (Maturana, Fialho, Saubion, Schoenauer, Lardeux & Sebag 2012) an evolutionary framework that adaptively selects its components during the run time is presented. In (Hamadi, Jabbour & Sais 2012) an adaptation technique to improve the performance of a component of the SATzilla platform is shown. Moreover, a software package named FRACE, illustrated in (Birattari, Yuan, Balaprakash & Stützle 2010), employs a statistical procedure for automatically performing the configuration of the algorithms. An extension of FRACE, namely IRACE, is presented in (Lopez-Ibanez, Dubois-Lacoste, Stützle & Birattari 2011). These are only some of the examples of the many studies/attempts on automatic generation of metaheuristics. Nonetheless, due to its complexity, an efficient and generally valid solution has not been found yet and, it will require still several years of research in mathematics and computer science. At the present stage, the above-mentioned pioneering studies on this topic tackle the automatic design by means of success/failure criteria, i.e. those parts of the algorithms that appear the most successful are more likely to be selected in the following stages of the optimisation. It can be argued that this kind of approach contains three major limitations. First, a success-based criterion is a trial-and-error strategy that requires a set of random choices before obtaining the selection of the correct solvers/operators. Second, an optimisation run is a dynamic process. Hence, for example, a successful search move at early stages of the optimisation may not be the most proper one in later stages. Third, this approach does not exploit the available pieces of information about the problem. More specifically, in real-world cases, optimisation problems very often must be tackled as a black box, i.e. the algorithmic designer may need to attempt to optimize an objective function even when the problem details are not available. This may happen, for example, when the objective function is the result of a measurement process within an actual experiment, see (Caponio et al. 2007) and (Salvatore et al. 2010). Nonetheless, the algorithmic designer can study how the problem looks like from the optimisation perspective by performing preliminary tests and extract those pieces of information that allow an algorithmic design tailored to the problem. On the basis of these considerations this work proposes to develop a software platform for automatically designing optimisation algorithms. This is the most general and final aim of this piece of research. Although the automatic design of meta-heuristics can be perceived as a meta-optimisation problem (the search of the optimal optimiser), it must be observed that, as proven in (Poli & Graff 2009), the NFL theorem is not valid in a meta-space (of operators) such that of Genetic Programming (GP) or Hyper-heuristics. In other words, a theoretical result guarantees that while for numerical problems (under the hypotheses of the theorem) there is no best optimiser, some design strategies can generally outperform some others. Thus, an automatic algorithmic designer can potentially tackle optimisation problems faster and more accurately than any other designing system.

1.3 Thesis structure

The results achieved during my doctoral studies cover different aspects of Computational Intelligence Optimisation (CIO) for continuous domains. Most of the algorithms mentioned or described in this thesis have been implemented and applied to real world applications, from Machine Learning to System Control and Robotics. For a complete description see the list of the published material in Appendix C. For the sake of clarity and consistency, this thesis has been restricted to the theoretical part of my research, focussing on those incremental steps leading to presentation and implementation of a prototype for the automatic generation of optimisers for continuous problems. The main research question opening this work has then been broken up in **Intermediate Research Questions (IRQ)**, and the work has then been reorganised as follow:

CHAPTER 2 provides an introduction to CIO starting with generalities about optimisation, and passing through a thorough literature review of the most important classic local searchers. A description of their working principle is given, as well as implementation details supported by pseudo-codes. Global optimisation is introduced, and also the concept of meta-heuristic, with fundamental single solution examples such as Simulated Annealing (SA). These techniques will be a part of more complex structures described in the reminder of this thesis. Their detailed description has been placed here in order to make the reading of the experimental part of this thesis easier and straightforward.

CHAPTER 3 is mainly focussed on a literature review of principal bedrocks of population-based optimisation. Recent state-of-the-art variants are explained in depth, since used for comparisons with the optimisers proposed during my PhD, or employed for the design of further optimisation algorithms in a memetic fashion.

CHAPTER 4 rounds off the literature review by introducing the concepts of MA and MC. The main differences (and similarities) between the two categories are explained and clarified by means of examples, i.e. through the presentations of recent algorithms belonging to the two classes (a complete taxonomy comprehending past most relevant achievements is already present in this preface), and finally, the principle of Ockhams Razor[1] in MC is introduced by presenting the 3SOME algorithm. The importance of this principle is fundamental for this piece of research, and represents the start point towards the study of memetic structures and the automatic generation of optimisation algorithms from scratch, according to a bottom-up logic.

CHAPTER 5 introduces the methodologies used for designing, testing and comparing novel algorithms against the state-of-the-art over a manifold studying board of functions in multiple dimensionality values. It clarifies the philosophy behind the algorithmic design, the software and hardware infrastructure set-up for generating data, and the notation adopted in tables for statistic validation. The parameters tuning procedure is also described in order to justify the parameters setting chosen for generating data, which is reported in Appendix D in order to allow the replication of this work.

[1] Well known principle in science from William of Ockham, also spelled as Occam Razor or indicated with the Latin "lex parsimoniae".

CHAPTER 6 investigates memetic structures starting with a deep analysis and experimentation on the structure of the 3SOME algorithm. 3SOME is then altered and compared with its variants: enhanced version of its operators are proposed, the final LS routine is changed with others popular methods, and different coordination strategies, e.g. Meta-lamarckian learning, are tested. The aim of this experiments is to better understand the role of each meme, in order to be able to draw up a grammar whose syntactic elements are memes (see Chapter 7), and to estimate the impact of the algorithmic structure on the performances. The last issue, is fundamental in order to get closer to the automatic design of MC optimisers, since each single meme has to be selected to tackle a certain property of the problem, and coordinated with the others within the algorithmic structure. The aim of this chapter can be formalised with the following intermediate research questions:

- ***"Is it possible to improve upon 3SOME so making its structure adapt to the problem, and able to handle different properties of the fitness landscape?"*** **(IRQ I)**
- ***Beside the efficiency and the understanding of the role of each chosen operator, does the algorithmic structure matter? What's the impact of the algorithmic structure in MC?"*** **(IRQ II)**
- ***"Is the algorithmic complexity in optimisation actually supported by the results? Is the complexity, in MC, an algorithmic feature which makes the algorithms any better? Moreover, is 3SOME's structure without redundancy or can be further simplified?"*** **(IRQ III)**

The answers to the first research question was achieved step by step with the following publications: (Caraffini, Iacca, Neri & Mininno 2012*b*), (Neri, Weber, Caraffini & Poikolainen 2012), (Poikolainen, Caraffini, Neri & Weber 2012) and (Poikolainen, Iacca, Caraffini & Neri 2013), which are summarised in Chapter 6.1. The second **IRQ** was answered by the study in (Caraffini, Iacca, Neri & Mininno 2012*a*), reported in Chapter 6.2. In order to reply to last **IRQ**, three seriously simple MC algorithms, (Caraffini, Neri, Gongora & Passow 2013), (Caraffini, Neri, Passow & Iacca 2013*a*) and (Iacca, Caraffini, Neri & Mininno 2013) , that can be seen as an extreme case of the of Ockhams Razor in MC, are presented in Section 6.3.

CHAPTER 7 starts with the definition of a new nomenclature for managing basic operators in elementary structures. A novel algorithm, namely PMS, based on the elementary "parallel structure" is proposed and experiments have been carried out in order to empirically prove its structural efficiency and the importance of having a diverse pool of operators. The definition of a general nomenclature for tackling MC algorithms and the idea that robust optimisers can be designed thanks to the parallel memetic structures by simply binding together operators from a pool, is the first step towards the prototype for the automatic generation of optimisation algorithms. These achievements are described in Chapter 7.1, and were published in (Caraffini, Neri, Iacca & Mol 2012). This paper made an important contribution for answering to the main research question of this thesis, but does not propose a general framework for the automatic design. Moreover, the Parallel Memetic Structure can be seen as a general purpose optimiser with a wide range of applicability, and requires few modifications in order to make it able to adapt to the problem at hand. This refinement has been done later, as described in Chapter 7.2, where a general prototype for the automatic generation of optimisers is also defined

and experimentally tested. According to the general scheme, which has been presented in (Caraffini, Neri & Picinali 2014), the software platform for the automatic generation of optimisation algorithms is supposed to be able to analyse multiple features of an optimisation problem, i.e. separability, multi-modality, ill-conditioning and dimensionality, and then perform the design by coherently selecting operators from a given pool and coordinate them together in sequential/parallel structures. It must be remarked that the first implementation of the prototype proposed in (Caraffini et al. 2014) does not completely code the general scheme, and so only partially answer to the main research question. The SPAM algorithm does not take into consideration multiple features at the same time, such as ill-conditioning, dimensionality and modality, but focuses only on the "degree of separability" of the problem at hand (indirectly evaluated in terms of correlation amongst the design variables) . In conclusion, this chapter answers to a part of the general research question, that can be reformulated as:

is it possible to understand the degree of separability of a given black-box continuous problem, and select operators from a pool for automatically generating a tailored algorithm accordingly? **(IRQ IV)**

CHAPTER 8 extends the work in (Caraffini et al. 2014) on the evaluation of the separability coefficient, proposing an alternative way for estimating the correlation amongst the design variables, and links the coefficient with another important features to be taken into consideration for automatically designing an optimisation algorithm: the dimensionality of the problem at hand. The study of the correlation between the variables of an optimisation problem in multiple (increasing) dimensionality values, help us understand how a solver has to be designed in order to face LSOP, which are usually more challenging that their corresponding low dimensional versions, having only "few" design variables to be considered. As a consequence, this chapter investigates the following research question:

What algorithmic mechanism appears promising for tackling large scale optimisation?

In order to get to this point, the research work has gone through the following intermediate research questions:

- ***"what are the most successful strategies for handling LSOP?"*** **(IRQ V)**
- ***"What do these strategies have in common? What algorithmic mechanism appears promising for tackling LSOP? Can we give an explanation to why these approaches are working on LSOP and thus understand when to apply them in the future?"*** **(IRQ VI)**

This study is essential in order to understand how to automatically approach a given problem. The dimensionality of the problem, is indeed a peculiar feature, i.e. it is know a-priori, and heavily impacts on the design of the algorithm. For this reason, the outcome of this last work is precious for the automatic generation of optimisation algorithms.

CHAPTER 9 summarises the final considerations and results achieved on each chapter. Moreover, ideas for future developments are proposed and discussed.

Chapter 2

Principles of optimisation: a literature review of classic local and global search

Every time we have to make a decision (pick a choice) in order to achieve one or more objectives we face an optimisation problem, that is, the search for that feasible solution which corresponds to the extreme of an objective function. It's evident that, according to this definition, almost every problem in the world can be seen as an optimisation problem, since each decision is made in order to achieve a certain goal by following some criteria of optimality. Thus, this discipline can be applied on a broad and heterogeneous variety of issues, making optimisation interdisciplinary, and sometimes extremely complex. Common problems are, for example, in engineering the design of an aircraft, in economics the selection of a mid-term investment, or in medicine the design of a therapy.

Once a problem-based objective function has been obtained, the optimisation process takes place by minimising or maximising the function according to the given necessity (Definition2.1). In this regard, the first scientific method for optimisation can be traced to the creation of "differential calculus" by Newton and Leibniz (17th century), who first introduced the concepts of derivative and gradient. The search for extrema followed with the "calculus of variation" by Euler, Weierstrass and Lagrange (18th-19th century), while probably the most important breakthrough for constraint dynamic optimisation is represented by the method of Lagrange Multipliers. Despite the strong mathematical validity behind these methods, it is impossible to apply such techniques to a vast set of optimisation problems. Indeed, the discontinuity of the physical phenomena, as well as other inconveniences (e.g. a noisy objective function, lack of an analytical expression of the objective function etc.), make the hypotheses of the exact methods unverified in the majority of the real-world cases. Very little progress could be made in tackling such difficulties until the diffusion of powerful digital computers, which have literally represented a turning point for modern optimisation, allowing researchers to develop new paradigms. Many numerical methods have been coded into the digital domain between the '60s and early '70s, see (Rosenbrock 1960), (Powell 1964), (Nelder & Mead 1965) and (Brent 1973) , but only later on, between the '70s and '90s, powerful and robust meta-heuristics, representing the current state-of-the-art of optimisation, have been proposed. During those years, important computational paradigms such as Simulated Annealing, Evolutionary and Swarm Intelligence Algorithms were defined, see for example (Eiben & Smith 2007). These

new methods are versatile and can be used without any hypothesis on the objective function, which can even be unknown, i.e. black box optimisation. This is possible by employing CI methods like machine learning and neural systems, or Artificial Intelligence (AI) techniques that mimic natural processes. We can refer to the discipline which integrates AI into algorithms for solving optimisation problems as CIO.

In order to present the most important CIO frameworks, a rigorous definition of the optimisation problem is first given in Chapter 2.1. It is followed, Chapter 2.2, by a brief introduction and literature review of the most important LS algorithms. Despite the fact that they are mainly from '60s, they still play a crucial role in modern optimisation, and their description is needed in order to better understand the remainder of this thesis.

2.1 Optimisation Problem: a general definition

Regardless of the nature of the application, optimisation problems have a common mathematical statement:

$$\begin{array}{ccc} Maximise/Minimise & f_m(\mathbf{x}) & m = 1, 2, \ldots, M \\ subject - to & g_j(\mathbf{x}) \leqslant 0 & j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \ldots, K \end{array} \tag{2.1}$$

where $f_1, f_2, \ldots, f_M$ is a set of M objective functions, commonly called *fitness* functions, f_m: $\mathcal{D} \longrightarrow \mathbb{R}$. Despite the simplicity of this formulation many different kind of problems arise from Definition 2.1. Starting from the domain $\mathcal{D}$, commonly called the *decision* or *search space*, it is possible to distinguish two main categories of optimisation problems: if this set ($\mathcal{D}$) contains a finite number of discrete solutions we deal with *combinatorial* optimisation, and in the alternative, if it is a "theoretically" infinite set of real numbers we have *continuous* optimisation. In this case, f_m: $\mathcal{D} \in \mathbb{R}^n \longrightarrow \mathbb{R}$, the decision space can be described as the Cartesian product of the intervals (the problem's bounds) where the design variable $\mathbf{x}$ (a vector of n components or design parameters $x_1, x_2, \ldots, x_i, \ldots, x_n$) takes values from, i.e. $[\mathbf{x^L}, \mathbf{x^U}]$.

For the sake of clarity, it must be remarked that the word "continuous" assumes a domain specific meaning in Computer Science, where mathematics is coded into the discrete world of a machine. Even if it is impossible to represent a strictly dense set of elements within a calculator, because of the accuracy of modern electronic devices, it is more superior than the precision requested by several everyday life problems and engineering applications, it is therefore fair to consider a set dense, in Computer Science, when the distance between each pair of consecutive points is not bigger than the machine precision (Neri et al. 2011). In other words, as it is impossible to store an infinite dense set of points, on a digital domain the need arise to consider as "dense" any set containing the maximum amount of points allowed by the precision of the hardware technology employed.

Once one has identified the nature of the problem (discrete/continuous/hybrid), a crucial point for the algorithmic design is the presence of multiple fitness functions. In *Multi-objective Optimisation*, $m \geq 1$, the algorithm must be designed in order to reach a trade-off between the goals (usually conflicting with each other) rather than looking for the global minimum/maximum point of the fitness function, as happens in *Single-objective Optimisation*. Regardless of the number of objectives to satisfy, another important feature of the

problem to be taken into consideration before designing an optimisation algorithm, is the time dependency of the fitness functions. In many cases, the problem under study does not vary its characteristics with time (*static optimisation*), but there are physical processes which evolve dynamically with time, thus causing variations on the fitness function and making its global optimum move within the search space (*dynamic optimisation*).

Finally, the functional g_j and h_k are inequality and equality constraints. Their presence ($j \geq 0 / k \geq 0$) makes the problem more or less severely constrained, otherwise the problem is said to be *unconstrained*.

This research focusses on a subset of CIO, namely **STATIC**, **SINGLE-OBJECTIVE**, **UNCONSTRAINED** and **CONTINUOUS OPTIMISATION**. Without a loss of generality, in the remainder of this thesis only minimisation problems will be considered. Maximisation problems can be indeed handled with a minimiser, and vice-versa, by simply multiplying the fitness value by -1. A final remark must be done on how boundaries are handled. As previously mentioned, the optimisation process takes place within a given search space $\mathcal{D}$, identified with its lower bounds, vector $\mathbf{x}^{\mathbf{L}}$, and upper bounds, vector $\mathbf{x}^{\mathbf{U}}$. Since most optimisers could potentially generate a solution falling out of the search space, every newly generated solution must undergoes a check, and so must be saturated. The most obvious and common ways of managing this scenario is to discard the infeasible solution and generate a new one, or brutally clip it to the lowest, or highest, value admissible. An alternative approach, that has been used for all the algorithms presented in this work, is the toroidal saturation scheme shown in Algorithm 1, which has proven to be beneficial in many cases (Price, Storn & Lampinen 2005). Although not

Algorithm 1 Toroidal Saturation

procedure SATURATETORO($\mathbf{x}$)
  for $i = 0 : n$ **do**
    $\mathbf{x_{sat}}[i] \leftarrow \frac{\mathbf{x}[i] - \mathbf{x^L}[i]}{\mathbf{x^U}[i] - \mathbf{x^L}[i]}$ ▷ Normalisation
    if $\mathbf{x_{sat}} > 1$ **then**
      $\mathbf{x_{sat}}[i] \leftarrow \mathbf{x_{sat}}[i] - fix\left(\mathbf{x_{sat}}[i]\right)$ ▷ $fix\left(\mathbf{x}\right)$: rounds $\mathbf{x}$ to the nearest integer towards zero
    else if $\mathbf{x_{sat}} < 0$ **then**
      $\mathbf{x_{sat}}\left[i\right] \leftarrow 1 - \left|\mathbf{x_{sat}}\left[i\right] - fix\left(\mathbf{x_{sat}}\left[i\right]\right)\right|$
    end if
    $\mathbf{x_{sat}}\left[i\right] \leftarrow \mathbf{x^L} + \mathbf{x_{sat}}\left[i\right] \cdot \left(\mathbf{x^U}[i] - \mathbf{x^L}[i]\right)$ ▷ Rescaling
  end for
  Output $\mathbf{x_{sat}}$
end procedure

mentioned in the proposed pseudo-codes, in order to keep them easy to follow, this procedure is applied after each functional call. If a solution falls in a forbidden region, this transformation toroidally "curves" the space in order to rescale it within $\mathcal{D}$.

2.2 Local Search

LS algorithms are those routines which explore and refine, where possible, a start solution within a subset of the search space, possibly leading to a local optimum. This kind of search plays an important role in modern optimisation, and it is a challenging concept, difficult to introduce. In effect, many techniques originally born as global optimisers are nowadays being used, after a proper tuning of their parameters, as local searchers. This is for instance the case of Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in (Molina, Lozano,

García-Martínez & Herrera 2010*a*), (1+1)-CMA-ES in (Caraffini, Iacca, Neri & Mininno 2012*b*), and many others such as Tabu Search and simulated Annealing (Tirronen, Neri, Kärkkäinen, Majava & Rossi 2008). A detailed description of these algorithms will be given in Chapter 3. In general, a local optimiser perturbs an initial point $\mathbf{p}$ applying a "move operator" having an exploratory magnitude (ϵ) confined within a "small" area surrounding $\mathbf{p}$. The "move" logic adopted helps us define the boundaries of this area, by introducing the concept of neighbourhood $N(\mathbf{p})$ of a solution $\mathbf{p}$. This is usually defined as the set of points from where p can be reached in one step (or a preassigned, reasonably small, number of steps) of the search. In the continuous case this concept is fundamental, in fact, despite what happens in discrete optimisation, it makes sense to talk about "small" movements along preferential directions, or in other words, to talk about the gradient. As for a Cauchy-continuous function, within $N(\mathbf{p})$, all the points are expected to have similar fitness values (no discontinuities), and the partial derivative of the fitness function f for a given generic variable x_i can be evaluated as follows:

$$\frac{\partial f\left(x_i\right)}{\partial x_i} \approx \frac{f\left(x_i+\epsilon\right)-f\left(x_i\right)}{\epsilon} \tag{2.2}$$

Due to this approximation, it can be noticed that a null gradient does not correspond to a critical point, as happens in mathematics, but rather implies that $N(\mathbf{p})$ is a plateau (Neri et al. 2011). Thus, gradient information alone is not sufficient to find a local optimum, but can only be used to detect promising directions, so avoiding unnecessary sectors of the search space. Higher derivatives (Hessian matrix) can also be used to improve the search. If the function is not Cauchy-continuous, i.e. presence of discontinuity points, $N(\mathbf{p})$ can still be considered, but a stochastic trial-and-error search is then essential, since the gradient information in such points becomes inconsistent. According to the information (order of derivative) employed, LS algorithms fall into three main categories:

- *Zeroth-order methods*: no use of higher order property of the fitness function. Also called *derivative-free* or *direct search* methods, they work with ordinal relations between objective function values (Trosset 1997). Methods such as Hooke-Jeeves, Nelder-Mead, Rosenbrock and Powell's Direction Set (described in the following sections) belong to this class, since the information coming from the fitness functional call is not used to approximate the gradient.
- *First-order methods:* based on the gradient vector of the fitness function, either directly accessed, as in Gradient Descent method (Section 2.2.1), or empirically estimated, see Simultaneous Perturbation Stochastic Approximation (SPSA) in Section2.2.7.
- *Second-order methods*: in spite of a growth of the computational cost, these methods rely on the Hessian matrix in order to improve convergence speed and accuracy. This is the case of Newton and Quasi-Newton methods (Section 2.2.1).

For the sake of completeness, it must be said that the proposed taxonomy can be subject to variations in the literature. Some authors, i.e. (Neri et al. 2011), consider derivative free methods as First-order methods. Moreover, they consider Powell's Direction Set algorithm as a Second-order method, since it implicitly makes assumptions on the Hessian matrix (see Section 2.2.6). This fact occurs although in (Powell 1964) it is widely

remarked that, as primary feature, this method does not require any approximation of the gradient.

Conversely, SPSA is presented in the original paper (Spall 1987) like a gradient free method, since it does not require a direct access of the gradient vector. The operating principle of this method, see Section 2.2.7, relies on a stochastic approximation of the gradient rather then a deterministic evaluation, but employing the gradient information can still be argued that SPSA is actually gradient-based. The simple consideration that the gradient will always be necessarily estimated in CIO, either thanks to Formula 2.2 or stochastically, in fact makes all the algorithms using the concept of derivative fall within the second category, in the way it has been defined in this thesis. In the end, it does not make much sense to discriminate between direct or indirect access of the gradient vector, because the fundamental source of information of any optimiser is the fitness function, and any other kind of information bound to derive from that numerical value. Despite both *stochastic* and *deterministic*[1] algorithms being able to make use of the gradient information, it is worth noticing that the former are preferable in noisy or multi-modal environments (high exploration capabilities), while the latter for mono-modal problems (being exploitative and sensible to the initial point). Deterministic LS is usually precise but can be subject to numerical instability, e.g. division by zero, inversion of the Hessian matrix is not possible and so on.

In conclusion, classification can be difficult or often even trivial and the proposed taxonomy has thus been adopted in order to stress the fact that, whether approximated or not, the gradient gives us precious information that can be exploited only in continuous optimisation, in order to privilege certain "lanes" and "short-cuts", rather than trying all the possible paths. For these reasons, this thesis will approach LS from a different point of view, focussing on three main aspects, namely the search logic (type of move performed within the search space), memory and time complexity. These characteristics are, in my opinion, crucial for the algorithmic design of an optimisation algorithm and give us valuable information that can be directly exploited during the design. Conversely, the disquisition on the way the gradient is approximated is often more philosophical than practical. Obviously, properties such as derivative-order and configurations for mathematical convergence (displayed in Table 2.1) have to be taken into account by the careful designer, but play only a secondary role with respect to the move logic spanning the search space. This has to be understood and considered with special care, in particular when combining more techniques as happens in MC. In this case, the search logic cannot be ignored, in order to maintain a certain degree of diversity leading to a balanced algorithm (see Section 7.1) not containing redundant operators (see Section 4.2). This theoretical reflection must also deal with applicative aspects (since optimisation faces real-world problems) such as real-time constraints and memory limitations. The memory footprint of each operator has been reported in Table 2.1. This special attention given to the memory requirements, for both those algorithms from the literature and originally designed, is due to the fact that a relevant number of the considered optimisers are meant for embedded systems with limited memory and computational capacity. As instance, one can see (Iacca, Caraffini & Neri 2013), an interesting case where the entire optimisation process takes place in real-time in a micro-controller with memory limitations, or also (Iacca, Caraffini & Neri 2013) for engineering applications with devices plagued by a modest computational power.

[1] Algorithm free of any form of randomisation logic, performing a predictable sequence of steps.

It follows a detailed description of the most known Local Searchers, which have been used in this work. The start point for the search is here indicated as $\mathbf{x_0}$, referring to the fact that the first value taken as input can be an initial guess but is, more likely, a design variable $\mathbf{x}$ coming from another operator. The index 0 means that the variable is about to undertake the LS process for a certain amount of iterations, e.g. n, and will be returned with its new value $\mathbf{x_n}$ at the end of the procedure.

2.2.1 Newton, Quasi-Newton and Gradient Descent Method

If a function $f : \mathcal{D} \in \mathbb{R}^n \longrightarrow \mathbb{R}$ is twice-differentiable in the neighbourhood of a point $\mathbf{x_0} \in \mathcal{D}$, then it can be approximated by a quadratic model by means of the second order Taylor expansion:

$$f(\mathbf{x}) \simeq f(\mathbf{x_0}) + \boldsymbol{\nabla} f(\mathbf{x_0})(\mathbf{x} - \mathbf{x_0}) + (\mathbf{x} - \mathbf{x_0})^T \frac{1}{2} \boldsymbol{\nabla}^2 f(\mathbf{x_0})(\mathbf{x} - \mathbf{x_0}) \tag{2.3}$$

If $\mathbf{x}^*$ is a critical point of f, then $\boldsymbol{\nabla} f(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}^*} = \emptyset$. This implies, by applying $\boldsymbol{\nabla}$ to both the members in Equation 2.3 and replacing $\mathbf{x} = \mathbf{x}^*$, that:

$$\mathbf{x}^* = -\boldsymbol{\nabla} f(\mathbf{x_0}) \mathbf{H_f^{-1}}(\mathbf{x_0}) + \mathbf{x_0} \tag{2.4}$$

where, for simplifying the notation, $\mathbf{H_f}(\mathbf{x_0})$ is the Hessian matrix of $f(\mathbf{x})\big|_{=\mathbf{x_0}}$. If $\mathbf{H_f}(\mathbf{x}^*)$ is positive definite, $\mathbf{x}^*$ is the minimum of f. Equation 2.4, leads to the iterative formula shown in Algorithm 2, where α can be tuned according to the problem, in order to speed up the convergence process. Where $\mathbf{H_f}(\mathbf{x})$ is equal to the identity matrix, the second order information vanishes, and *Newton's method* becomes the so called *gradient descent*. In addition, if $\mathbf{H_f}(\mathbf{x})$ is locally Lipschitz continuous[2], this method mathematically q-quadratically[3] converges to the local optimum. Unfortunately, this hypothesis does not hold in the majority of real-world cases. Moreover, inverting the Hessian matrix is a computationally expensive (infeasible in some cases) operation. This drawback is overcome by Quasi-Newton methods, such as Davidon-Fletcher-Powell and Broyden-Fletcher-Goldfarb-Shanno, or the Conjugate Gradient method (Press 2007), where $\mathbf{H}^{-1} f(\mathbf{x})$ is directly built-up by exploiting successive gradient vectors. Howsoever, these methods are computationally expensive and their applicability is subject to strong assumptions for uni-modal functions.

2.2.2 Nelder-Mead Method

This is a very popular LS algorithm better known as the "Simplex method". The idea of exploring the search space by simply linearly-combining $n + 1$ points, forming a simplex, was first introduced in (Spendley, Hext & Himsworth 1962) and then completed in (Nelder & Mead 1965). In the former implementation, the simplex could only vary in size, but not in shape, thanks to two transformations *reflection* and *shrinkage*. Conversely,

[2] $\forall \mathbf{x_1}, \mathbf{x_2} \in \mathcal{D}, \exists\, C \in \mathbb{R}^+ + \{0\} \mid \|\mathbf{H_f}(\mathbf{x_1}) - \mathbf{H_f}(\mathbf{x_2})\| \leqslant C\|\mathbf{x_1} - \mathbf{x_2}\|$

[3] $\mathbf{x_k}$ q-quadratically converge to $\mathbf{x}^*$ if $\exists\, C \in \mathcal{R}^+ \mid \|\mathbf{x_{k+1}} - \mathbf{x}^*\| \leqslant C\|\mathbf{x_k} - \mathbf{x}^*\|^2$

Algorithm 2 Newton's Method

procedure NEWTON($\mathbf{x_0}$)
  $k \leftarrow 0$
  while *stop condition not met* **do** ▷ usually $\|\nabla f(\mathbf{x_k})\|$ or $\|\mathbf{x_k} - \mathbf{x_{k-1}}\| \backsimeq 0$
    $\mathbf{N_k} \leftarrow \boldsymbol{\nabla} f(\mathbf{x_k}) \, \mathbf{H_f^{-1}}(\mathbf{x_k})$ ▷ N_k: Newton's step at iteration k
    $\mathbf{x_{k+1}} \leftarrow -\alpha \mathbf{N_k} + \mathbf{x_k}$ ▷ α: step control parameter
    $k \leftarrow k + 1$
  end while
  Output $\mathbf{x_k}$
end procedure

in the latter, due to two additional stages, *expansion* and *contraction*, it can change its shape so adapting to the local landscape and slanting towards the local optimum. At the beginning of the optimisation, for a given n-dimensional problem, $n+1$ vertices are generated and ordered according to their fitness value. It is thus possible to identify the point with the lowest fitness value $\mathbf{x_l}$ (best point), the second worst $\mathbf{x_s}$ and the one with the highest value (worst point) $\mathbf{x_h}$. All the new vertices generated by this algorithm lie on the lines between these points and the centroid of the simplex $\bar{\mathbf{x}}$, and are evaluated by employing the aforementioned transformations as thoroughly reported in Algorithm 3. The proposed pseudo-code refers to a more modern implementation than the one provided by Nelder and Mead, where two different contractions, namely "Outside" and "Inside" Contractions (Lagarias, Reeds, Wright & Wright 1998), are being used. Despite this minor change, the main idea is the same: each iteration tries to replace $\mathbf{x_h}$ with a better vertex by reflecting the new point ($\mathbf{x_r}$) away from the previous, or expanding a too small simplex by generating $\mathbf{x_e}$, or also with a contraction ($\mathbf{x_c}$) toward a position confined between $\mathbf{x_r}$ and $\mathbf{x_h}$. If one of these new points succeeds, then it replaces $\mathbf{x_h}$, otherwise the simplex is shrunk in order to focus on a smaller area of the research space.

The transformations involved in this process are very simple and computationally cheap. Their coordination, displayed in Algorithm 3, can appear intricate at first glance, but it is based on an if/else logic that can be easily coded. The main drawback of this method is the importance in the choice for the initial simplex. A good configuration can lead to the optimal solution in few iterations. Conversely, an inadequate initial simplex can compromise the entire process. Unfortunately, in many real-world applications no a-priori information is given to know about the problem (black box system), making this choice difficult. Among the possible ways for sampling vertices, the most commonly used generates n points around an initial guess $\mathbf{x_0}$ by adding a quantity h to each dimension. In this way, the resulting simplex will surely be *non degenerate*, i.e. the vertices do not belong to the same hyperspace. In this case, for a strictly convex problem in one or two dimensions, this method has been proven to converge to a local optimum (Lagarias et al. 1998). A general proof of convergence for higher dimensions does not exist.

Algorithm 3 Nelder-Mead Method

procedure NELDER-MEAD($\mathbf{x_0}$)
  for $i = 1 : n$ **do**
    $\mathbf{x_i} \leftarrow \mathbf{x_0} + h_i \mathbf{e_i}$ ▷ $\mathbf{e_i}$: unit vector in $\mathbb{R}^n$, the size of the neighbourhood can be tuned via h_i
    $i \leftarrow i + 1$
  end for
  while *stop condition not met* **do** ▷ max budget or $\sqrt{\sum_{i \neq h} \frac{(f(\mathbf{x_i}) - f(\bar{\mathbf{x}}))^2}{n}} < 10^{-8}$ (Nelder & Mead 1965)
    Ordering
    $\mathbf{x_l} \leftarrow arg\{min_i f(\mathbf{x_i})\}$
    $\mathbf{x_h} \leftarrow arg\{max_i f(\mathbf{x_i})\}$
    $\mathbf{x_s} \leftarrow arg\{max_i f(\mathbf{x_{i \neq h}})\}$
    Centroid
    $\bar{\mathbf{x}} \leftarrow \frac{1}{n} \sum_{i \neq h} \mathbf{x_i}$
    Transformation
    $\mathbf{x_r} \leftarrow \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x_h})$ ▷ Reflection: $\alpha > 0$, usually set to 1
    if $f(\mathbf{x_l}) \leq f(\mathbf{x_r}) < f(\mathbf{x_s})$ **then**
      $\mathbf{x_h} \leftarrow \mathbf{x_r}$
    else if $f(\mathbf{x_r}) < f(\mathbf{x_l})$ **then**
      $\mathbf{x_e} \leftarrow \bar{\mathbf{x}} + \gamma(\mathbf{x_r} - \bar{\mathbf{x}})$ ▷ Expansion: $\gamma > \alpha$, usually set to 2
      if $f(\mathbf{x_e}) < f(\mathbf{x_r})$ **then**
        $\mathbf{x_h} \leftarrow \mathbf{x_e}$
      else
        $\mathbf{x_h} \leftarrow \mathbf{x_r}$
      end if
    else if $f(\mathbf{x_s}) \leq f(\mathbf{x_r}) \leq f(\mathbf{x_h})$ **then**
      $\mathbf{x_c} \leftarrow \bar{\mathbf{x}} + \beta(\mathbf{x_r} - \bar{\mathbf{x}})$ ▷ Outside contraction: $0 < \beta < 1$, usually set to 0.5
      if $f(\mathbf{x_c}) \leq f(\mathbf{x_r})$ **then**
        $\mathbf{x_h} \leftarrow \mathbf{x_c}$
      else
        for $i = 0 : n$ **do**
          $\mathbf{x_i} \leftarrow \mathbf{x_l} + \delta(\mathbf{x_i} - \mathbf{x_l})$ ▷ Shrinkage: $0 < \delta < 1$, usually set to 0.5
          $i \leftarrow i + 1$
        end for
      end if
    else if $f(\mathbf{x_r}) \geq f(\mathbf{x_h})$ **then**
      $\mathbf{x_c} \leftarrow \bar{\mathbf{x}} + \beta(\bar{\mathbf{x}} - \mathbf{x_h})$ ▷ Inside contraction
      if $f(\mathbf{x_c}) < f(\mathbf{x_h})$ **then**
        $\mathbf{x_h} \leftarrow \mathbf{x_c}$
      else
        for $i = 0 : n$ **do**
          $\mathbf{x_i} \leftarrow \mathbf{x_l} + \delta(\mathbf{x_i} - \mathbf{x_l})$ ▷ Shrinkage
          $i \leftarrow i + 1$
        end for
      end if
    end if
  end while
  Output $\mathbf{x_l}$
end procedure

2.2.3 Hooke-Jeeves Method

The Hooke-Jeeves method (Hooke & Jeeves 1961), is a single-solution deterministic procedure employing two simple perturbation logics: "exploratory" and "pattern" move. The first one, systematically perturbs one dimension at time by adding a certain quantity δ both onwards and, if needed, backwards. Conversely, the latter speeds-up the search process by performing a "jump" towards the direction of the best improvement. The pseudo-code in Algorithm 4 explains this process in a more formal way. The logic is as simple as it is efficient:

Algorithm 4 Hooke-Jeeves Method

procedure HOOKE-JEEVES($\mathbf{x_0}$)
  $\mathbf{x_{hj}} \leftarrow x_0$
  $\mathbf{x_{old}} \leftarrow x_0$
  while *stop condition not met* **do** ▷ max budget or f does not decrease
    Exploratory move
    for $i = 1 : n$ **do**
      $\mathbf{x_{trial}}[i] \leftarrow \mathbf{x_{hj}}[i] - \delta$ ▷ δ is a percentage of the search space size
      if $f(\mathbf{x_{trial}}) \leq f((\mathbf{x_{hj}})$ **then**
        $\mathbf{x_{hj}} \leftarrow \mathbf{x_{trial}}$
      else
        $\mathbf{x_{trial}}[i] \leftarrow \mathbf{x_{hj}}[i] + \delta$
        if $f(\mathbf{x_{trial}}) \leq f(\mathbf{x_{hj}})$ **then**
          $\mathbf{x_{hj}} \leftarrow \mathbf{x_{trial}}$
        end if
      end if
      $i \leftarrow i + 1$
    end for
    if $f(\mathbf{x_{hj}}) == f(\mathbf{x_{old}})$ **then**
      $\delta \leftarrow \frac{\delta}{d}$ ▷ $d > 0$, decreasing factor
    else
      pattern move
      $\mathbf{x_{trial}} = \mathbf{x_{hj}} + \alpha(\mathbf{x_{hj}} - \mathbf{x_{old}})$ ▷ $\alpha > 0$, speed-up factor
      if $f(\mathbf{x_{trial}}) \leq f(\mathbf{x_{hj}})$ **then**
        $\mathbf{x_{hj}} \leftarrow \mathbf{x_{trial}}$
        $\mathbf{x_{old}} \leftarrow \mathbf{x_{hj}}$
      end if
    end if
  end while
  Output $\mathbf{x_{hj}}$
end procedure

if the exploratory move fails δ is decreased, otherwise a pattern search is performed and, if successful, the newly generated point is the new start point for the next iteration. The magnitude for the pattern search move can be tuned via a parameter $alpha$: the higher its value, the longer the step performed along the line between the previous solution and the current (best) point. Being purely deterministic, the main drawback is the risk of getting stuck on a local optimum. If the problem is multi-modal, its performance highly relies on the initial solution.

2.2.4 Short Distance Exploration (S)

First introduced in (Tseng & Chen 2008) and then readjusted in (Iacca, Neri, Mininno, Ong & Lim 2012) with the name "Short Distance Exploration", or simply S, this operator is a powerful greedy-descent method

employing a perturbation logic derived from Hooke-Jeeves (Section 2.2.3). As shown in Algorithm 5, it executes an asymmetric sequence of steps along the axes: for each dimension ($i = 0, 1, \dots n$) a full step $\boldsymbol{\delta}[i]$ is taken, if it does not bring any improvement then a half step is performed on the opposite direction. Not employing the "pattern" move, as happens in Hook-Jeeves, this unbalanced search minimises the risk of visiting the same point twice. This mechanism is continued until the computational budget allocated expires. According

Algorithm 5 Short Distance Exploration

procedure S($\mathbf{x_0}$)
  $\mathbf{x_s} \leftarrow \mathbf{x_0}$
  $\mathbf{x_{old}} \leftarrow \mathbf{x_0}$
  for $i = 1 : n$ **do**
    $\boldsymbol{\delta}[i] = \alpha \left(\mathbf{x^U}[i] - \mathbf{x^L}[i]\right)$ ▷ $\alpha \in]0, 1]$, suggested value: 0.4 (Tseng & Chen 2008)
    $i \leftarrow i + 1$
  end for
  while *stop condition not met* **do** ▷ Condition on budget or equation 2.5
    for $i = 1 : n$ **do**
      $\mathbf{x_{trial}}[i] \leftarrow \mathbf{x_s}[i] - \boldsymbol{\delta}[i]$
      if $f(\mathbf{x_{trial}}) \leq f(\mathbf{x_s})$ **then**
        $\mathbf{x_s} \leftarrow \mathbf{x_{trial}}$
      else
        $\mathbf{x_{trial}}[i] = \mathbf{x_s}[i] + \frac{\delta}{2}$
        if $f(\mathbf{x_{trial}}) \leq f(\mathbf{x_s})$ **then**
          $\mathbf{x_s} \leftarrow \mathbf{x_{trial}}$
        end if
      end if
      $i \leftarrow i + 1$
    end for
    if $f(\mathbf{x_s}) == f(\mathbf{x_{old}})$ **then**
      $\boldsymbol{\delta} \leftarrow \frac{\boldsymbol{\delta}}{2}$
    end if
  end while
  Output $\mathbf{x_s}$
end procedure

to (Iacca, Neri, Mininno, Ong & Lim 2012) 150 iterations of the aforementioned procedure are sufficient to reach a good precision, then the exploratory radius has to be reinitialised (Algorithm 5 has to be restarted) because the search would not lead to significant improvements. For this reason , a different stop criterion has been adopted in (Caraffini, Neri, Passow & Iacca 2013*a*), where the search along the axes (depicted in Figure 2.1) is interrupted only when the exploratory radius $\boldsymbol{\delta}$ becomes too small. More formally, the operator is stopped and restarted when the 2-norm of $\boldsymbol{\delta}$, normalized per each element by the corresponding search interval, is smaller than a prefixed threshold, as follows:

$$\frac{1}{\sqrt{n}} \cdot \sqrt{\sum_{i=1}^{n} \left(\frac{\boldsymbol{\delta}[i]}{\mathbf{x^U}[i] - \mathbf{x^L}[i]} \right)^2} < \varepsilon \tag{2.5}$$

In this way, the hyper-rectangular search space (employed in the majority of the benchmarks), the threshold ε can be seen as a scale factor for the search space width. S search logic has shown to be generally valid, see Section 4.2, and particularly efficient in dealing with separable and LSOP, see Chapter 7.2 and Chapter 8 respectively.

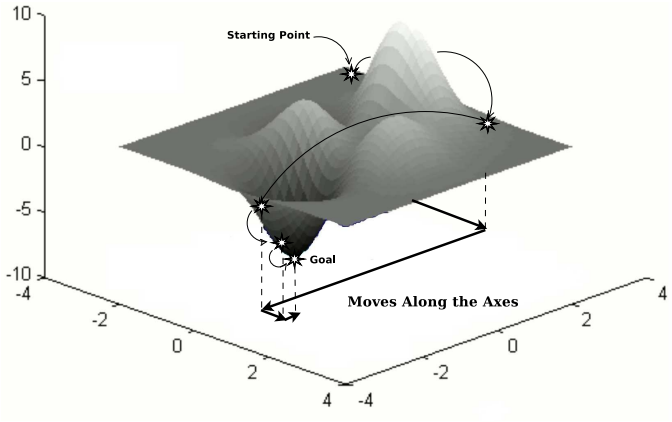


Figure. 2.1. Graphical representation of S search logic.

2.2.5 Rosenbrock Method

The Rosenbrock algorithm is a classical deterministic local search proposed in (Rosenbrock 1960) which, under specific conditions, has been proven to always converge to a local optimum (Bazaraa, Sherali & Shetty 2006).It works on a set of n orthogonal directions $\boldsymbol{\xi_1}, \boldsymbol{\xi_2} \ldots, \boldsymbol{\xi_n}$ which are rotated at each major step, so that at least one of the new directions is more closely conformed to the local behaviour of the function to be optimised. At the beginning of the optimisation $\boldsymbol{\xi}$ is an $n \times n$ identity matrix, thus performing steps along the n axes as follows:

$$\mathbf{x_r} = \mathbf{x_k} + \mathbf{d}[i]\boldsymbol{\xi}_i \quad i = 1, 2, \ldots, n \tag{2.6}$$

where $\mathbf{d}$ is an n-dimensional vector which is updated after every perturbation. If the search along the i-th direction has been successful, $\mathbf{d}[i]$ is amplified by a factor $\alpha > 0$ (bigger step forwards), conversely is multiplied by $-\beta \in]0, 1[$ (step backward). All the successful coefficients $\mathbf{d}[i]$ are summed up and stored in vector $\boldsymbol{\lambda}$ into the corresponding position $\lambda[i]$. At least one direction has to bring an improvement to restart this process, which is on the contrary interrupted. In this case, if $\mathbf{d}$ has reached a too small magnitude to make an improvement, or the distance between the old and the newly found solution is not relevant, a restart is needed ($\min(|\mathbf{d}|) > \varepsilon \; OR \; \min(|\mathbf{x_s} - \mathbf{x_0}|) > \varepsilon$, Line 25 in Algorithm 6). The algorithm goes back to the initial prefixed value of $\mathbf{d}$, updates $\boldsymbol{\xi}$, and explores the new set of direction through 2.6. This time, due to a change of the coordinate system, a diagonal move is performed within the search space, as shown in Figure 2.2. The rotation, according to the original method proposed by Rosenbrock, takes place in two steps. First, a matrix $\mathbf{A}$ must be filled with n vectors $\mathbf{A_i} = \sum_{k=i}^{n} \boldsymbol{\lambda}[k]\boldsymbol{\xi_k}$, then the new direction can be found with $\boldsymbol{\xi_i^{new}} = \frac{\mathbf{B_i}}{|\mathbf{B_i}|}$, where $\mathbf{B_i} = \mathbf{A_i} - \sum_{j=1}^{n-1} \left(\mathbf{A_i}\boldsymbol{\xi_j^{new}}\right)\boldsymbol{\xi_j^{new}}$. Since the last formula can lead to numerical instability in many problems, an improved version proposed in (Palmer 1969) has been used in this thesis, whose detailed implementation can be found in Algorithm 6.

Algorithm 6 Rosenbrock Method

procedure ROSENBROCK($\mathbf{x_0}$)
  $\mathbf{x_k} \leftarrow \mathbf{x_0}$
  $\mathbf{d} \leftarrow \frac{1}{10} ones(n)$ ▷ n-dimensional vector of 0.1
  $\boldsymbol{\lambda} \leftarrow \emptyset$
  $\boldsymbol{\xi} \leftarrow eye\,(n, n)$ ▷ $n \times n$ identity matrix
  $y^{''} \leftarrow f\,(x_r)$
  while *stop condition not met* **do**
    $y \leftarrow y^{''}$
    do
      $y^{'} \leftarrow y$
      for $i = 0 : n$ **do**
        $\mathbf{x_r} \leftarrow \mathbf{x_k} + \mathbf{d}[i]\boldsymbol{\xi}_i$
        if $f\,(\mathbf{x_r}) \leq y$ **then**
          $\boldsymbol{\lambda}[i] \leftarrow \boldsymbol{\lambda}[i] + \mathbf{d}[i]$
          $\mathbf{d}[i] \leftarrow \alpha \cdot \mathbf{d}[i]$ ▷ $\alpha = 3$ in (Rosenbrock 1960), 2 in (Caraffini, Iacca, Neri & Mininno 2012*a*)
          $\mathbf{x_k} \leftarrow \mathbf{x_r}$
          $y \leftarrow f\,(\mathbf{x_r})$
        else
          $\mathbf{d}[i] \leftarrow -\beta \cdot \mathbf{d}[i]$ ▷ $\beta = 0.5$ (Rosenbrock 1960)
        end if
        $i \leftarrow i + 1$
      end for
    while $y < y^{'}$
    if $y < y^{''}$ **then**
      if $\min(|\mathbf{d}|) > \varepsilon\, OR\, \min(|\mathbf{x_s} - \mathbf{x_0}|) > \varepsilon$ **then** ▷ $\varepsilon = 10^{-5}$ (Rosenbrock 1960)
        $\mathbf{A_n} \leftarrow \boldsymbol{\lambda} \circ \boldsymbol{\xi_n}$
        for k=n-1:1 **do**
          $\mathbf{A_k} \leftarrow \mathbf{A_{k+1}} + \boldsymbol{\lambda} \circ \boldsymbol{\xi_k}$
          $k \leftarrow k - 1$
        end for
        $\mathbf{t}[n] \leftarrow \boldsymbol{\lambda}[n]^2$
        for i=n-1:1 **do**
          $\mathbf{t}[i] \leftarrow \mathbf{t}[i+1] + \boldsymbol{\lambda}[i]^2$
          $i \leftarrow i - 1$
        end for
        for i=n:2 **do**
          $\left(\xi_\mathbf{i} \leftarrow \boldsymbol{\lambda_{i-1}} \circ \mathbf{A_i} - \boldsymbol{\xi_{i-1}} \circ \mathbf{t}\right) / \sqrt{\mathbf{t}[i-1]\mathbf{t}[i]}$
          $i \leftarrow i - 1$
        end for
        $\xi_\mathbf{1} \leftarrow \mathbf{A_1} / \sqrt{\mathbf{t}[1]}$
        $\mathbf{x_0} \leftarrow \mathbf{x_k}$
        $\boldsymbol{\lambda} \leftarrow \emptyset$
        $\mathbf{d} \leftarrow \frac{1}{10} ones(n)$
        $y^{''} \leftarrow y$
      end if
    end if
  end while
  Output $\mathbf{x_r}$
end procedure

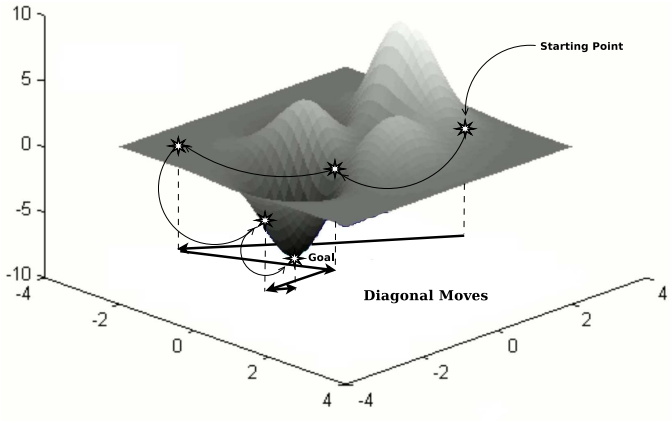


Figure. 2.2. Graphical representation of Rosenbrock search logic.

2.2.6 Powell's Direction Set Method

The conjugate direction set Powell algorithm is a derivative-free local searcher based on the idea of using a set of non-interfering directions ($\boldsymbol{\xi} = [\boldsymbol{\xi_1}, \boldsymbol{\xi_2}, \ldots, \boldsymbol{\xi_n}]$) to search for a minimum in ill-conditioned functions. The procedure proposed by Powell in (Powell 1964), makes use of a generic search method to minimise the function along a single direction, e.g. common choices are the Brent or the Golden Section Search (GSS) methods (Press 2007). According to the original description, $\boldsymbol{\xi}$ can be initialised as an $n \times n$ identity matrix, $\mathbf{P_0}$ denotes the initial guess, and the basic procedures to be iterated consist of the following steps:

1. $\forall i = 1, 2, \ldots, n$ $\mathbf{P_i} = \lambda + \boldsymbol{\xi_i}$ with $\lambda \,\big|\, f\left(\mathbf{P_{i-1}} + \lambda \boldsymbol{\xi_i}\right)$ is minimised.
2. $\forall i = 1, 2, \ldots, n-1$ $\xi_\mathbf{i} = \xi_{\mathbf{i+1}}, \xi_\mathbf{n} = \mathbf{P_n} - \mathbf{P_0}$.
3. Replace $\mathbf{P_0}$ with $\mathbf{P_n} - \lambda \boldsymbol{\xi_n}$ with $\lambda \,\big|\, f\left(\mathbf{P_{i-1}} + \lambda \boldsymbol{\xi_i}\right)$ is minimised.

Powell introduced this method in 1964 for quadratic forms, since in this case k iterations of the above basic procedure produce a set of directions whose last k members are mutually conjugate. Therefore, after n iterations (equivalent to $n\,(n+1)$ line minimisations) it exactly minimises a quadratic form. Unfortunately, it has been noticed that this procedure of replacing ξ_0 at each iteration could eventually lead to a set of linear dependent vectors, thus exploring the same direction multiple times and converging to the wrong solution. This flaw is likely to happen when the dimensionality of the problem is bigger than 5 variables (this is a strong limitation considering the complexity of current real-world problems). Algorithm 7 refers to a second version that, despite giving up to the quadratic convergence property, improves upon the first framework by decreasing the probability of having linearly dependent directions. As a consequence, the resulting algorithm is more robust and applicable to a vast set of problems. The directions can adapt to the fitness function, so allowing the search through diversified landscapes, not necessarily quadratic forms. The $\boldsymbol{\xi}$ matrix is here updated by means of a heuristic that discards the old direction along which f made its largest decrease (Δf). This is the most

Algorithm 7 Powell's Direction Set method

procedure POWELL($\mathbf{x_0}$)
  $\mathbf{P_0} \leftarrow \mathbf{x_0}$
  $f_{ret} \leftarrow (\mathbf{x_0})$
  $\mathbf{x_p} \leftarrow \mathbf{x_0}$
  $\boldsymbol{\xi} \leftarrow eye(n)$
  while *stop condition not met* **do** ▷ Until f stops decreasing
    $f_0 \leftarrow f_{ret}$
    $\Delta f \leftarrow 0$
    $i_\Delta \leftarrow 0$
    for $i = 1 : n$ **do**
      $f_{ptt} \leftarrow f_{ret})$
      $\mathbf{P} \leftarrow lineMin\left(\mathbf{P_{i-1}}, \xi_\mathbf{i}\right)$
      $f_{ret} \leftarrow f\left(\mathbf{P}\right)$
      if $f_{ret} < f\left(x_p\right)$ **then**
        $\mathbf{x_p} \leftarrow \mathbf{P}$
      end if
      if $f_{ptt} - f_{ret} \geqslant \Delta f$ **then**
        $\Delta f \leftarrow f_{ptt} - f_{ret}$
        $i_{Delta} \leftarrow i$
      end if
      $i \leftarrow i + 1$
    end for
    $f_E \leftarrow (2\mathbf{P} - \mathbf{P_0})$
    if $f_E < f\left(x_p\right)$ **then**
      $\mathbf{x_p} \leftarrow 2\mathbf{P} - \mathbf{P_0}$
    end if
    if $f_E < f_0$ **then**
      if $2 \cdot (f_0 - 2 \cdot f_{ret}) \cdot (f_0 - f_{ret})^2 - \Delta f \cdot (f_0 - f_E)^2 < 0$ **then**
        $\mathbf{P} \leftarrow lineMin\left(\mathbf{P}, \mathbf{P} - \mathbf{P_0}\right)$
        $f_{ret} \leftarrow f\left(\mathbf{P}\right)$
        if $f_{ret} < f\left(x_p\right)$ **then**
          $\mathbf{x_p} \leftarrow \mathbf{P}$
        end if
        $\boldsymbol{\xi_{i_\Delta}} \leftarrow \boldsymbol{\xi_n}$
        $\boldsymbol{\xi_n} \leftarrow \mathbf{P} - \mathbf{P_0}$
      end if
    end if
  end while
  Output $\mathbf{x_p}$
end procedure

promising among the directions, and dismissing it could seem inappropriate, but for this reason it is most likely to lead to premature convergence. In addition, two more conditions, see Algorithm 7 line 27 and 28, prevent the matrix from being unnecessarily updated. For the sake of convenience, the generic line minimisation method generating the new point $\mathbf{P_i}$ by moving $\mathbf{P_{i-1}}$ along $\boldsymbol{\xi_i}$ will be addressed via the routine $lineMin\left(\mathbf{P_{i-1}}, \boldsymbol{\xi_i}\right)$, which returns the local minimum P_i. A detailed implementation of this algorithm is given in Algorithm 7. Line Search techniques can be found in (Press 2007).

2.2.7 Simultaneous perturbation stochastic approximation

SPSA is a gradient based optimiser where the standard finite-difference gradient approximation is replaced with a stochastic equivalent (Spall 1987), thus making it computationally less expensive than deterministic derivative based methods and numerically stable. The main idea is to make an estimate of the gradient by simultaneously and randomly varying all of the variables of the problem under study, rather than perturbing one dimension at time as it happens in finite-difference based algorithms. Only two fitness functional calls are required in one iteration. The new point is generated via a perturbation vector $\boldsymbol{\Delta_k}$ whose component must be sampled from a zero-mean distribution, e.g. $Bernoulli \pm 1$ with probability 0.5. If f and all the elements of $\boldsymbol{\Delta_k}$ satisfy all the conditions in (Spall 1992), then the procedure converges to the local optimum, otherwise return a sub-optimal point. For the sake of clarity, it must be remarked that the $Bernoulli \pm 1$ distribution is the one suggested by James Spall in the original implementation, but many other can be chosen. Anyway, normal and uniform distribution should be avoided as they have infinite inverse moment, i.e. inversion could lead to numerical instability. The perturbation vector is used to approximate the gradient as shown in Algorithm 8. The proposed pseudo-code follows the implementation given in (Spall 1998). This is one of the most widely used stochastic

Algorithm 8 SPSA

procedure SPSA($\mathbf{x_0}$)
  $k \leftarrow 0$
  $\boldsymbol{\theta_k} \leftarrow \mathbf{x_0}$
  while *stop condition not met* **do** ▷ Max budget or f does not significantly decrease
    $c_k \leftarrow \frac{c}{(k+1)^{\gamma}}$ ▷ $\gamma = 0.101$ in(Spall 1998)
    $a_k \leftarrow \frac{a}{(k+A+1)^{\alpha}}$ ▷ $\alpha = 0.602$, "Stability factor" $A = 10\%$ of allowed iterations (Spall 1998)
    for i=1:n **do**
      $\boldsymbol{\Delta_k}[i] \leftarrow \mathcal{B}^{\pm 1}\left(0.5\right)$ ▷ i.e. $Prob(\pm 1) = 0.5$
    end for
    $\boldsymbol{\theta}_{\boldsymbol{k}}^{+} \leftarrow \boldsymbol{\theta_k} + c_k \boldsymbol{\Delta_k}$
    $\boldsymbol{\theta}_{\boldsymbol{k}}^{-} \leftarrow \boldsymbol{\theta_k} - c_k \boldsymbol{\Delta_k}$
    $\hat{\mathbf{g}}_{\mathbf{k}} \leftarrow \frac{f\left(\boldsymbol{\theta}_{\boldsymbol{k}}^{+}\right) - f\left(\boldsymbol{\theta}_{\boldsymbol{k}}^{-}\right)}{2c_k} \begin{bmatrix} 1/\boldsymbol{\Delta_k}[1] \\ 1/\boldsymbol{\Delta_k}[2] \\ \vdots \\ 1/\boldsymbol{\Delta_k}[n] \end{bmatrix}$ ▷ Stochastic gradient
    $k \leftarrow k + 1$
    $\boldsymbol{\theta_k} \leftarrow \boldsymbol{\theta_{k-1}} - a_k \hat{\mathbf{g}}_{\mathbf{k}}$
  end while
  Output $\boldsymbol{\theta_k}$
end procedure

gradient-based heuristics. For further details and a comprehensive report on stochastic gradient approximation one can consider (S. Bhatnagar & Prashanth. 2003).

2.2.8 Solis-Wets Method

Solis and Wets algorithm is a randomised hill-climber making use of a framework proposed in (Solis & Wets 1981). The main idea is to generate points by sampling them from a multivariate normal distribution centred close to the current best solution with covariance matrix $\rho\mathbf{I}$. ρ acts as an exploratory radius, determining the diameter of the neighbourhood in which the new point $\boldsymbol{\xi}$ can fall. The exploratory radius can be expanded or contracted according to the number of successive successful or unsuccessful samplings. This logic, has been theoretically proven to provide a sequence that, if not stopped (infinite budget), leads to a local optimum (see original paper). With reference to Algorithm 9, it can be notice that $\boldsymbol{\xi}$ is the linear composition of three vectors.

Algorithm 9 Solis-Wets Method

procedure SOLIS-WETS($\mathbf{x_0}$)
  $\mathbf{x_{sw}} \leftarrow \mathbf{x_0}$
  $\rho \leftarrow 1$
  $\#S \leftarrow 0$ ▷ Number of successive successes
  $\#F \leftarrow 0$ ▷ Number of successive failures
  $\mathbf{b} \leftarrow \emptyset$
  while *stop condition not met* **do** ▷ Fixed computational budget or ρ smaller than a threshold (accuracy)
    $\boldsymbol{\xi} \leftarrow \mathbf{x_{sw}} + \mathbf{b} + \mathcal{N}(0, \rho\mathbf{I})$
    if $f(\boldsymbol{\xi}) < f(\mathbf{x_{sw}})$ **then**
      $\mathbf{b} \leftarrow 0.4(\xi - x_{sw}) + 0.2\mathbf{b}$
      $\mathbf{x_{sw}} \leftarrow \boldsymbol{\xi}$
      $\#S \leftarrow \#S + 1$
      $\#F \leftarrow 0$
    else if $f(2\mathbf{x_{sw}} - \boldsymbol{\xi}) < f(\mathbf{x_{sw}})$ **then**
      $\mathbf{b} \leftarrow \mathbf{b} - 0.4(\boldsymbol{\xi} - \mathbf{x_{sw}})$
      $\mathbf{x_{sw}} \leftarrow 2\mathbf{x_{sw}} - \xi$
      $\#S \leftarrow \#S + 1$
      $\#F \leftarrow 0$
    else
      $\#S \leftarrow 0$
      $\#F \leftarrow \#F + 1$
    end if
    if $\#S > S_{ex}$ **then** ▷ Threshold S_{ex}: 5 in (Solis & Wets 1981)
      $\rho \leftarrow ex \cdot \rho$ ▷ "expand" factor ex: 2 in (Solis & Wets 1981)
    else if $\#F > F_{ct}$ **then** ▷ Threshold F_{ct}: 3 in (Solis & Wets 1981)
      $\rho \leftarrow ct \cdot \rho$ ▷ "contract" factor ct: 0.5 in (Solis & Wets 1981)
    end if
  end while
  Output $\mathbf{x_{sw}}$
end procedure

The last one is a sample from a normal distribution whose mean value consists of a shifted version, due to the bias vector $\mathbf{b}$, of the current solution being processed. Once $\boldsymbol{\xi}$ has been generated, if unsuccessful, a step in the opposite direction is taken $(2\mathbf{x_{sw}} - \boldsymbol{\xi})$ and evaluated. Then, the bias factor is accordingly updated in order to drive the next sampling in favour of the most promising directions. This method relies on a simple logic, with a low memory footprint (there is no need to store the whole covariance matrix, but only ρ) and low computational cost. All the design variables are simultaneously perturbed, i.e. the functional call is performed after n sampling, thus making this method suitable for non-separable landscapes.

The Solis and Wets' performances decrease when the dimensionality of the problem grows. Due to the need of employing fast and light algorithms in tackling LSOP, an improved "large-scale" variant called Sub-grouping Solis-Wets has been proposed in (Molina, Lozano & Herrera 2010). This version features the simplicity of the

classical Solis-Wets combined with a mechanism that splits the problem under study into sub-problems. As shown in Algorithm 10, a subset of design variables is stochastically chosen and grouped before being optimised by Solis-Wets. In this way, the resulting optimiser (Algorithm 11) faces only low dimensional sub-problems.

Algorithm 10 SetSubSet

procedure SETSUBSET(maxVars)
  $iniVar \leftarrow \mathcal{I}(1, n)$ ▷ Random integer number uniformly distributed in $[0, n] \subset \mathbb{N}$
  $numVars \leftarrow 0.2 \cdot n$
  $\mathbf{subSet} \leftarrow \emptyset$
  if $numVars \geq maxVars$ **then**
    $numVars \leftarrow maxVars$
  end if
  for $i = iniVar : (iniVar + numVars - 1)$ **do**
    $\mathbf{subSet}\,[i\%n] \leftarrow 1$
  end for
  Output $\mathbf{subSet}$
end procedure

Algorithm 11 Subgrouping-Solis-Wets Method

procedure SSW($\mathbf{x_0}$)
  $\mathbf{x_{sw}} \leftarrow \mathbf{x_0}$
  $\rho \leftarrow 1$
  $\#S \leftarrow 0$ ▷ Number of successive successes
  $\#F \leftarrow 0$ ▷ Number of successive failures
  $\mathbf{b} \leftarrow \emptyset$
  while $j < budget$ **do**
    $\mathbf{change} \leftarrow \emptyset$
    if $j\%maxEvalSubSet$ **then** ▷ $maxEvalSubSet = \frac{total\ budget}{10}$ (Molina, Lozano & Herrera 2010)
      $\mathbf{change} \leftarrow$SETSUBSET($maxVars$) ▷ Algorithm 10, $maxVars = 50$ (Molina, Lozano & Herrera 2010)
    end if
    for $i = 1 : n$ **do**
      if $\mathbf{change}\,[i]$ **then**
        $\boldsymbol{\xi}\,[i] \leftarrow \mathbf{x_{sw}}\,[i] + \mathbf{b}\,[i] + \mathcal{N}(0, \rho)$
      end if
      $i \leftarrow i + 1$
    end for
    $j \leftarrow j + 1$
    if $f(\boldsymbol{\xi}) < f(\mathbf{x_{sw}})$ **then**
      $\mathbf{b} \leftarrow 0.4\,(\xi - x_{sw}) + 0.2\mathbf{b}$
      $\mathbf{x_{sw}} \leftarrow \boldsymbol{\xi}$
      $\#S \leftarrow \#S + 1$
      $\#F \leftarrow 0$
    else if $f(2\mathbf{x_{sw}} - \boldsymbol{\xi}) < f(\mathbf{x_{sw}})$ **then**
      $\mathbf{b} \leftarrow \mathbf{b} - 0.4\,(\boldsymbol{\xi} - \mathbf{x_{sw}})$
      $\mathbf{x_{sw}} \leftarrow 2\mathbf{x_{sw}} - \xi$
      $\#S \leftarrow \#S + 1$
      $\#F \leftarrow 0$
    else
      $\#S \leftarrow 0$
      $\#F \leftarrow \#F + 1$
    end if
    if $\#S > S_{ex}$ **then** ▷ Threshold S_{ex}: 5 in (Solis & Wets 1981)
      $\rho \leftarrow ex \cdot \rho$ ▷ "expand" factor ex: 2 in (Solis & Wets 1981)
    else if $\#F > F_{ct}$ **then** ▷ Threshold F_{ct}: 3 in (Solis & Wets 1981)
      $\rho \leftarrow ct \cdot \rho$ ▷ "contract" factor ct: 0.5 in (Solis & Wets 1981)
    end if
  end while
  Output $\mathbf{x_{sw}}$
end procedure

Table 2.1 highlights the main properties of the proposed methods in terms of their search logic and memory requirement. Each one of them can be used in a strategic way according to the problem: stochastic methods,

Table 2.1. Local Searchers main properties (n refers to the dimensionality of the problem).

LS algorithm	Search Logic	Derivatives	Memory Footprint	Processed Points	Convergence
NEWTON	*Deterministic (gradient descent)*	1^{st} *and* 2^{nd} *order*	$\mathcal{O}\left(n^2\right)$ *(Hessian matrix)*	*Single-solution*	q-quadratically[a]
HOOK-JEEVES	*Deterministic*	*Derivative free*	$\mathcal{O}\left(n\right)$	*Single-solution*	No proof
S	*Deterministic (along the axis)*	*Derivative free*	$\mathcal{O}\left(n\right)$	*single-solution*	No proof
NELDER-MEAD	*Deterministic*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(simplex vertices)*	*Multiple-solution*	Convergence[b]
ROSENBROCK	*Deterministic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(rotation matrix)*	*Single-solution*	Convergence[c]
POWELL	*Deterministic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n^2\right)$ *(directions matrix)*	*Single-solution*	$n\left(n+1\right)$ steps[d]
SOLIS-WETS	*Stochastic (diagonal move)*	*Derivative free*	$\mathcal{O}\left(n\right)$	*Single-solutions*	Convergence[e]
SPSA	*Stochastic (gradient descent)*	1^{st} *order*	$\mathcal{O}\left(n\right)$	*Single-solutions*	Convergence[f]

[a] Only for uni-modal and locally twice Lipschitz continuously differentiable functions.
[b] For convex functions in 1 and 2 dimensions.
[c] Under hypothesis on the fitness, such as differentiability, and on the line search method (Rinaldi 2012)
[d] Using the classic method in (Powell 1964) on quadratic forms.
[e] If f quasi-convex and inf-compact, converges in a neighbourhood of the local minimum. (Solis & Wets 1981).
[f] Under conditions on f and the distribution of probability used as in (Spall 1992)

for instance, are more robust in the presence of noise, on the other hand deterministic gradient based methods converge much quicker and with a higher accuracy in uni-modal problems. Deterministic evaluation of the gradient (and Hessian matrix) could lead to numerical instability beyond being computationally expensive. Stochastic approximation usually overcomes this problem but once again, the memory occupation issue and the temporal overhead introduced during the Hessian matrix processing still persist. For these reasons LS methods must be carefully chosen also taking into account external additional considerations. An example is the case of Engineering, where one could have a simple quadratic mono-modal problem but not be able to use algorithms such as Newton, Rosenbrock or Powell's method, because the project specifications require the optimiser to be plugged into an embedded system, or the optimisation process to be undertaken in real-time. A trade-off must always be taken in selecting a component during the algorithmic design phase. With reference to the last column of Table 2.1, must be said that the presence of theoretical proof of convergence could also be useful information in order to pick the right algorithm, e.g. if a problem satisfies all the hypothesis for the Newton method, then that one could be the best choice. Anyway, in the majority of cases, such mathematical conditions don't hold in the real world, or not much is known about the problem under study. So, the presence of a proof of convergence does not necessary imply that the algorithm is better or preferable with respect to the others in any case. Other issues in combining more LS algorithms or LS+Global Search (GS) together will be discussed in Chapter 4.

2.3 Global Search

The main goal of global optimisation is to find the extreme of a function regardless of its number of local optima. Unlike LS, it is in this case essential to explore the search space as much as possible rather than exploit an initial guess toward the direction of maximum decrease. Therefore, a good global optimiser should not only rely on the gradient information, but be able to sift the search space "jumping" out from local critical points. The simplest logic one can think of is based on a *brute force* approach, known as *enumeration*. A grid of K^n points is built within $\mathcal{D}$, e.g. each dimension is digitalised in n equally spaced values, and their fitness is evaluated. The bigger the number of points, the higher the reliability (as well the computational cost). A stochastic counterpart has also been proposed in (Gross & Harris 1985) with the name *Random Walk*. It simply consists of sampling randomly generated numbers within $\mathcal{D}$ and recording each improvement that occurs. These two examples, representing the first attempts of global optimisation, feature the following issues. The former needs a seriously high amount of functional calls in order to reach a reasonable accuracy, resulting in a computationally expensive and, at the same time, inefficient method. Due to the theoretically infinite number of points forming a continuous problem, this "greedy" approach is indeed weak and useless in many cases. On the other hand, the latter performs a completely randomised search which is good in handling noisy and highly multi-modal landscapes, but requires too much time in order to converge. Modern real world applications have become mightily complex and connected, e.g. networks sensors, telecommunications and neural systems, thus requiring real-time processing, robustness and reliability. The aforementioned techniques can no longer be used. In this light, the need of "putting intelligence" into algorithms arises. Modern global optimisers, as further discussed in section 2.4, are operators drawing their inspiration from intelligent and self-organising systems present in nature. The same way primates have evolved, DNA fragments persist in new generations, storms of animals organise themselves without a central coordination, or an atomic cluster reshapes itself in a lower energetic configuration when cooling down, so CIO methods evolve and refine a set of promising solutions. By mimicking these processes, CIO provides robust and versatile tools, displaying a good balance between exploration and exploitation of the search space. These techniques have been mainly developed during the last few decades, but are still the object of research.

2.4 Meta-Heuristics

The need to optimise a black-box system led computer scientists to develop versatile algorithms which do not require any assumption on the fitness function f. These general-purpose techniques are usually addressed with the term *meta-heuristics*, a Greek expression which literally means "beyond the search". Indeed, the search takes place on a higher level, where a master strategy inspired by natural phenomena, rules sub-operators in order to solve a given problem. The strength of meta-heuristics is their vast applicability. Regardless of the nature of the fitness function, which cannot even have an analytical expression (e.g. row data coming from a sensor), they work on an abstract level requiring only fitness evaluations. According to this definition,

methods like Rosenbrock, S, Nelder-Mead, Solis-Wets and Hooke-Jeeves can be considered meta-heuristics, since they don't make any hypothesis on f, thus not needing to know the problem under study. It must be said that nowadays this word is assuming a stronger meaning, referring mainly to a general framework for global optimisation, dealing with non-linear and multi-modal problems.

There exists a vast literature about meta-heuristics for global optimisation, spanning from Bee or Ant Colony Optimisation to Bacterial Foraging, Quantum Algorithms and many others metaphors and variations. All in all, most of them are diversifications of the same core concepts, and can be grouped under 4 big families: Evolutionary Algorithm (EA), Swarm Intelligence (SI) in Optimisation, DE and MA/MC. Among the huge variety of algorithms belonging to EA and SI, those which historically made a turning point will be briefly described in Chapter 3. A literature review presenting some recent advances in these fields will be given, in order to present in detail some modern versions which have been implemented and employed as comparison algorithms in this piece of research. MC will be extensively treated. Chapter 4 provides an introduction to the topic, explains the rationale behind this approach and also presents recent advances in the field. Chapter 6 and 7 include further investigations in MC, describing the majority of the findings that have been achieved during my doctoral studies.

It must be said that Chapter 3 will mainly focus on a class of meta-heuristic methods called population based algorithms, i.e. algorithms handling a set of multiple solutions. This class has been intensively studied since having "population of solutions" has been proven to carry a number of beneficial aspects (Prügel-Bennett 2010). For instance, the initial random sampling for generating the population performs an exploratory random walk within the search space. The population also acts as a filter which averages the landscape, so being able to handle small basins of attraction and small spikes present in the fitness. Evolving more solutions also improves the stability of the algorithm, i.e. standard deviation of the final values is reasonably small. For these reasons, population based approaches are usually preferred to single-solution methods. On the other hand, single-solution algorithms have a simpler structure with a lower memory footprint and, usually, a lower computational overhead. This makes them suitable for real-time applications and easy to embed in a hardware platform with memory limitations. In addition, it has recently been shown how (with some algorithmic precautions) they can display a similar (if not better) performance of consolidated population-based algorithms. In this regard, see (Caraffini, Neri, Iacca & Mol 2013), (Caraffini, Neri, Passow & Iacca 2013*a*), (Iacca, Neri, Mininno, Ong & Lim 2012) or in this work, Section 4.2, Section 6.2 and Chapter 7.1. Among popular single-solution meta-heuristic for global optimisation, it is worth mentioning SA (Kirkpatrick, Gelatt & Vecchi 1983) and its improved variant (Xinchao 2011). The latter has been used for comparison in this thesis, therefore, a brief description is given in the following subsection.

2.4.1 Simulated Annealing

SA is a stochastic meta-heuristic for global optimisation inspired from metallurgy, where metal is heated and subsequently cooled down in order to increase its crystal and reduce its defects. In effect, at high temperatures,

atoms are not confined within the crystal structure but free to move chaotically (exploration), while they settle down as soon as the temperature decreases (exploitation). By controlling the cooling, atoms are placed in a minimum energy configuration within the new crystal. This position cannot be achieved by dropping down the temperature immediately. Intermediate steps, with poor configurations occur. In the same way, SA stochastically move a solution within $\mathcal{D}$, initially covering big steps, and then converging by reducing the exploratory move. A good solution could be discarded during the convergence process in favour of worse configurations that can randomly occur. This process can be seen as a local search enhanced with a restart system for avoiding local minima. The risk of getting stuck on a local minimum is reduced by accepting poor configurations, with a probability which decreases exponentially $e^{-\frac{f(\mathbf{x_k})-f(\mathbf{x_{k-1}})}{T_k}}$. Many versions exist in the literature, Algorithm 12 presents the general framework. The typical perturbations strategy for SA makes use of a random number generator (usually Gaussian or Cauchy) to sample a new point $\mathbf{x_h}$, while the temperature is commonly updated with the following:

$$T_k = \frac{T_0}{ln\,(1+k)} \tag{2.7}$$

where T_0 is the initial (or maximum) temperature. This algorithm were first proposed for combinatorial optimisation in (Kirkpatrick et al. 1983), but has then also been largely used in continuous optimisation. Among the implementations for continuous problems, a recent version called Non uniform Simulated Annealing (nuSA) has shown to be promising in both toy-problems (Xinchao 2011) and robotic applications (Iacca, Caraffini & Neri 2013). This variant improves upon SA thanks to a non-uniform perturbation

$$\mathbf{x_k}[i] = \begin{cases} \mathbf{x_{k-1}}[\mathbf{i}] + \Delta\left(k, \mathbf{x^U}[i] - \mathbf{x_k}[i]\right) & if\ \mathcal{U}\,(0,1) > 0.5 \\ \mathbf{x_{k-1}}[\mathbf{i}] - \Delta\left(k, \mathbf{x_k}[i] - \mathbf{x^L}[i]\right) & otherwise \end{cases} \quad i = 1, 2, \ldots, n \tag{2.8}$$

acting as a classic Cauchy perturbation at early stages, i.e. global moves are taken within $\mathcal{D}$, and subsequently as a Gaussian perturbation, i.e. locally distributed around its mean value, when k is close to maximum amount of fitness functional calls K. This behaviour is guaranteed by $\Delta\,(k, y) = y \cdot \left(1 - \mathcal{U}\,(0,1)^{\left(1-\frac{k}{\mathrm{K}}\right)^b}\right)$, which returns a value approaching to zero as k increases. The b parameter is problem dependent and takes integers values, e.g. $b = 5$ in (Xinchao 2011). Regarding the cooling procedure, the following geometric progression is used:

$$T_k = \alpha \cdot T_{k-1} \tag{2.9}$$

with $\alpha \in [0.9, 0.99]$. It must be noted that in this case, the initial temperature T_0 is not allocated in advance, but estimated on the problem under investigation. A set of (e.g. 10) randomly sampled points must be generated and the best one $\mathbf{x_{best}}$, as well as the worst one ($\mathbf{x_{worst}}$), are collected to evaluate $T_0 = -\frac{f(\mathbf{x_{worst}})-f(\mathbf{x_{best}})}{ln(\xi)}$, where ξ is the initial acceptance rate for the worse solution. A value of $\xi = 0.9$ assures a good ability to explore the landscape at the beginning of the optimisation process.

Algorithm 12 Simulated Annealing

$k \leftarrow 0$
$\mathbf{x_k} \leftarrow randomSampling\,(1, N, \mathcal{D})$
$\mathbf{x_{sa}} \leftarrow \mathbf{x_{sa}}$
while *budget condition* **do**
  $k \leftarrow k + 1$
  $\mathbf{x_k} \leftarrow createNeighborSolution\,(k)$ ▷ e.g. Equation 2.8 in nuSA
  $T_k \leftarrow cooling\,(k)$ ▷ e.g. Equation 2.7 and 2.9 for SA ans nuSA respectively
  if $f\,(\mathbf{x_k}) \leq f\,(\mathbf{x_{k-1}})\ ||\ e^{-\frac{f(\mathbf{x_k}) - f(\mathbf{x_{k-1}})}{T_k}} < \mathcal{U}\,(0, 1)$ **then**
    if $f\,(\mathbf{x_k}) \leq f\,(\mathbf{x_{sa}})$ **then**
      $\mathbf{x_{sa}} \leftarrow \mathbf{x_k}$
    end if
  else
    $\mathbf{x_k} \leftarrow \mathbf{x_{k-1}}$
  end if
end while
Output $\mathbf{x_{sa}}$

Chapter 3

Modern meta-heuristics: literature review and advances of the most popular families

This chapter is a compendium of the most important "milestones" in modern optimisation. A detailed description of existing optimisation methods would fall outside the scope of this thesis. Nonetheless, a brief description of the outlines of the main techniques is needed in this thesis as it introduces methods that are modified, combined, and reinterpreted in the following chapters. Meta-heuristics presented in the 1980s and 1990s are classified into two main categories, EA and SI algorithms, respectively. In addition, DE is an optimisation framework displaying intermediate features between EA and SI algorithms. For the indicated decades, these algorithms were originally designed aiming to find a "universal algorithm", capable of global problem solving and outperforming all the others. The common thought that meta-heuristics could potentially be used to exactly solve every problem and thus the search of the best optimiser radically changed in 1997 when Wolpert and Macready theoretically disproved (under some hypotheses) this belief. This theoretical result has as a consequence the fact that algorithmic versatility offsets by a loss of performance. In other words, the wider the range of applicability, the lower the the performance in tackling a specific problem. For example, EA are not globally better than SI, but only better over a class of problems and vice-versa. To some extent, a basic Random Walk is not worse than EA over some specific problems, and according to (Wolpert & Macready 1997) the average performance of all meta-heuristics over all the possible problems is the same (see Appendix B for further details). This finding did not stop research in optimisation, but stressed the point that general purpose techniques, e.g. a random walk, must only be used when no knowledge of the problem is available (Black-box). On the other hand, an algorithm specifically tailored to the problem would provide a better solution, as graphically shown in Figure 3.1. It must be finally said that optimisation algorithms must be designed while taking into account practical aspects of real-world problems. For example, in some situations, a trade-off between algorithm's flexibility and specific performance must be found.

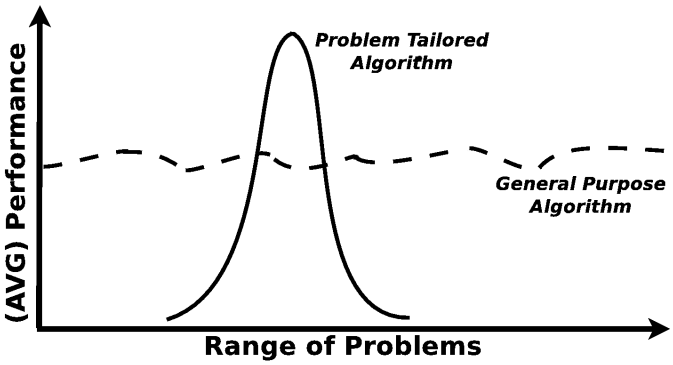


Figure. 3.1. Algorithms performance.

3.1 Evolutionary Algorithms

The idea of exploiting Evolution as a way of optimisation dates back to 1948 with Alan Turing. Even though he first proposed an "evolutionary approach", extensive experiments could be run only later on in the '60s (Bremermann 1962). In those years, computer scientists started designing algorithms inspired by Genetics and Darwinian Evolution, which have afterwards been grouped together under the umbrella name EA, a subset of a big branch of computer science commonly named Evolutionary Computing (EC). Regardless of the metaphor adopted, every EA is based on the same basic concept, and all of them share the same general structure as shown in Algorithm 13. First of all, EAs evolve a population of individuals, that is a set of solutions in $\mathcal{D}$, whose size has to be tuned according to the problem (noisy, mono-modal, multi-modal, etc.). During one iteration of the algorithm (generation) one or more new individuals are generated. The external force which

Algorithm 13 Evolutionary Algorithm

```
Initialisation                                   ▷ Randomly sample initial population
while condition on budget or fitness do
    Selection
    Recombination
    Mutation
    Replacement
end while
Output best individual
```

guides the evolution by awarding promising (fitter, from which comes the name *fitness function*) solutions is the *Selection* operator. Usually two "parents" are selected according to their fitness, the lower the better, but a certain probability of selecting bad individuals is always present in order to avoid premature convergence. A new point commonly called "offspring", expecting to have a fitness whose quality is at least as good as the one of its parents, is generated by mixing the previously selected points (*Recombination*). This exploration is somehow limited within the surrounding of the parents, therefore, by means of *Mutation* a further move attempts to keep population's diversity high, looking for unexplored sectors. Finally, with the *Replacement* operator it is decided whether or not new individuals can get a place within the population. The main idea is that better individuals go on, while bad ones are discarded (survival of the fittest), so forcing the population to adapt to the environment as happens in nature. This simple trial-and-error procedure does not make any assumption

on the fitness landscape and it is thus generally applicable to any kind of problem. Even in the presence of discontinuities, the population would be able to gradually reshape and conform to the fitness landscape.

3.1.1 Genetic Algorithm

The first instance of implementation of this popular algorithm, given in (Holland 1975), was meant to be applied in combinatorial problems and search logic, but has been then widely employed in mixed and continuous optimisation, producing a vast literature which could be treated apart as a separate topic, rather than as a dialect of EAs. The Genetic Algorithm (GA) without any doubt deserves to be mentioned, despite currently having a marginal role. Unlike other frameworks, when used for continuous optimisation, GA can encode individuals, often referred to as "chromosomes", by means of both real-valued numbers and binary strings (usually processed with Gray Code). According to the population representation adopted, recombination and mutation must be opportunely chosen. Recombination, takes here the name of "*Cross-Over*", and consists of randomly mixing two parents by swapping, copying or scrambling a portion of their components, commonly called "genes". In the real-value case, also a linear combination of the components of the parents can be performed (Herrera, Lozano & Sánchez 2003). Consequently, the offspring undergoes mutation, and with a certain probability one or more dimensions are flipped (or randomly perturbed with a Gaussian or Cauchy number within the bounds when real coding is used). This is the most exploratory moment, that allows us to reach sectors of $\mathcal{D}$ that would be contrarily uncovered. The last, but not least important stage, namely "replacement" or "survivor selection", eventually performs the logic for creating a new population by keeping/rejecting some new and old individuals. This step can be implemented in many different ways, since during a single iteration of a GA, cross-over and mutation can be used one or multiple times in order to generate offspring from a parental pool of selected chromosomes. In detail, by following a well known nomenclature, let μ be the population size and λ the number of newly "born" solutions. Then, μ points must be taken out from a set of μ old plus λ new solutions. In the simple GA, also called generational GA, a mating pool of size μ is considered (the full population) and $\lambda = \mu$ sons are reproduced in a single iteration. Only when a new generation is ready, from which comes the name "generational", does the old population face extinction, being brutally replaced by the new one. This replacement strategy is technically named (μ, μ) survivor selection. On the other hand, λ strictly less than μ, we talk about steady-state GA. Usually, a mating pool of only two individual is used and, according to the employed cross-over, $\lambda = 1$ or $\lambda = 2$ dual individuals are generated. These offspring can simply replace the oldest solutions in the current population (aged-based replacement), or replace the worst between the parents (regardless of whether their fitness is worse or better than one of the removed solutions), or the best μ solutions out of $\mu + 1/2$ are taken as the new generation ($(\mu + 1/2)$ survivor selection). It must be noted that the last method guarantees the prevention of the fittest element's presence onto the new generations. All the algorithms preserving the best ever found solution in the population are said to be *elitist*. Elitism is an important feature that could be required in some cases, e.g. in hybrid algorithms it assures that promising solutions coming from an LS, or other algorithms, are then not brutally discarded by a generational approach or an aged-based replacement.

Many different schemes exist for parent selection. A peculiarity of GA is that parents are selected in a stochastic way, so that the same individual can be chosen multiple times for breeding. The main idea is that fitter solutions have a higher probability to be selected in order to guide the evolution, since they are potentially able to generate a promising offspring. At the same time, a small probability of accepting mediocre individuals has to be present, otherwise population diversity would wane and the algorithm could get stuck on a local minimum. This goal was firstly achieved via Fitness Proportional Selection (FPS). As suggested by the name, a direct use of the fitness value is made to process the probability for each individual to be selected for mating, and this probability has to be proportional to its fitness value ($Prob\left(\mathbf{x_{selected}} = \mathbf{x_i}\right) = f\left(\mathbf{x_i}\right) / \sum_{j=1}^{M} f\left(\mathbf{x_j}\right)$). Common routines for FPS, are for instance the Roulette Wheel Selection (RWS) and the Stochastic Universal Sampling (SUS), see (Eiben & Smith 2007) for implementation. Since the first method does not give a good sample of the fitness distribution probability, the second way is usually preferred when multiple points have to be consecutively drawn from the distribution, while the first is preferred whenever only one sample is required. Algorithm 14 reports a routine for performing a single sampling with RWS. Both the methods mimic the functioning of an unconventional roulette wheel, having different sizes for each sector, i.e. sectors represent individuals of the population. While RWS needs a spin for each selection, SUS works on a single spin of the roulette wheel, then the desired number of individuals to be selected, let it be n_{ind}, is sampled by proceeding clockwise, from the start individual, with an angular step of $\frac{2\cdot\pi}{n_{ind}}$ radians. This procedure guarantees that individuals selections are in line with the expected frequencies. In a nutshell, if a solution occupies 2.5% of the wheel, after 100 samplings it would be selected no less than 2 times and no more than 3. These methods rely on a direct access on the fitness function value of each point in the population. Moreover, it has been noticed that this approach becomes inconsistent for steady landscapes, where individuals tend to have similar fitness values (It would result in a uniform random selection). This issue can be tackled by also taking into account an extra information on top of the absolute fitness value, i.e. population average fitness and standard deviation, as in the Sigma Scaling method proposed by Goldberg (Goldberg 1989). An alternative way to avoid this problem, is given by Rank-based selection: the actual fitness value is assessed to assign a score to each individual, then the probability is generated according to the rank so obtaining a diverse set of values. In case of large population size or where no global knowledge of the population is given, the $k-$Fitness Tournament Selection scheme can be also used, where k elements are randomly picked from the population, and the one with the best fitness among them is then selected. Finally, it is worth mentioning the Negative Assortative Mating selection procedure, originally introduced in (Fernandes & Rosa 2001), which has been found to be suitable in hybrid algorithms, see (Molina, Lozano, García-Martínez & Herrera 2010*a*) described in Chapter 4, in order to maintain a high the fitness diversity in the population of solutions. In order to achieve this goal, the first parent is simply sampled via RWS, while the second is chosen as the most distant individual from the first parent belonging to a set of k points (that can be sampled via RWS as well). This selection scheme was first introduced for binary encoding, employing the Hamming distance in order to measure the diversity between individuals. A real-valued version has been also proposed in (Molina, Lozano, García-Martínez & Herrera 2010*a*), where the Euclidean distance has replaced the Hamming distance and all the individuals involved are randomly drawn from a uniform distribution, so avoiding ranking procedures and RWS. It must be said that Positive Assortative Mating also exists, i.e. the closest individual is selected rather than the farthest, and can be used to pursue a quicker convergence. As for cross-over and mutation, a plethora of variants

exist in the literature. In (Molina, Lozano, García-Martínez & Herrera 2010*a*), the BLend X-over (BLX) crossover operator (Eshelman & Schaffer 1992), and the Breeder Genetic Algorithm (BGA) mutation (Mühlenbein & Schlierkamp-Voosen 1993) have been suggested as a good combination to enhance the exploration of the fitness landscape. Given two individuals $\mathbf{x_1}$ and $\mathbf{x_2}$ and a parameter $\alpha \geq 0$, 0.5 in this case, BLX-α generates an offspring $\mathbf{x_{offspring}}$ by means of the following:

$$\mathbf{x_{offspring}}\,[i] = \mathcal{U}\left(min_i - \delta_i \cdot \alpha, max_i - \delta_i \cdot \alpha\right) \;\; i = 0, 1, ..., N \tag{3.1}$$

where min_i and max_i are the maximum and the minimum value between the $i-th$ component of $\mathbf{x_1}$ and $\mathbf{x_2}$ respectively, and δ_i is the interval $max_i - min_i$. A value of 0.5 for α is sufficient to spread the distribution of the chromosome during the evolution, while BLX-0 is needed where the aim is to shrink the population distribution. BGA mutation then takes place on the newly generated offspring by altering only some of its genes according to:

$$\mathbf{x^{mut}_{offspring}}[i] = \mathbf{x_{offspring}}[i] + \mathcal{B}^{\pm 1}\,(0.5) \cdot range_i \cdot \sum_{k=0}^{15} \alpha_k \cdot 2^{-k} \;\; i = 0, 1, ..., N \tag{3.2}$$

with $range_i = 0.1 \cdot \left(\mathbf{x^U}[i] - \mathbf{x^L}[i]\right)$, and α_k randomly sampled in $\{0, 1\}$ with a probability $p\,(\alpha_k = 1) = \frac{1}{16}$. Such probability implies the mutation is expected to occur 1 time on each gene, since the summation in Formula 3.2 iterates 16 times ($k = 0, 1, ..., 15$), so generating a mutated gene within the interval $\left[\mathbf{x_{offspring}}[i] - range_i, \mathbf{x_{offspring}}[i] - range_i \cdot 2^{-15}\right] \cup \left[\mathbf{x_{offspring}}[i] + range_i \cdot 2^{-15}, \mathbf{x_{offspring}}[i] + range_i\right]$. This generated solution, is then immediately inserted in the population in the fashion of steady-state elitist GAs, since this combination of operators was thought to be used together with an LS, see (Molina, Lozano, García-Martínez & Herrera 2010*a*), whose beneficial effects could be spoilt by a non-elitist approach.

Algorithm 14 Roulette Wheel Selection Algorithm

```
procedure RWS(Values)                          ▷ Values: in GAs contains population's fitness values
    treshold ← 0
    s ← 0
    I ← size of Values
    for i = 1 : I do
        s ← s + Values[i]
    end for
    r ← U(0, s)
    for i = 0 : I do
        treshold ← treshold + Values[i]
        if r ≤ treshold then
            Break
        end if
    end for
    Output i
end procedure
```

3.1.2 Evolution Strategy

In Evolution Strategy (ES) the concept of self-adaptive parameters, i.e. the parameters automatically tune themselves during the optimisation process, was introduced for the first time, (Schwefel 1965)[1] and (Rechenberg 1971). ESs are completely randomised algorithms, where individuals are sampled "run-time" from a distribution which identifies the population, and so it makes sense to talk about parent-population and offspring-population rather than parents and offspring. The distribution of probability, usually a multivariate Gaussian function, is constantly updated using the best individuals and the self-adapting parameters which have lead to them. For this reason, a great amount of information must be stored in memory. In the most general situation, an individual is a compound of three vectors: $\langle \mathbf{x}, \boldsymbol{\sigma}, \boldsymbol{\alpha} \rangle$. The first one encodes the coordinates of a point in $\mathcal{D}$, the remaining are strategy parameters for the mutation operator. The idea is that better individuals also contain a better parameters set that, if used in order to adapt the parental distribution to the problem, can potentially lead to the generation of promising new solutions. Both object variable and strategy parameters are mutated, evolution is then mainly driven by the mutation operator.

Recombination plays only a minor rule in ES but it is still present under two principal variants: *discrete* and *intermediary*. In the first scheme, two points are drawn and for each dimension, the offspring has a 50% probability to inherit a gene from the first or second parent. In the latter variant, each gene of the offspring is the average of the corresponding genes coming from the parents. For the sake of completeness, it must be said that beside the described pairwise (or local) scheme, a multi-parent procedure (or global) also exists, where the whole population or just a subset of its most promising solutions are considered, and for instance, averaged with random weights in order to produce a new point. Both the recombination strategies can be applied at the same time on a single individual, e.g. local-discrete recombination is preferred for the object variables part, while global-intermediary is preferred for strategy parameters. The best choice among these methods, as always, depends on the specific problem. It must be noted that there is no particular selection rule, points are simply randomly drawn when needed.

The mutated version $\mathbf{x}'$ of a generic child $\mathbf{x}$ is then obtained from a Gaussian distribution centred in $\mathbf{x}$ with covariance matrix $\mathbf{C}$. Simply $\mathbf{x}' \sim \mathcal{N}(\mathbf{x}, \mathbf{C})$, where $\mathbf{C}$ is constructed after $\boldsymbol{\sigma} = \sigma_1, \sigma_2, ..., \sigma_{n_\sigma}$ and $\boldsymbol{\alpha} = \alpha_1, \alpha_2, ..., \alpha_{n_\alpha}$ being updated. For the sake of clarity, $\boldsymbol{\sigma}$ represents the standard deviation vector (also called step sizes vector), i.e. used to work out the variance on the diagonal of $\mathbf{C}$, while $\boldsymbol{\alpha}$ contains the angles of the rotation matrix for the Normal distribution, which therefore can be used to calculate the cross-correlation between two variables in $\mathbf{C}$. By updating these parameters the normal distribution can rotate, vary in size and shape so adapting to the landscape. This peculiarity makes ES suitable for ill-conditioned problems. It must be said that this is the most general scheme. One can also use uncorrelated mutations, for example in *uncorrelated mutation with n step sizes* $\boldsymbol{\alpha}$ vanishes, preventing rotation and allowing only changes in shape along the axes. In *uncorrelated mutation with* 1 *step size* instead, $\boldsymbol{\sigma}$ degenerates into a scalar value σ ($\Rightarrow \mathbf{C} = \sigma\mathbf{I}$ where $\mathbf{I}$ is the identity matrix), resulting in the circular shape (for a 2-dimensional problem) shown in Figure 3.2 that can only move and change in size.

A set of λ new points are sampled via mutation, and then evaluated for replacement. There are two widely

[1] A description in English is given in (Schwefel 1981).

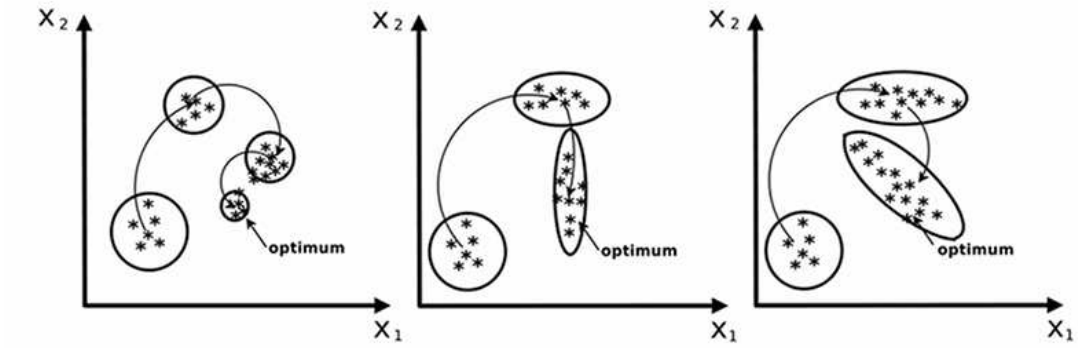


Figure. 3.2. Graphical representation of ES mutation operators. Uncorrelated mutation with 1 step size (left), uncorrelated mutation with n step sizes (middle) and correlated mutation (right). The oval represents the distribution from which points are sampled.

used strategies: (μ, λ) survivor selection or "comma" selection, and $(\mu + \lambda)$ survivor selection or "plus" selection. Unlike what happens in GA, the first strategy can also have $\lambda > \mu$, and the old population is replaced by choosing the μ out of λ fittest individuals. The second method, preserves old dated solutions (good in dynamic environments) by selecting the best μ individuals out of a complete set of all $\mu + \lambda$ solutions.

Among ESs, CMA-ES is probably the most important and efficient. A more detailed description of this framework and some of its variants are reported below.

3.1.2.1 Covariance Matrix Adaptation Evolutionary Strategy

The Covariance Matrix Adaptation ES (Hansen & Ostermeier 1996) is one of the most elegant and powerful meta-heuristic optimisers, based on a solid mathematical background and featuring several desirable properties, i.e. lack of problem dependent parameters and invariance to many transformations. CMA-ES is a completely randomised framework evolving a multivariate normal distribution whose mean and covariance matrix are adaptively updated from a subset of promising solutions, making it suitable for ill-conditioned landscapes. Due to the necessity of evaluating and storing in memory the covariance matrix at each iteration, this solver is difficult to apply in LSOP, for memory consumption and time complexity issues. Despite these flaws, this is one of the most efficient frameworks for general-purpose optimisation, especially when applied on mono-modal problems.

Referring to the standard CMA-ES with rank-μ-update and weighted (or global) recombination, as presented in (Nikolaus & Stefan 2004), the basic step, that is the sampling of an individual $\mathbf{x_k}$ $(k = 1, 2, ..., \lambda)$ in a generic generation $g + 1$, is given by:

$$\mathbf{x_k^{(g+1)}} \sim \mathcal{N}\left(\langle \mathbf{x} \rangle_w^g, (\sigma^g)^2 \, \mathbf{C}^g\right) \tag{3.3}$$

where $\mathcal{N}\left(m, \sigma^2\mathbf{C}\right)$ is a multivariate normal distribution of mean m, step-size σ, and estimated covariance matrix $\mathbf{C}$. The mean vector $\langle\mathbf{x}\rangle_w^g$ is a weighted sum of the μ candidate solutions ($\mu \leq \lambda$, with $\lambda = 4 + floor^2\left(3 \cdot n\right)$) displaying the best performance in term of fitness in the generation g. This vector corresponds to a recombination result $\langle\mathbf{x}\rangle_w^g = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^g$, where $\mathbf{x}_{i:\lambda}^g$ denotes the i^{th} best individuals at the generation g amongst the λ available and w_i are weight factors, see (Nikolaus & Stefan 2004) for details. Step-size σ and covariance matrix are progressively updated in each generation as follows:

$$\mathbf{C}^{g+1} = (1 - c_{cov})\mathbf{C}^g + c_{cov} \cdot \frac{1}{\mu_{cov}} \mathbf{p_c}^{g+1} \left(\mathbf{p_c}^{g+1}\right)^T + c_{cov} \cdot \left(1 - \frac{1}{\mu_{cov}}\right) \sum_{i=1}^{\mu} \left(\mathbf{x}_{i:\lambda}^{g+1} - \langle\mathbf{x}\rangle_w^g\right) \left(\mathbf{x}_{i:\lambda}^{g+1} - \langle\mathbf{x}\rangle_w^g\right)^T \tag{3.4}$$

where c_{cov} is a parameter determining the learning rate for the estimated covariance matrix $\mathbf{C}$, and $\mathbf{p_c}$ is a vector namely evolution path that determines the adaptation of the covariance matrix. The update formula is given by:

$$\mathbf{p_c}^{g+1} = (1 - c_c)\mathbf{p_c}^g + H_\sigma^{g+1} \sqrt{c_c \cdot (2 - c_c)} \cdot \frac{\sqrt{\mu_{eff}}}{\sigma^g} \left(\langle\mathbf{x}\rangle_w^{g+1} - \langle\mathbf{x}\rangle_w^g\right) \tag{3.5}$$

where $\mu_{eff} = \frac{1}{\sum_1^\mu w_i^2}$, c_c is a parameter, H_σ^{g+1} is a function defined by cases that can take values 0 or 1. Also the step-size σ^{g+1} is iteratively updated. Details about CMA-ES different implementations can be found e.g. in (Hansen & Ostermeier 2001), (Hansen, Müller & Koumoutsakos 2003) and in the tutorial (Nikolaus 2005). At the end of each generation the μ individuals displaying the best performance are selected and used to compute $\langle\mathbf{x}\rangle_w^{g+1}$. After a certain amount of generations, the matrix $\mathbf{C}$ evolves and reliably approximates the (theoretical) covariance matrix.

Subsequently, enhanced versions of CMA-ES have been proposed to tackle specific problems. One of the main issue with CMA-ES is the deterioration of its performances when dealing with multi-modal functions. This flaw has been overcome in (Auger & Hansen 2005), proposing a multi-start system which resizes the population size after each restart (G-CMA-ES).

An interesting version for tackling separable problems has been also proposed in (Ros & Hansen 2008) with the name sep-CMA-ES. This variant simply makes use of a diagonal covariance matrix (uncorrelated mutation), thus performing stochastic moves along the axes, as in Figure 3.3, which are preferable for solving separable problems. It was noticed that this algorithmic scheme improves upon the classic CMA-ES in high dimensional problems (n larger than 100), even on non-separable objective functions. In other words, it was shown that while the interaction among variables is very important in low dimensions, it appears not so important when the dimensionality grows. This phenomenon will be examined and discussed in detail in Chapter 8.

3.1.2.2 $(1 + 1)$-Covariance Matrix Adaptation Evolution Strategy

A significant and computationally efficient single solution variant of CMA-ES, namely the $(1 + 1)$ Covariance Matrix Adaptation Evolution Strategy ((1+1)-CMA-ES) algorithm, was presented in (Igel, Suttorp & Hansen 2006). It combines a classic $(1 + 1)$-ES scheme, i.e. elitist single-solution algorithm, with an improved mechanism for building the covariance matrix which replaces the computationally onerous Covariance Matrix

[2] maps a real-valued number to the largest previous integer.

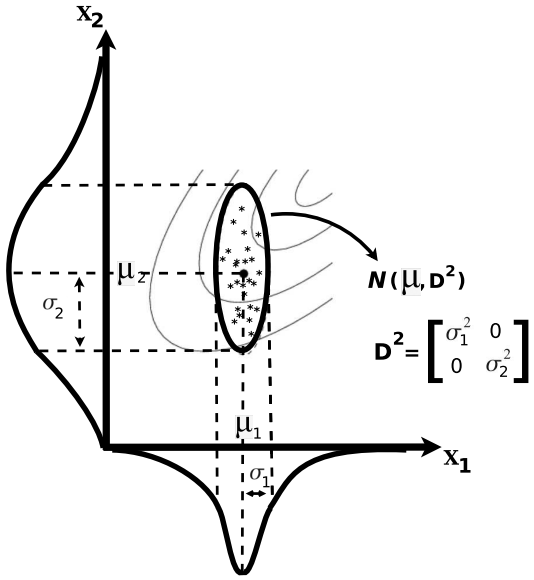


Figure. 3.3. Graphical representation of the sep-CMA-ES strategy for separable and large scale problems. The oval represents the distribution from which points are sampled.

decomposition ($\mathcal{O}\left(n^3\right)$ operations) with an incremental update of the Cholesky factors ($\mathcal{O}\left(n^2\right)$ operations). Even though (1+1)-CMA-ES does not employ a population of solutions, it needs to store the covariance matrix, thus requiring an amount of memory that grows quadratically with the dimensionality of the problem as it happens in CMA-ES. On the other hand, (1+1)-CMA-ES is numerically less demanding than CMA-ES, still being able to equal it on non-separable problems. For the sake of clarity, it must be said that in the original

Algorithm 15 $(1+1)$-Covariance Matrix Adaptation Evolution Strategy

procedure $(1+1)$-CMA-ES($\mathbf{x_0}$)
  $\mathbf{x_{parent}} \leftarrow \mathbf{x_0}$
  $\mathbf{I} \leftarrow eye\,(n)$
  $\mathbf{A} \leftarrow \mathbf{I}$
  $\mathbf{x_{parent}} \leftarrow \mathbf{x_0}$
  $\bar{p}_{succ} \leftarrow p_{succ}^{target}$ ▷ $\bar{p}_{succ}$: averaged success rate, $p_{succ}^{target} = \frac{2}{11}$ (Igel et al. 2006)
  $\mathbf{p_c} \leftarrow \emptyset$
  while *condition on budget* **do**
    $\mathbf{z} \leftarrow \mathcal{N}\left(\emptyset, \mathbf{I}\right)$
    $\mathbf{x_{offspring}} \leftarrow \mathbf{x_{parent}} + \sigma \mathbf{A} \times \mathbf{z}$ ▷ The initial value for σ is problem dependent, typical value is 1
    if $f\left(\mathbf{x_{offspring}}\right) \leq f\left(\mathbf{x_{parent}}\right)$ **then**
      $\mathbf{x_{parent}} \leftarrow \mathbf{x_{offspring}}$
      $\lambda \leftarrow 1$
    else
      $\lambda \leftarrow 0$
    end if
    $\bar{p}_{succ} \leftarrow (1 - c_p)\,\bar{p}_{succ} + c_p \lambda$ ▷ $0 < c_p \leq 1, c_p = \frac{1}{12}$ in (Igel et al. 2006)
    $\sigma \leftarrow \sigma e^{\frac{1}{d}\left(\bar{p}_{succ} - \frac{p_{succ}^{target}}{1 - p_{succ}^{target}}(1 - \bar{p}_{succ})\right)}$ ▷ $d = 1 + \frac{n}{2}$ (Igel et al. 2006)
    if $f\left(\mathbf{x_{offspring}}\right) \leq f\left(\mathbf{x_{parent}}\right)$ && $\bar{p}_{succ} < p_{thresh}$ **then** ▷ $p_{thresh} = 0.44$ (Igel et al. 2006)
      $\mathbf{A} \leftarrow \sqrt{1 - c_{cov}}\mathbf{A} + \frac{\sqrt{1 - c_{cov}}}{\|\mathbf{z}\|^2} \cdot \left(\sqrt{1 + \frac{c_{cov}\|\mathbf{z}\|^2}{1 - c_{cov}}} - 1\right)(\mathbf{A} \times \mathbf{z})\,\mathbf{z}^T$ ▷ $c_{cov} = \frac{2}{n^2+6}$ (Igel et al. 2006)
    end if
  end while
  Output $\mathbf{x_{parent}}$
end procedure

paper Hansen proposes two algorithms based on the $(1+1)$ recombination. While the first one processes the covariance matrix in the fashion of CMA-ES (slow processing), the second version never computes the covariance matrix explicitly, since operating with the Cholesky factors only (quick processing). Since the second version has the advantage of saving precious computational operations, it seemed fair to report the implementation (Algorithm 15) referring to the "quick" variant. In order to avoid confusion, it must be noted that this implementation is the one that will be addressed in the remainder of this thesis (Section 6.2), with the name (1+1)-CMA-ES.

3.1.3 Evolutionary Programming

Evolutionary Programming (EP) is an historical member of EC, see (Fogel, Owens & Walsh 1966), which encodes individuals within a real-valued vector containing either a description of the candidate solution $\mathbf{x}$, and a set of self-adapting parameters $\boldsymbol{\sigma}$ called *meta-EP*. These parameters adapt to the environment, and guide mutation in order to mimic how new features are revealed in living beings. Each individual consists of:

$$\langle \mathbf{x}, \boldsymbol{\sigma} \rangle = \langle x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n \rangle \tag{3.6}$$

At every iteration all the individuals in the current population, *Pop* of size μ, are systematically mutated (no recombination is needed!) by means of a Gaussian perturbation in order to create a temporary population $Pop^{'}$:

$$\begin{cases} \sigma_i^{'} = \sigma_i \left(1 + \alpha \cdot \mathcal{N}(0,1)\right) \\ x_i^{'} = x_i + \sigma_i^{'} \cdot \mathcal{N}(0,1) \end{cases} \quad i = 1, 2, \ldots, n \;\; \forall \langle \mathbf{x}, \boldsymbol{\sigma} \rangle \in Pop \tag{3.7}$$

where α is a control parameter usually set equal to 0.2. Replacement takes place via a variant of the $(\mu + \mu)$-survivor selection: each $\langle \mathbf{x}, \boldsymbol{\sigma} \rangle \in Pop \cup Pop^{'}$ is compared according to its fitness value against μ randomly sampled individuals. Only the μ tuples which have won more comparisons are stored in the new population. Apart from the lack of recombination, EP still follows the general scheme in Algorithm 13.

3.1.4 Genetic Programming

GP can be seen as an extension of GA, proposed by Koza (Koza 1992), where a model is evolved rather than a set of solutions. Individuals in this case are not vectors, but data structures encoded as a parse tree. Their internal nodes can encapsulate a mathematical formula, a symbolic expression or a fragment of code. External leaves represent variables and constant that can be used as input. By following the basic steps in algorithm 13, this population is evolved in order to obtain an individual carrying the mathematical model, or software procedure with maximum quality. The fitness function is indeed used to express how well the evolutionary generated model or the software performs. Parents are commonly selected according to their fitness value in order to proceed with recombination, where sub-trees are swapped between parents in order to generate two sons. The

original framework did not contemplate mutation, but it has then been proved in (Luke & Spector 1997) that some random changes in the offspring are of help in the evolutionary process. The population is updated by following the generational scheme so, at each generation, the old population is completely replaced with new individuals.

3.2 Swarm Intelligence Optimisation

SI is a very broad and interdisciplinary subject AI. The main idea behind SI comes from the observation of groups of animals, e.g. a flock of birds crossing the sky or school of fish turning together in the sea, displaying a remarkable self-organisation despite the fact there is no central intelligence or coordination. The principle that, by following simple rules, a collective intelligence can arise from apparently non intelligent units, has attracted the attention of researchers who encoded this behaviour in artificial systems. The expression "Swarm Intelligence" appeared for the first time in Robotics (Beni & Wang 1989), but telecommunication systems and optimisation algorithms quickly adopted this concept.

In optimisation, a set of solutions shape a swarm exploring $\mathcal{D}$. Each one of them is singularly moved by following simple behavioural rules, e.g. follow the swarm leader or adapt your velocity to those of yours neighbours and consequently update the current position (and/or others physical quantities). In terms of structure and implementation SI optimisation algorithms are not so different from EA. Even though the underlying metaphor envisages a swarm rather than a population, spacial coordinates instead of genes, position perturbation with respect to mating and mutation of individuals, the same concept of moving a set of points within the search space is encoded. Nonetheless, they have a more rigid scheme for selecting and updating points in the swarm, characterising this subclass of optimisers. With reference to Algorithm 16, it can be seen that solutions are cyclically perturbed one at time, and updated right after the perturbation if the new point outperforms the current solution being considered. This mechanism, commonly called "1-to-1 spawning", distinguishes SI and makes sure that every perturbation (usually involving the best point or neighbour points in the swarm) works on an updated set of solutions. Not employing any fitness-based selection of the point to be perturbed (all of them are sequentially considered), replacement has to come right after the move, since the optimisation is guided by making the worse elements in the swarm follow the better ones, rather than breeding promising individuals allowed in a mating pool. The described procedure follow the same scheme of steady state (μ+1)-EAs employing and age-based replacement strategy. Many optimisers have been designed

Algorithm 16 Swarm Intelligence Optimisation

```
Initialisation                                          ▷ Randomly sample initial swarm
while Condition on budget do
    for each x ∈ Swarm do
        Update perturbation parameters and apply perturbation: x ⟹ x'
        Update swarm
    end for
end while
Output Best Individual
```

after SI philosophy. Among them, Ant Colony Optimisation (Colorni, Dorigo, Maniezzo et al. 1991) which

mimics the food-seeking behaviour of ants, and Bacterial Foraging Optimisation (Passino 2002), which models the foraging and reproductive behaviour of the Escherichia coli bacteria, are worth mentioning and amongst the most successful and widely used PSO. Regardless of the metaphor adopted, most of these algorithms implements the same ideas. For instance, both Ant Colony and Bacterial Foraging Optimisation algorithms implements a similar working principle from Estimation of Distribution Algorithm (EDA), while PSO shares similarities with (μ+1)-EAs.

3.2.1 Particle Swarm Optimisation

PSO was originally designed by J. Kennedy and R. C. Eberhart for simulating social behaviour, and subsequently simplified and readjusted for dealing with optimisation (Kennedy & Eberhart 1995). A set of P solutions, the "particles", in the parameter space, the swarm, are perturbed taking into consideration the history of the swarm. A simple entity like a particle, can interact with its neighbour by sharing basic information about its past motion, and from this social interaction a global capability of problem solving arises from non intelligent units. In order to do this, each particle $\mathbf{x_p}$, $p = 1, 2, \ldots, P$, is encoded as a real-valued vector containing its position within $\mathcal{D}$. On top of that, two more vectors are associated to each point so storing the best and the worst position occupied so far from the particle ($\mathbf{x_p^b}$ and $\mathbf{x_p^w}$ respectively). Finally, a third array, called velocity[3] $\mathbf{v_p}$ is required for the perturbation scheme. A particle can thus be seen as a quartet (in the most general case, $\mathbf{x_p^w}$ is not usually needed) $\langle \mathbf{x_p}, \mathbf{x_p^b}, \mathbf{x_p^w}, \mathbf{v_p} \rangle$. The vector $\mathbf{x_p^b}$, with p so that its fitness value is the highest in the swarm (highest fitness in minimisation is the lowest objective function value), must be constantly detected and stored as global best $\mathbf{x_{gb}}$. The standard perturbation consists of updating $\mathbf{v_p}$:

$$\mathbf{v_p^{'}} = \phi_1 \mathbf{v_p} + \phi_2 \left(\mathbf{x_p^b} - \mathbf{x_p} \right) + \phi_3 \left(\mathbf{x_{gb}} - \mathbf{x_p} \right) \tag{3.8}$$

and evaluating:

$$\mathbf{x_p^{'}} = \mathbf{x_p} + \mathbf{v_p} \tag{3.9}$$

where ϕ_1, ϕ_2, and ϕ_3 are three weights which need to be chosen carefully on the problem, since performances heavily depend on their values (Yuhui & Eberhart 1998), and the superscript represents the new updated value to be used. Normally, these weights contain a random component, as in (Clerc & Kennedy 2002), but many other strategies can be adopted for their tuning. In (Zheng, Ma, Zhang & Qian 2003), for instance, a dynamic time-increasing inertia weight ϕ_1 is employed. It can be said that there are two moves embedded in Formula 3.8: a first one towards the best position by far explored by the current particle, and a second towards the global best solution. These two forces, together with the old velocity, are weighted and summed up in order to generate a new velocity vector. In turn, the new velocity change the current position of the particle as in Formula 3.9. It can be observed that in the case ϕ_2 equal to zero, then PSO behaves like an ES where the second parent chosen for recombination is the fittest, and the mutation operator is the addition of $\mathbf{v_p}$. The replacement procedure, "survivor selection", then compare the fitness value of the newly generated particle,

[3]It must not be confused with the physical meaning of this term, strictly speaking it is a displacement rather than a velocity

and in case of improvement replaces the local best particle accordingly. The global best point has to be checked as well. This simple, but extremely efficient logic has been applied on a vast set of problems and real-world applications of a different nature (Poli 2008). Many variants exist in the literature, and many new perturbation schemes have been proposed. Multiple perturbation strategies have also been grouped and combined within a single optimiser, see for example Frankenstein's PSO (Montes de Oca, Stutzle, Birattari & Dorigo 2009). Among recent advances in PSO, three significant examples are reported here. These examples are selected on the basis of considerations of their peculiar features and performances (numerical results can be found in Appendix E).

An efficient and relatively simple PSO algorithm, Comprehensive Learning Particle Swarm Optimiser (CLPSO), has been proposed in (Liang, Qin, Suganthan & Baskar 2006). In CLPSO velocity is updated with a particular formula considering all the best ever explored positions of each particle:

$$\mathbf{v_p} = \phi_1 \mathbf{v_p} + \phi_2 \mathbf{U} \times \left(\mathbf{x_p^{rb}} - \mathbf{x_p} \right) \tag{3.10}$$

where $\mathbf{U}$ is a $n \times n$ matrix of uniform distributed random numbers, $\mathbf{x_p^{rb}}$ is a vector built up by randomly sampling components from all the local bests vectors $\mathbf{x_p^b}$. The sampling procedure makes use of a threshold, P_c, which has to be generated for each dimension $i = 1, 2, \ldots, n$:

$$P_{c-i} = 0.05 + 0.45 \cdot \frac{e^{\frac{10(i-1)}{P-1}} - 1}{e^{10} - 1} \tag{3.11}$$

where P is swarm size (number of particles). A random number is then drawn and, if higher than P_c, the $i-th$ component of the corresponding local best particle is followed. Otherwise, two randomly selected local best particles are compared according to their fitness value. The fittest of the two donates the $i-th$ component to $\mathbf{x_p^{rb}}$. This scheme as shown to be versatile and efficient in particular up to 50 design variables.

A single solution variant has also been considered in this thesis. The Intelligent Single Particle Optimisation (ISPO) algorithm features the basic PSO perturbation strategy albeit employing a swarm degenerately narrowed to one particle. As described in (Zhen, Jiarui, Huilian & Qing-Hua 2010), the single particle is perturbed for each of the n design variables H times. Social interaction is not possible, so velocity cannot be evaluated according to the standard way. For $h = 1, 2, \ldots, H$ every $i-th$ variable is then evaluated as follows:

$$v_i^{h+1} = \frac{A}{(h+1)^P} \cdot \mathcal{U}(-0.5, 0.5) + B \cdot L^h \tag{3.12}$$

where A, P, and B are three problem dependent parameters called acceleration, acceleration power factor, and learning coefficient. The learning factor L is instead initially set to zero and then updated after each perturbation, i.e. $x[i]^{t+1} = x[i]^t + v[i]^{t+1}$. If the new particle improves upon or equals the old one then $L^{h+1} = v^{h+1}[i]$, otherwise (if $L \neq 0$) $L^{h+1} = \frac{L^h}{\mathbf{S_F}}$, where $\mathbf{S_F}$ is a shrinking factor. L is reinitialised to zero whether its absolute value is too small, i.e $|L| < E = 10^{-5}$, due to a number of unsuccessful perturbations . This mechanism ensures a randomised search along each dimension, whose exploratory radius is larger at the

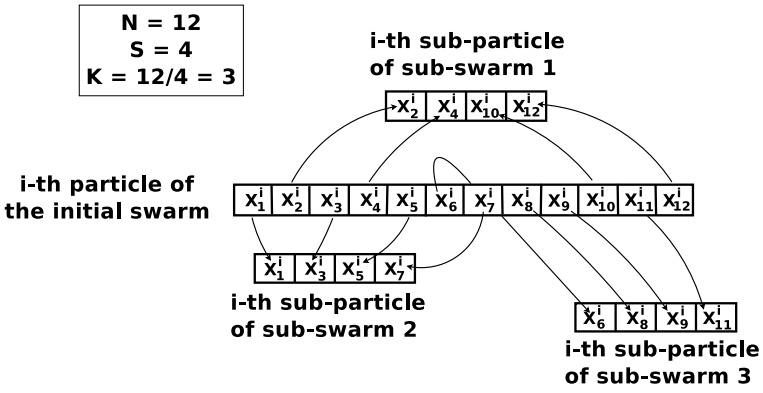


Figure. 3.4. Graphical representation of CCPSO2 strategy for large scale optimisation. N is the number of variables, S is the number of variables per sub-particle, and K is the number of sub-particles.

beginning of the optimisation process and progressively shrinks as h is increased. Despite the lack of a swarm causing a loss of performance in some class of problems, this algorithm is characterised by a minimalistic memory footprint, making it easy to plug in embedded systems, see (Iacca, Caraffini & Neri 2013). In addition, since performing optimisation on each coordinate axis at a time, it has been proven to perform as well as population based algorithms on separable objective functions, as well as on LSOP.

A recent implementation specifically tailored for LSOP is given in (Li & Yao 2012). The proposed algorithm, namely Cooperatively Coevolving Particle Swarms for Large Scale Optimisation (CCPSO2), faces high dimensional problems by decomposing them in lower-dimensional sub-problems, which can be optimised easier by a simplified and completely randomised PSO variant. As can be seen in Appendix E, numerical results are competitive with those of other state-of-the-art algorithms in both low and high dimensional functions, proving also that a simple perturbation can provide accurate solutions if the coordination logic is well structured and designed. According to the dimensionality of the problem under study, a set of admissible divisors must be pre-allotted. As graphically showed in Figure 3.4 a divisor S is randomly drawn from this set in order to split the original N-dimensional swarm in $K = \frac{N}{S}$ S-dimensional sub-swarms. Each sub-swarm is then processed one at time, and for each sub-swarm only one perturbation is applied to each sub-particle. If this process does not lead to any fitness minimisation, another divisor is drawn and different sub-swarms are initialised, on the contrary the successful configuration is kept for the next iteration. No velocity vector is needed, perturbation adopts a simplified scheme employing Gaussian and Cauchy random moves:

$$\mathbf{x}_i = \begin{cases} \mathbf{x^b}_i + \mathcal{C}(0,1)\,|\mathbf{x^b}_i - \mathbf{x^{gb}}_i| & if\ \mathcal{U}(0,1) < p \\ \mathbf{x^{gb}}_i + \mathcal{N}(0,1)\,|\mathbf{x^b}_i - \mathbf{x^{gb}}_i| & otherwise \end{cases} \qquad i = 1, 2, \ldots, S \tag{3.13}$$

where $\mathbf{x^b}$ and $\mathbf{x^{gb}}$ refer to the personal best and the global best of the current sub-swarm. Fitness functional calls are performed by reconstructing the design vector concatenating all the components coming from sub-swarms. It can be seen like a local perturbation on a sub-set of design variables while keeping the others constant. To some extent, this approach is the population-based counterpart of the modified Solis and Wets

method for LSOP in Algorithm 11, also employing dimensionality reduction and stochastic Gaussian moves.

3.3 Differential Evolution

DE was designed by Rainer Storn and Kenneth Price in 1995 in an attempt to implement a simple optimiser for solving a fitting problem. The main idea was to use the scaled version of the difference vector between two randomly selected individuals of a population, from which comes the name "Differential" Evolution, in order to create a new point within $\mathcal{D}$. Two basic combinations of difference vectors were originally defined in a technical report (Storn & Price 1995), then due to its versatility and efficiency this method has been formalised in (Rainer & Kenneth 1997) and further developed and applied to various problems during the following years, see (Storn 1996), (Storn 1999), (Storn 2005) and (Price et al. 2005), thus becoming a proper stand-alone optimisation framework. It is indeed difficult to categorise since sharing, as well as the innovative perturbation scheme, some features in common with EAs, such as the population of individuals and the cross-over operator in the fashion of GAs, but also employs the 1-to-1 spawning selection of SI optimisers. In more detail, the structure of any DE based algorithm follows the schematic in Algorithm 17. After sampling an initial population of M individuals, each one of them is sequentially perturbed from the first one to the last (generation cycle). Despite performing 1-to-1 spawning, unlike PSO in case the new individual displays a better fitness value it will get a place into the population only after a complete generation cycle, and not right after the perturbation. This generational approach has shown to be more efficient and also allows parallelisation (Rainer & Kenneth 1997), since each mutation is independent of the others. According to the very first version, assuming to process the $j\text{-}th$ target vector, three random individuals need to be randomly picked from the population: $\mathbf{x_{r_1}}$, $\mathbf{x_{r_2}}$ and $\mathbf{x_{r_3}}$, with $r_3 \neq r_2 \neq r_1 \neq j$ $(j = 1, 2, ..., M)$. The difference vector can so be evaluated and a mutant vector obtained from Equation 3.14, where the scale factor F must be chosen in $[0, 2]$. Mutation is basically a linear combination of individuals, the reason behind this logic is that this approach guarantees large moves in the early stages of the optimisation, becoming smaller and smaller, and so more precise around the optimum, as soon as the population converges, i.e. if individuals are converging in a neighbourhood of the optimum the magnitude of the difference vector decreases and the new generated point ($\mathbf{x_m}$ also called mutant vector) falls in the surroundings of the minimum. In order to assure a certain amount of diversity, the current target vector and the mutant vector undergo cross-over (originally binary cross-over as in Algorithm 18), so generating an offspring ($\mathbf{x_{off}}$) which compete with its parent ($\mathbf{x_j}$) for survival. If it improves upon its parent, then the new point will replace the target vector in the new population, when starting a new generation cycle. The presented algorithm is also known as DE/rand/1/bin, but many other combinations can be used employing some of the most, by this time, common mutations:

- rand/1:

$$\mathbf{x_m} = \mathbf{x_{r_1}} + F\left(\mathbf{x_{r_2}} - \mathbf{x_{r_3}}\right) \tag{3.14}$$

Algorithm 17 Differential Evolution

$g \leftarrow 1$ ▷ First generation
$\mathbf{Pop^g} = randomSampling\,(M, n, \mathcal{D})$
$\mathbf{x_{best}} \leftarrow$ *fittest individual* $\in \mathbf{Pop^g}$
while *Condition on budget* **do**
  for each $\mathbf{x_j} \in \mathbf{Pop^g}$ **do** ▷ $j = 0, 1, 2, ..., M$
    $\mathbf{x_m} \leftarrow$*Mutation* ▷ e.g. Equations 3.14-3.18 in (Storn & Price 1995), Equation 3.17 in (Rainer & Kenneth 1997)
    $\mathbf{x_{off}} \leftarrow$*CrossOver*$(\mathbf{x_j}, \mathbf{x_m})$ ▷ e.g. Agorithm 18 in (Storn & Price 1995)
    if $f\,(\mathbf{x_{off}}) \leq f\,(\mathbf{x_j})$ **then**
      $\mathbf{Pop^{g+1}}[j] \leftarrow \mathbf{x_{off}}$
    else
      $\mathbf{Pop^{g+1}}[j] \leftarrow \mathbf{x_j}$
    end if
  end for
  $g \leftarrow g + 1$ ▷ Replace the old with the new generation
  $\mathbf{x_{best}} \leftarrow$ *fittest individual* $\in \mathbf{Pop^g}$ ▷ update best individual
end while
Output *Best Individual* $\mathbf{x_{best}}$

- best/1:

$$\mathbf{x_m} = \mathbf{x_{best}} + F\,(\mathbf{x_{r_1}} - \mathbf{x_{r_2}}) \tag{3.15}$$

- rand/2:

$$\mathbf{x_m} = \mathbf{x_{r_1}} + F\,(\mathbf{x_{r_2}} - \mathbf{x_{r_3}}) + F\,(\mathbf{x_{r_4}} - \mathbf{x_{r_5}}) \tag{3.16}$$

- best/2:

$$\mathbf{x_m} = \mathbf{x_{best}} + F\,(\mathbf{x_{r_1}} - \mathbf{x_{r_2}}) + F\,(\mathbf{x}_{r_3} - \mathbf{x_{r_4}}) \tag{3.17}$$

- current-to-best/1:

$$\mathbf{x_m} = \mathbf{x_j} + F\,(\mathbf{x_{best}} - \mathbf{x_j}) + F\,(\mathbf{x_{r_1}} - \mathbf{x_{r_2}}) \tag{3.18}$$

- current-to-rand/1:

$$\mathbf{x_m} = \mathbf{x_j} + K\,(\mathbf{x_{r_1}} - \mathbf{x_j}) + F^{'}\,(\mathbf{x_{r_2}} - \mathbf{x_{r_3}}) \tag{3.19}$$

- rand-to-best/1:

$$\mathbf{x_m} = \mathbf{x_{r_1}} + F\,(\mathbf{x_{best}} - \mathbf{x_j}) + F\,(\mathbf{x_{r_2}} - \mathbf{x_{r_3}}) \tag{3.20}$$

- rand-to-best/2:

$$\mathbf{x_m} = \mathbf{x_{r_1}} + F\,(\mathbf{x_{best}} - \mathbf{x_j}) + F\,(\mathbf{x_{r_2}} - \mathbf{x_{r_3}}) + F\,(\mathbf{x_{r_4}} - \mathbf{x_{r_5}}) \tag{3.21}$$

together with different cross-over operators, e.g. binary or exponential (Algorithm 18 and 19 respectively). A particular scheme can be identified by means of the DE/x/y/z notation, where x refers to the vector being mutated (namely the one to which difference vectors are added), y is the number of difference vectors used and z indicates the cross-over. For example, x could be "rand" (Equations 3.14 and 3.16), "best" ((Equations 3.15) and 3.17) or even two vectors as in "current-to-best", "current-to-rand" and "rand-to-best" (Equations 3.18, 3.19, 3.20 and 3.21). The above listed mutations provide a set of different moves across $\mathcal{D}$ and can be chosen according to the problem. The current-to-best/1 strategy, for instance, can be of help in speeding up the convergence when dealing with non-critical fitness functions, but could be inadequate for highly multi-modal

functions since privileging the direction toward the current best solution (a basic rand/1 would be preferred in this case). The current-to-rand/1 mutation (Equation 3.19 with $K = \mathcal{U}(0,1)$ and $F^{'} = K \cdot F$) is instead preferable in problems having a strong linkage among variables (e.g. rotated functions). It is worth mentioning that the latter scheme does not require cross-over, since already contains a built-in arithmetic cross-over and so features the advantage of being rotational invariant, as showed (Takahama & Sakai 2010). In effect, it can be easily seen that arithmetic cross-over, i.e. $\mathbf{X_{offspring}} = r\mathbf{x_{parent1}} + (1-r)\,\mathbf{x_{parent2}}$ with $r = \mathcal{U}(0,1)$, since performing a simple linear combination of two points, keeps the same symmetry for the offspring no matter whether it is applied before or after a rotation of the parents. Conversely, the rotation-invariance property does not hold in those cross-over operators placing offspring on the available vertices of the hyper-rectangle built by considering parents as opposite vertices along one of its diagonals, including binary and exponential cross-over. With reference to Algorithms18 and 19, it can be seen that they operate in substantially different ways. In the former scheme, the probability of inheriting the $i-th$ gene from the first parent is exactly CR ($1-CR$ from the second parent). The latter copies a burst of genes from the first parent into a copy of the second parent starting from a random position, by iterating on the condition $\mathcal{U}(0,1) \leq CR$ (probability follows the geometric progression and decays exponentially, from which comes the name). In both cases, the expected percentage of genes to be exchanged can be fixed by setting the cross-over rate CR. While this can be trivial in the binomial version, when using the exponential scheme one needs to pay attention to the dimensionality of the problem and evaluate CR accordingly. This can be automatically achieved by establishing the amount of variable to be replaced n_e and expressing CR in terms of problem dimensionality n as:

$$CR = \frac{1}{\sqrt[n\alpha]{2}} \tag{3.22}$$

with "inheritance factor" $\alpha = \frac{n_e}{n}$, like shown in (Iacca, Neri, Mininno, Ong & Lim 2012).

Algorithm 18 Binary Cross-Over

```
procedure XoverBin(x1, x2)
    Index ← I(1, n)                ▷ Random integer number uniformly distributed in [0, n] ⊂ ℕ
    for i = 1 : n do
        if U(0, 1) ≤ CR || i == Index then
            x_off[i] ← x1[i]
        else
            x_off[i] ← x2[i]
        end if
    end for
    Output x_off
end procedure
```

DE performances depend on the choice of F and CR. If this small number of parameters to be tuned is a strength of DE, on the other hand they play an important role and should be chosen on the specific problem in order achieve the desired goal. This is obviously not a trivial issue, and many DE variants have been designed with the aim of making these parameters self-adaptive. Some of the most important versions are briefly described below, for an extensive literature review and survey see (Neri & Tirronen 2010) and (Das & Suganthan 2011). Originally, F was allowed in $[0,2]$ and CR, being a probability, in $[0,1]$. After further experiment and research these ranges have been narrowed, and common values assuring best performances should be picked in $F \in [0.5,1]$ and $CR \in [0.8,1]$ (Liu & Lampinen 2005). In the same study a population

Algorithm 19 Exponential Cross-Over

```
procedure XoverExp(x_1, x_2)
    x_off ← x_2
    Index ← I(1, n)                          ▷ Random integer number uniformly distributed in [0, n] ⊂ ℕ
    x_off[Index] ← x_2[Index]
    i ← Index + 1
    while U(0, 1) ≤ CR || i ≠ Index do
        x_off[i] ← x_1[i]
        i ← i + 1
        if i > n then
            i ← 1
        end if
    end while
    Output x_off
end procedure
```

size of about ten times the dimensionality of the problem is also suggested. Determining the right population size is the first difficulty in any population-based algorithm. Obviously, a big size is preferable in highly multi-modal problems, but it requires more time to converge, conversely a small population tends to come together quicker, so carrying other advantages and issues. Premature convergence can also happen with a high number of individuals. For instance, in GAs, when the fitness landscape does not change significantly within $\mathcal{D}$, the lack of diversity can make selection fruitless. Despite the fact that DE does not follow a fitness-based parent selection scheme this phenomenon is also present due to the fact that the number of moves is limited. Even if increasing M, one can increase the number of difference vectors, the kind of move performed by mutation is basically the same. Premature convergence or inability to reach convergence can be handled by employing multiple mutations, hybridisation with LSs and other expedients, see (Caraffini, Neri, Cheng, Zhang, Picinali & G. Iacca 2013), (Caraffini, Neri & Poikolainen 2013) and (Iacca, Neri, Caraffini, & Suganthan 2014). A more challenging problem, still under investigation, is instead the so called "stagnation", which occurs when the algorithm is not able to improve upon any individual in the population, being impossible to converge towards a promising (sub-)optimal solution even though chromosomes diversity is still high. It has also been observed that in DE also a small population in some cases can provide good results, even when smaller than the dimensionality of the problem (Neri & Tirronen 2008), and that stagnation may also occur in this case. Some modern DE variants are reported here in order to show how self-adapting parameters setting and multiple mutation schemes can be included within the optimiser.

In Self-adaptive Differential Evolution Algorithm for Numerical Optimisation (SADE), parameters are automatically adjusted on the strength of successful and failed past attempts, performed in a temporal window containing a fraction of the total number of generations. A set of multiple mutations is similarly adapted, so promoting those strategies which have performed better during the so called "Learning Period" LP. For each individual, a personal F value is drawn from a normal distribution with fixed mean (0.5) and standard deviation (0.3). Conversely, as for CR only the standard deviation (equal to 0.1) is kept fixed during the optimisation process while the mean value CR_m, initially set to 0.5, is constantly subject to refinement. With reference to the original implementation (Qin & Suganthan 2005), every 5 generations a new tuple $\langle F, CR\rangle$ has to be refreshed by means of the aforementioned distributions. After every functional call, if the relative CR value has lead to an improvement then has to be stored in a dedicated memory stack. After a cycle of 25 generations the median value of the obtained CR-distribution replaces CR_m, and the memory stack is emptied. This

mechanism assures an automatic tuning of CR on the problem even without having any previous source of information about it. Two well known DE schemes can be applied: DE/rand/1/bin (Equation 3.14, completely randomised) and DE/current-to-Best/2/bin (Equation 3.19, move towards the best individual). The same cross-over is applied in both cases, but mutations are stochastically chosen. At the beginning, the two strategies "1" and "2" have the same probability $p = 50\%$ to be applied, i.e. for each individual if $\mathcal{U}(0,1) < p$ then "1" is performed, otherwise "2". The mutation probability p is changed at the end of a learning period of LP generations (suggested value 50), where the counters ns_1 and nf_1 are incremented whether "1" makes an improvement or a failure respectively. In the same way, nf_1 and nf_2 have to be updated for "2", so that p evaluated:

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)} \tag{3.23}$$

For the sake of completeness, a modern implementation of the SADE algorithm has been released in(Qin et al. 2009). The original algorithm has been equipped here with two additional mutations, namely DE/rand/2 (Equation 3.16) and DE/current-to-rand/1 (Equation 3.19), in order to be able to perform a more diverse set of moves within $\mathcal{D}$. This version has shown to be more versatile and is the one tested in this thesis.

As simple as efficient, Self-Adapting Control Parameters in Differential Evolution (jDE) is a basic DE/rand/1/bin which has been opportunely modified in order to evolve and auto-tune scale factor and cross-over rate (Brest, Greiner, Bošković, Mernik & Žumer 2006). Each individual in the population is associated with its personal parameters, and can be seen as a triple $\langle \mathbf{x}, F, CR \rangle$. The main idea is that if a good combination of parameters has led to a better individual, after survivor selection not only the genetic patrimony but also F and CR are transmitted into the new generations. In this way parameters are automatically tuned and to some extent evolved. Values for F and CR are chosen randomly and are refreshed after a certain amount of functional calls according to a given probability τ_1, for scale factor, and τ_2 for cross-over rate. More formally, before each mutation if $\mathcal{U}(0,1) < \tau_1$ then $F = F_l + \mathcal{U}(0,1) \cdot F_u$, if $\mathcal{U}(0,1) < \tau_2$ then $CR = \mathcal{U}(0,1)$. Suggested values for the lower and the upper limits are $F_l = 0.1$ and $F_u = 0.9$. It could seem that this optimiser requires a manual tuning of two parameters, τ_1 and τ_2, as it happens in the classic DE for F and CR. Anyway, it must be said that jDE performances are less sensitive to variations of τ_1 and τ_2, which can both be set, according to the experimental tuning performed by the author, equal to 0.1. Despite the simplicity, jDE has shown to be competitive with modern and complex optimisers, both on benchmark problems and machine learning applications (Iacca, Caraffini & Neri 2014).

Another successful adaptive DE variant is the Adaptive Differential Evolution with Optional External Archive (JADE), based on a novel mutation strategy called DE/current-to-pbest/1 (Zhang & Sanderson 2009). This mutation scheme can be seen as a generalisation of Equation 3.18 (current-to-best), where the best individual $\mathbf{x_{best}}$ is replaced with a point randomly selected among the top p% best individuals in the population $\mathbf{P}$ of size M, as follow:

$$\mathbf{x_m} = \mathbf{x_j} + F_j \left(\mathbf{x}_{\mathbf{best}}^{\mathbf{p\%}} - \mathbf{x_j} \right) + F_j \left(\mathbf{x_{r_1}} - \mathbf{x_{r_2}} \right) \tag{3.24}$$

where $\mathbf{x_{r_1}}$ is randomly sampled from P and $\mathbf{x_{r_2}}$ from $\mathbf{P} \cup \mathbf{A}$ ($\mathbf{x_{r_1}} \neq \mathbf{x_{r_2}} \neq \mathbf{x_{r_j}}$). $\mathbf{A}$ is an auxiliary external archive that can either be $\mathbf{A} = \emptyset$ or contain M individuals. The additional archive can be used to preserve

population diversity. In case there are no memory constraints, $\mathbf{A}$ (initially empty) can be filled with those points failing the survivor selection procedure, meaning that at most, M individuals can be stored in a single generation cycle. In order to maintain constant size, after every generation $|\mathbf{A}| - M$ solutions are randomly deleted in case $|\mathbf{A}| > M$. One more peculiarity is on the scale factor and cross-over rate (binary cross-over is adopted) which are, also in this case, self-adaptive. For each $j-th$ individual in P a correspondent scale factor F_j and cross-over rate CR_j are generated from two distributions of probability whose mean value is adapted to the problem under study. Since F_j could potentially span on a bigger range than CR_j, a Cauchy distribution $F_j = \mathcal{C}(\mu_F, 0.1)$ is preferred, while a Gaussian is more suitable for the second parameter $CR_j = \mathcal{N}(\mu_{CR}, 0.1)$ ($\mu_F = \mu_{CR} = 0.5$ for the first iteration). After being drawn, parameters undergo a check in order to avoid invalid values, i.e. $F_j = 0$ and $CR_j \notin [0, 1]$, then mutation and cross-over take place. In case of improvement upon the old parent, both the parameters are stored in dedicated memory stacks, $\mathbf{S_F}$ and $\mathbf{S_{CR}}$, for successful F_j and CR_j respectively. At the and of every generation cycle, the Cauchy and Gaussian distributions can so be updated by means of the followings:

$$\mu_F = (1-c) \cdot \mu_F + c \cdot \frac{\sum\limits_{F \in \mathbf{S_F}} F^2}{\sum\limits_{F \in \mathbf{S_F}} F} \qquad \mu_{CR} = (1-c) \cdot \mu_{CR} + c \cdot \frac{\sum\limits_{CR \in \mathbf{S_{CR}}} CR}{|\mathbf{S_{CR}}|} \tag{3.25}$$

with $c \in [0, 1]$.

A different approach has been used in Ensemble of Parameters and Mutation Strategies Differential Evolution (EPSDE), where as an alternative to adaptation, F and CR are not drawn from Gaussian or Cauchy distributions but rather randomly picked up from a pool of promising values. According to the original paper (Mallipeddi et al. 2010) optimal pools are $\{0.5, 0.9\}$ and $\{0.1, 0.5, 0.9\}$ for F and CR respectively. The same logic is applied for mutation and cross-over: during each generation for each individual, a perturbation logic is randomly selected from a pool containing DE/current-to-pbest/1 and DE/current-to-rand/1 in order to generate a mutant vector. Subsequently one between binary and exponential cross-over is selected from a third pool for mating, and finally they are combined together and applied by using the previously picked values for F and CR. Once again, it is worthwhile noting that in case of DE/current-to-rand/1 no cross-over has to be performed since it is implicitly present in the mutation strategy.

In the wake of JADE, a recent publication (Islam, Das, Ghosh, Roy & Suganthan 2012) has introduced a novel mutation namely DE/current-to-gr_best/1 and a new "p-best" cross-over, in order to enhance exploration of the search space. For each j-th individual ($j = 1, 2, 3, ..., M$) of the population the following scheme is applied: first a mutant vector has to be generated by means of

$$\mathbf{x_m} = \mathbf{x_j} + F_j(\mathbf{x_{q\_best}} - \mathbf{x_j}) + F_j(\mathbf{x_{r_1}} - \mathbf{x_{r_2}}) \tag{3.26}$$

with $\mathbf{x_{q\_best}}$ the best point of a set containing a percentage q of randomly selected individuals from the population. Then, a binary crossover is applied to the mutant vector and a second vector that has to be randomly sampled among the best p individuals in the population. Unlike q, which is a pre-established value,

p is updated run-time depending on the numbers of passed generations g: $p = ceil\left[\frac{n}{2} \cdot \left(1 - \frac{g-1}{g_{max}}\right)\right]$ (g_{max} is the maximum number of allowed generations). Scale factor and cross-over rate are drawn from Cauchy and Gaussian distributions with variable mean value μ_F and location parameter μ_{CR}. As in JADE, two memory stacks containing successful values for F_j and CR_j are needed ($\mathbf{S_F}$ and $\mathbf{S_{CR}}$ respectively). Updating formulas differ from the former proposed for JADE:

$$\mu_F = W_F \cdot \mu_F + (1 - W_F) \cdot \sum_{F \in \mathbf{S_F}} \left(\frac{F^z}{|\mathbf{S_F}|}\right)^{\frac{1}{z}} \tag{3.27}$$

$$\mu_{CR} = W_{CR} \cdot \mu_{CR} + (1 - W_{CR}) \cdot \sum_{F \in \mathbf{S_{CR}}} \left(\frac{CR^z}{|\mathbf{S_{CR}}|}\right)^{\frac{1}{z}} \tag{3.28}$$

with $z = 1.5$, $W_f = 0.8 + 0.2 \cdot \mathcal{U}(0,1)$ and $W_f = 0.9 + 0.1 \cdot \mathcal{U}(0,1)$. The use of power mean for μ_F, plus Cauchy distribution, leads to large perturbations, thus minimising the risk of premature convergence. Gaussian distribution is preferable for CR_m since confined in a smaller range, and the long tail of Cauchy distribution could be inadequate, generating too many values out of $[0, 1]$ thus requiring multiple saturation. This algorithm, named Modified Differential Evolution with p-Best Crossover (MDE-pBX), performs well in low dimensions, e.g. see Appendix E, and has been used for comparison in this work.

Before concluding this literature review, an interesting DE-based variant of a class of algorithms called EDA, is briefly described since involved in the experiment set-up of this thesis. This algorithm, namely compact Differential Evolution (cDE), in the fashion of popular EDA algorithms such as (Baluja 1994), makes use of an explicit probabilistic models for sampling promising candidate solutions, see (Pelikan, Goldberg & Lobo 2000), which replaces the population of individuals. Evolution is stochastically driven by building and progressively updating the probabilistic model starting with a uniform distribution, encoding al the possible solutions in the search space, and ending with the model generating only (sub)optimal solutions. In cDE (Mininno, Neri, Cupertino & Naso 2011), this principle is adopted in order to mimic the behaviour of a DE despite not having the burden of storing in memory the actual population, which is indeed replaced with a Probability Density Function (PDF). This approach attempts to reproduce the beneficial aspects of a population based method while dealing with strict memory constraint.

The expression "compact" algorithm was first introduced in (Harik, Lobo & Goldberg 1999) for the compact Genetic Algorithm (cGA), which was then improved in (Harik, Lobo & Sastry 2006), and finally provided with a real-valued chromosome representation, from which comes the name Real-Coded Compact Genetic Algorithm (rcGA), in (Mininno, Cupertino & Naso 2008*a*). The real-valued encoding together with a novel representation for a population's PDF via polynomial approximation of truncated Gaussian distribution, has been then taken from rcGA and extended to other optimisation families such as cDE, but also compact Bacterial Foraging Optimisation (cBFO) and compact Particle Swarm Optimisation (cPSO), see (Iacca, Neri & Mininno 2012) and (Neri, Mininno & Iacca 2013) respectively.

In cDE the optimisation process takes place in a normalised domain $[-1, 1]$, thus the initial solution need to be scaled into this interval and then the final output must be re-sized into the original bounds. A $(2 \times n)$-matrix,

namely perturbation vector $\mathbf{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$, must be pre-allocated with $\boldsymbol{\mu} = \emptyset$ and $\boldsymbol{\sigma} = 10ones\,(n)$. Such a high value for $\boldsymbol{\sigma}$ makes the $\mathbf{PV}$ distribution behave, at the early stage of the optimisation process, as a uniform distribution (plotted in Figure 3.5 for a single dimension). A solution $\mathbf{x_e}$, "elite", initially sampled from $\mathbf{PV}$, must be kept up to date with the best individual ever found during the search, which takes place by working out a mutant vector via a classic DE/rand/1 mutation scheme. In order to apply the scheme, at each generic iteration t three solutions, $\mathbf{x_r}$, $\mathbf{x_s}$ and $\mathbf{x_t}$, are drawn from the distribution. The replacement procedure cannot obviously happen in the classic way, since the population is lacking, but 1-to-1 survivor selection still occurs and the information coming from this comparison is used to update $\mathbf{PV}$. For each i-th design variable forming a generic vector to be sampled, a truncated Gaussian PDF, i.e. tails are taken off and then resized in $[-1, 1]$ so keeping its area unitary, is defined making use of $\boldsymbol{\mu}\,[i]$ and $\boldsymbol{\sigma}\,[i]$ from $\mathbf{PV} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$:

$$PDF\,(truncNorm\,(\mathbf{x}[i])) = \frac{e^{-\frac{(\mathbf{x}-\boldsymbol{\mu}[i])^2}{2\boldsymbol{\sigma}[i]^2}} \cdot \sqrt{\frac{2}{\pi}}}{\boldsymbol{\sigma}\,[i] \cdot \left(\text{erf}\left(\frac{\boldsymbol{\mu}[i]+1}{\sqrt{2}\boldsymbol{\sigma}[i]}\right) - \text{erf}\left(\frac{\boldsymbol{\mu}[i]-1}{\sqrt{2}\boldsymbol{\sigma}[i]}\right)\right)} \tag{3.29}$$

where erf is the error function, see (Gautschi 1972). The corresponding Cumulative Distribution Function (CDF) is derived by means of Chebyshev polynomials by following the procedure in (Cody 1969). It must be observed that the co-domain of CDF is $[0, 1]$, so, in order to draw the design variable $\mathbf{x_r}[i]$ from PV the inverse function of CDF must first be calculated, and then evaluated in correspondence of a uniformly distributed random number in that interval: $\mathcal{U}\,(0, 1)$, as shown in Figure 3.6. As mentioned above, in order to obtain the phenotype value in the original interval $[a, b]$ here indicated with $\mathbf{x_{phen}}$, the following operation must be performed:

$$\mathbf{x_{phen}}[i] = \mathbf{x_r}[i] \cdot \frac{(b-a)}{2} + a. \tag{3.30}$$

Once all the required points have been drawn, they undergo mutation so generating a mutant vector $\mathbf{x_m}$, that can be mated via either binary or exponential cross-over with the elite solution x_e. The produced offspring $\mathbf{x}_{off}$ is expected to outperform, in term of fitness value, the elite solution. If this happen the elite solution must be replaced with the new point, while regardless of the outcome of this comparison two vectors: $\mathbf{x_{winner}}$ and $\mathbf{x_{loser}}$ have to be filled, with the best and worst value respectively, after each iteration for updating $\mathbf{PV}$ components:

$$\boldsymbol{\mu}^{(t+1)} = \boldsymbol{\mu}^{(t)} + \frac{1}{N_p} \cdot (\mathbf{x_{winner}} - \mathbf{x_{loser}})\,, \tag{3.31}$$

where N_p is a parameter called virtual population size, and subsequently $\boldsymbol{\sigma}^2$:

$$\boldsymbol{\sigma}^{2\ (t+1)} = \boldsymbol{\sigma}^{2\ (t)} + \boldsymbol{\mu} \circ \boldsymbol{\mu}^{\ (t)} - \boldsymbol{\mu} \circ \boldsymbol{\mu}^{\ (t+1)} + \frac{1}{N_p} (\mathbf{x_{winner}} \circ \mathbf{x_{winner}} - \mathbf{x_{loser}} \circ \mathbf{x_{loser}}) \tag{3.32}$$

These operations are repeated over time for a given budget. For parameters setting see Appendix D, referring to the perturbation DE/rand/1/exp proposed in (Mininno et al. 2008*a*). The main disadvantage of cDE lies on the sampling procedure, which introduces a certain temporal over-head since individuals have to be sampled rather than simply accessed from the population. Where possible, the number of samplings must be kept minimum. This is the idea behind its light version, compact Differential Evolution light (cDE-light), which has been proposed in (Iacca, Caraffini & Neri 2012) to address not only the memory saving necessities, but also

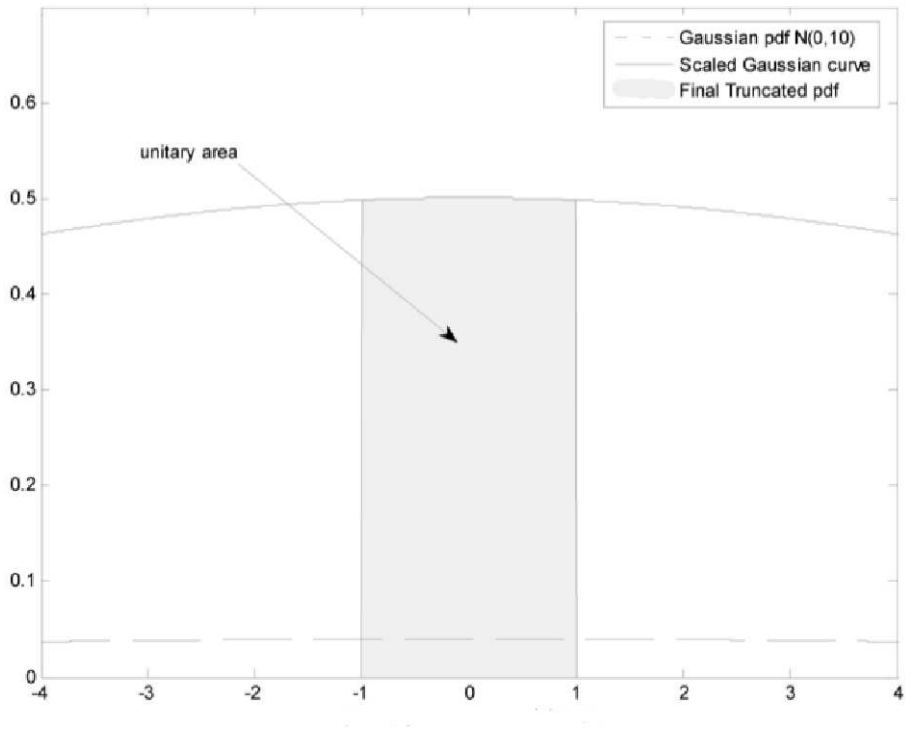
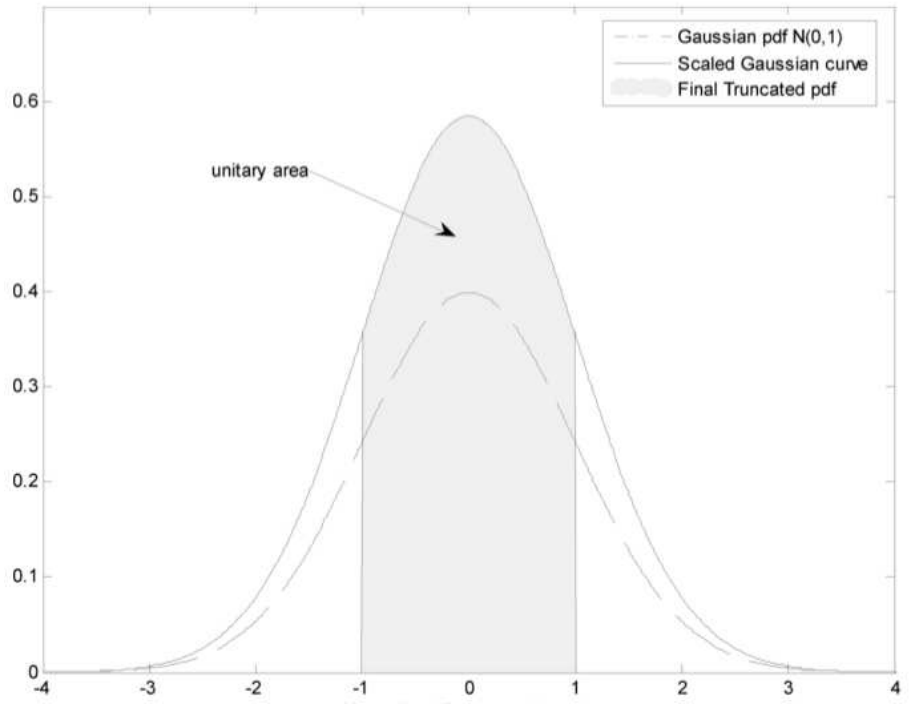


Figure. 3.5. Truncated Gaussian PDF at the beginning (top) and during (bottom) of the optimisation process.



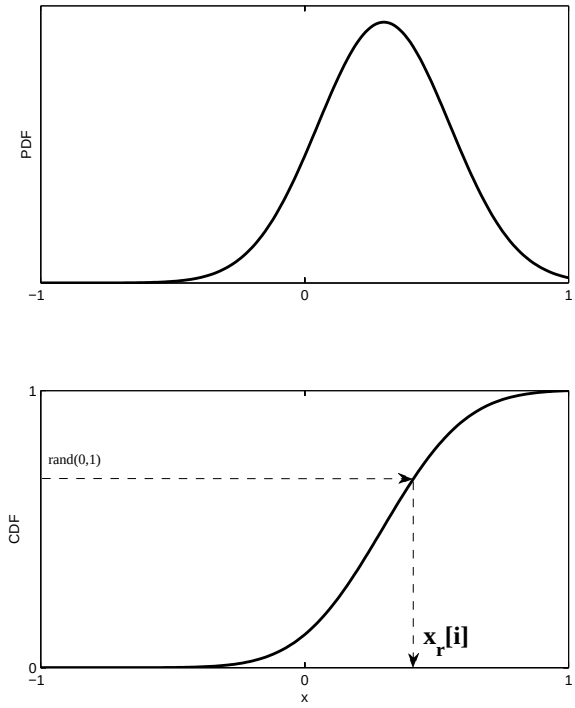


Figure. 3.6. PDF, CDF and sampling mechanism.

real-time requirements without a performance loss. Both mutation and cross-over have been modified ad-hoc as follows.

Since the Chebyshev polynomials are computationally expensive to use, in the so-called "Mutation Light" the truncated Gaussian PDF has been intentionally confused with a Gaussian PDF ($\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, where C is a diagonal covariance matrix containing $\boldsymbol{\sigma}^2$ as its diagonal). In this light, the three solutions $\mathbf{x_r}$, $\mathbf{x_s}$ and $\mathbf{x_t}$ can be seen as stochastic variables:

$$\left.\begin{array}{l}\mathbf{x_r}\\ \mathbf{x_s}\\ \mathbf{x_t}\end{array}\right\} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C}). \tag{3.33}$$

By applying the properties of normally distributed variables, it can be easily shown that applying the DE/rand/1 scheme is equivalent to sample $\mathbf{x_m}$[i] ($\forall i = 1, 2, \ldots, n$) from:

$$\mathbf{x_m}[i] \sim \mathcal{N}\left(\boldsymbol{\mu}[i], (1 + 2 \cdot F^2) \cdot \boldsymbol{\sigma}^2[i]\right) \tag{3.34}$$

In fact, considering without a loss of generality a mono-dimensional case, a linear combination of M Gaussian independent random variables, each one having mean value equal to μ_m and variance equal to ${\sigma_m}^2$ ($m = 1, 2, \ldots, M$), is still a Gaussian random variable with average $\mu' = \alpha_1 \cdot \mu_1 + \alpha_2 \cdot \mu_2 + \cdots + \alpha_M \cdot \mu_M = \sum_{m=1}^{M} \alpha_m \cdot \mu_m$ and variance $\sigma'^2 = \alpha_1^2 \cdot \sigma_1^2 + \alpha_2^2 \cdot \sigma_2^2 + \cdots + \alpha_M^2 \cdot \sigma_M^2 = \sum_{m=1}^{M} \alpha_m^2 \cdot \sigma_m^2$. The same reasoning

applied to Formula 3.14 implies:

$$\begin{cases} \boldsymbol{\mu'}[i] = \boldsymbol{\mu}[i] + F \cdot \boldsymbol{\mu}[i] - F \cdot \boldsymbol{\mu}[i] = \boldsymbol{\mu}[i] \\ \boldsymbol{\sigma'}^2[i] = \boldsymbol{\sigma}^2[i] + F^2 \cdot \boldsymbol{\sigma}^2[i] + (-F)^2 \cdot \boldsymbol{\sigma}^2[i] + F^2 = \left(1 + 2 \cdot F^2\right) \cdot \boldsymbol{\sigma}^2[i] \end{cases} \tag{3.35}$$

It must be highlighted that this is true only under the hypothesis of statistic independence among the variables. For the sake of clarity this result is proven below by evaluating the first and the second order moment of the stochastic variable $\mathbf{x_m}$ (in order to avoid confusion with vector matrices angle brackets are use for the expected value $E\langle\cdot\rangle$). The mean value can be easily found:

$$E\left\langle \mathbf{x_t}[i] + F\left(\mathbf{x_r}[i] - \mathbf{x_s}[i]\right)\right\rangle = E\left\langle \mathbf{x_t}[i]\right\rangle + F \cdot E\left\langle \mathbf{x_r}[i]\right\rangle - F \cdot E\left\langle \mathbf{x_s}[i]\right\rangle = \boldsymbol{\mu}[i] \tag{3.36}$$

$\mathbf{C}$ requires a bit more processing. Every i-th element in its diagonal $\boldsymbol{\sigma}$ can be worked out in two steps, by decomposing the problem in two terms. First, let us consider the random variable $\mathbf{y} = \mathbf{x_r} - \mathbf{x_s}$, as follows:

$$\boldsymbol{\mu_y}[i] = E\left\langle \mathbf{x_r}[i] - \mathbf{x_s}[i]\right\rangle = 0 \tag{3.37}$$

and

$$\begin{gathered} \boldsymbol{\sigma_y}^2[i] = E\left\langle \left(\mathbf{y}[i] - \boldsymbol{\mu_y}[i]\right)^2\right\rangle = E\left\langle \left(\mathbf{x_r}[i] - \mathbf{x_s}[i]\right)^2\right\rangle = \\ = E\left\langle \mathbf{x_r}[i]^2 - 2 \cdot \mathbf{x_r}[i] \cdot \mathbf{x_s}[i] - \mathbf{x_s}[i]^2\right\rangle = \\ = E\left\langle \mathbf{x_r}[i]^2\right\rangle - 2 \cdot E\left\langle \mathbf{x_r}[i] \cdot \mathbf{x_s}[i]\right\rangle + E\left\langle \mathbf{x_s}[i]^2\right\rangle \end{gathered} \tag{3.38}$$

Now, since the two variables are independent:

$$E\left\langle \mathbf{x_r}[i] \cdot \mathbf{x_s}[i]\right\rangle = E\left\langle \mathbf{x_r}[i]\right\rangle \cdot E\left\langle \mathbf{x_s}[i]\right\rangle = \boldsymbol{\mu}[i]^2 \tag{3.39}$$

Moreover, considering the definition of variance, the following relations hold:

$$E\left\langle \mathbf{x_r}[i]^2\right\rangle = Var\left\langle \mathbf{x_r}[i]\right\rangle + E\left\langle \mathbf{x_r}[i]\right\rangle^2 = \boldsymbol{\sigma}^2[i] + \boldsymbol{\mu}[i]^2 \tag{3.40}$$

Replacing 3.39 and 3.40 in the equation 3.38, we have:

$$E\left\langle \left(\mathbf{x_r}[i] - \mathbf{x_s}[i]\right)^2\right\rangle = \boldsymbol{\sigma}^2[i] + \boldsymbol{\mu}[i]^2 + \boldsymbol{\sigma}^2[i] + \boldsymbol{\mu}[i]^2 - 2 \cdot \boldsymbol{\mu}[i]^2 = 2\boldsymbol{\sigma}^2[i] \tag{3.41}$$

Summarizing, we obtain the following condition:

$$\mathbf{y} \sim \begin{cases} \boldsymbol{\mu_y} = \emptyset \\ \boldsymbol{\sigma_y}^2 = 2\boldsymbol{\sigma}^2 \end{cases} \tag{3.42}$$

Finally, the variance vector for $\mathbf{x_{off}}$ can then be calculated as:

$$\begin{gathered} E\left\langle \left(\mathbf{x_t}[i] - \boldsymbol{\mu}[i]\right)^2 \cdot\right\rangle + F^2 E\left\langle \mathbf{y}[i]^2\right\rangle + 2 \cdot F \cdot E\left\langle \mathbf{y}[i] \cdot \left(\mathbf{x_t}[i] - \boldsymbol{\mu}[i]\right)\right\rangle = \\ \boldsymbol{\sigma}^2[i] + F^2 \cdot \left(2 \cdot \boldsymbol{\sigma}^2[i]\right) + 2 \cdot F \cdot E\left\langle \mathbf{y}[i]\right\rangle \cdot E\left\langle \left(\mathbf{x_t}[i] - \boldsymbol{\mu}[i]\right)\right\rangle = \left(1 + 2 \cdot F^2\right) \cdot \boldsymbol{\sigma}^2[i] \end{gathered} \tag{3.43}$$

It must be said that this operation is not fully equivalent to sampling three solutions from a truncated Gaussian

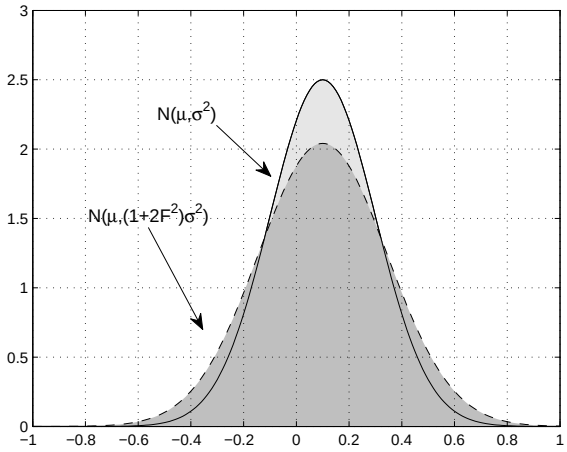


Figure. 3.7. Truncated Gaussian distributions displaying virtual population (solid line) and modified distribution (dashed line) for provisional offspring sampling during mutation light

distribution, but is an approximation which neglects the tails of the distribution outside the interval $[-1, 1]$. In other words, while the representation of the population and their update, see formulas (3.31) and (3.32), is performed by means of truncated Gaussian distributions, the generation of new candidate solutions by means of mutation light is done considering that the population is encoded by means of actual (non-truncated) Gaussian distributions. In order to better understand what happens by transforming the original distribution, the outcome of this transformation is graphically depicted in Figure 3.7. In addition, it must be noted that the $\mathbf{PV}$ matrix itself is not a distribution, but only an useful way of storing, for each i-th axis, $\boldsymbol{\mu}[i]$ and $\boldsymbol{\sigma}[i]$ so that the realisation of each design variable $\mathbf{x}[i]$ can be drawn.

Exponential cross-over has been replaced as well with its computational saving version: Cross-Over Light,

Algorithm 20 Exponential Cross-Over Light

procedure XOVEREXPLIGHT($\mathbf{x_1}, \mathbf{x_2}$)
  $i_{start} = \mathcal{I}(1, n)$
  $\mathbf{x_{offspring}} \leftarrow \mathbf{x_1}$
  $\mathbf{x_{offspring}}[i_{start}] \leftarrow \mathbf{x_2}[i_{start}]$
  $xoverL \leftarrow round\left(\frac{\log(rand(0,1))}{\log(CR)}\right)$
  $i \leftarrow i_{start} + 1$
  $j \leftarrow 1$
  while $i \neq i_{start}$ && $j \leq xoverL + 1$ **do**
    $\mathbf{x}_{off}[i] \leftarrow \mathbf{x_1}[i]$
    $i \leftarrow i + 1$
    $j \leftarrow j + 1$
    if $i == n$ **then**
      $i \leftarrow 1$
    end if
  end while
  Output $\mathbf{x_{offspring}}$
end procedure

see Algorithm 20, with the aim of reducing the algorithmic time overhead. When the exponential cross-over is applied, the probability that the first gene is inherited from the provisional offspring is 1. Subsequently, the adjacent gene of the provisional offspring has a probability equal to CR to be transmitted to the offspring. The

third gene has a probability of CR^2. The generic $m-th$ gene is associated with a probability CR^{m-1}. In order to simplify the notation, let us consider the deterministic copy of the first gene and the probabilistic copy of the other genes as independent events. The probability that m genes, on the top of the first one, are copied from the provisional offspring to the actual offspring is CR^m. More formally the discrete probability Pr that the crossover length $xoverL$ is equal to m is known as geometric distribution and is given by:

$$Pr\left(xoverL = m\right) = CR^m \tag{3.44}$$

where $m = 1, 2, \ldots, n-1$. In order to extract the number of genes m to be copied, it is enough to apply the inverse formulas and obtain:

$$xoverL \sim round\left(\log_{CR}\left(\mathcal{U}\left(0,1\right)\right)\right) = round\left(\frac{\log\left(\mathcal{U}\left(0,1\right)\right)}{\log\left(CR\right)}\right) \tag{3.45}$$

where the last equality is due to the change of base of a logarithm. In other words, crossover light consists of performing the deterministic copy and subsequently the copy of $xoverL$ genes, where $xoverL$ is determined by formula (3.45). For the sake of clarity, the pseudo-code of cDE-light (employing persistent elitism strategy), together with the one relative to cDE, are united in Algorithm 21. cDE-light has proven to perform similarly to its predecessor cDE despite working remarkably faster, see (Iacca, Caraffini & Neri 2012), in particular when the dimensionality of the problem grows. Thanks to their minimalistic memory footprint, both these DE variants have been successfully used for embedded engineering applications, such as trajectory planning optimisation for robotic arms(Iacca, Caraffini, Neri & Mininno 2012). The light variant in particular, has shown to be useful when implemented on-board a micro-controller for real-time optimisation robotic applications, see (Iacca, Caraffini & Neri 2013).

Many others population based optimisers have been designed during the last decade, and also their "compact" version have been implemented as previously mentioned. Nonetheless, a detailed inspection of all the existing techniques would fall out of the scope of this thesis. A recapitulatory overview of the mentioned algorithms is given in Table 3.1

Table 3.1. General-purpose algorithms overview (t and m refer to time and space complexity respectively).

Algorithm	**Solutions Representation**	**Generation cycle**	**Variation operators**	**Comments**
		SIMULATED ANNEALING		
SA	*Real-valued*	*Non Elitist*	*Stochastic perturbation*	*Single-Solution*
nuSA	"	"	*Non-uniform Perturbation*	"
		EVOLUTIONARY ALGORITHMS		
GA	*-Binary String* *-Real-valued Vector*	*-Generational* *-Steady state*	*Cross-Over & Mutation*	*Variety of replacement schemes*
ES	*Real-valued* *Triplets* $\langle \mathbf{x}, \boldsymbol{\sigma}, \boldsymbol{\alpha} \rangle$	*"Comma" or "Plus"*	*Recombination & Mutation*	*self-adaptive*✓
CMA-ES	$\mathbf{x}$, σ, $\mathbf{C}$	*comma* (λ, μ)	*Intermediary-Gaussian*	$t \sim \mathcal{O}\left(n^3\right) m \sim \mathcal{O}\left(n^2\right)$
$(1+1)$*-CMA-ES*	$\mathbf{x}$, σ, $\mathbf{A}$	*elitist single-solution*	*Gaussian mutation*	$t \sim \mathcal{O}\left(n^2\right) m \sim \mathcal{O}\left(n^2\right)$
sep-CMA-ES	*Tuple* $\langle \mathbf{x}, \boldsymbol{\sigma} \rangle$	*comma* (λ, μ)	*Intermediary-Gaussian*	$t \sim \mathcal{O}\left(n\right) m \sim \mathcal{O}\left(n\right)$ ✓
EP	*Tuple* $\langle \mathbf{x}, \boldsymbol{\sigma} \rangle$	*plus* $(\mu + \mu)$	*Gaussian Perturbation*	*self-adaptive* ✓
GP	*Parse Trees*	*Generational*	*Cross-Over & Mutation*	*Evolves a model* ✓
		SWARM INTELLIGENCE		
PSO	*Real-valued* $\langle \mathbf{x_p}, \mathbf{x_p^b}, \mathbf{x_p^w}, \mathbf{v_p} \rangle$	*1-to-1 Spawning*	*Velocity update &* *Position perturbation*	*Replacement occurs immediately*
CLPSO	*Swarm*	"	"	*Build* $\mathbf{x_p^{rb}}$ *vector*
ISPO	*Single-Particle*	"	"	*Moves along the axes*
CCPSO2	*Sub-swarms* $(no\ velocity\ \mathbf{v_p})$	"	*Cauchy/Gaussian* *move on sub-positions*	*Large Scale* ✓
		DIFFERENTIAL EVOLUTION		
DE	*Real-valued*	*1-to-1 Spawning*	*Mutation & Cross-Over*	*Replacement occurs at the end*
SADE	$\langle \mathbf{x}, F, CR \rangle$	"	*Multiple mutations &* *Binary Cross-Over*	*Self-adaptive* (F, CR) ✓
jDE	"	"	*rand/1/bin*	*Self-adaptive* (F, CR) ✓
JADE	"	"	*current-to-pbest/1/bin*	*Self-adaptive* (F, CR) ✓ & $archive\ \mathbf{A}$
EPSDE	"	"	*Multiple* *mutations & Cross-Over*	*Pools of parameters* $F\ and\ CR$✓
MDE-pBX	"	"	*current-to-gr_best/1/pBX*	*Self-adaptive* (F, CR) ✓
cDE	*real-valued* **PV** *PDF*	*Elitist* *(also non Elitist)*	*rand/1/exp* (any scheme is applicable)	*Memory Footprint*✓
cDE-light	*real-valued* **PV** *PDF*	*Elitist* *(also non Elitist)*	*Mutation Light* *Cross-Over Light*	*Memory Footprint*✓ *Real-time*✓

Algorithm 21 Compact Differential Evolution/Light

$t = 1$
$boldsymbol{\mu}^t \leftarrow \emptyset$
$\boldsymbol{\sigma}^t \leftarrow \lambda ones\,(n)$ ▷ e.g. $\lambda = 10$
$\mathbf{x_{elite}} \sim \mathbf{PV}$
while *Budget available* **do**
**** for implementing cDE ****
$\left.\begin{matrix}\mathbf{x_r}\\ \mathbf{x_s}\\ \mathbf{x_t}\end{matrix}\right\} \sim \mathbf{PV^{(t)}}$
$\mathbf{x_m} \leftarrow \mathbf{x_{r_r}} + F\,(\mathbf{x_{r_s}} - \mathbf{x_{r_t}})$ ▷ DE/rand/1, Equation 3.14
$\mathbf{x_{offspring}} \leftarrow$XOVEREXP$(\mathbf{x_{elite}}, \mathbf{x_m})$ ▷ Algorithm 19
**** for implementing cDE Light ****
prepare $\mathbf{PV_{light}}^{(t)}$ *from* $\mathbf{PV}^{(t)}$ *and apply mutation light:*
$$\mathbf{x_m} \sim \mathbf{PV_{light}}^{(t)} = \begin{bmatrix} \boldsymbol{\mu}^{(t)}\,[1] & \boldsymbol{\mu}^2\,[2] & \dots & \boldsymbol{\mu}^{(t)}\,[n] \\ \left(1+2\cdot F^2\right)\cdot \boldsymbol{\sigma^2}^{(t)}\,[1] & \left(1+2\cdot F^2\right)\cdot \boldsymbol{\sigma^2}^{(t)}\,[2] & \dots & \left(1+2\cdot F^2\right)\cdot \boldsymbol{\sigma^2}^{(t)}\,[n] \end{bmatrix}$$
$\mathbf{x_{offspring}} \leftarrow$XOVEREXPLIGHT$(\mathbf{x_{elite}}, \mathbf{x_m})$ ▷ Algorithm 20
**** common part ****
if then $f\left(\mathbf{x_{offspring}}\right) \leq f\left(\mathbf{x_{elite}}\right)$
$\mathbf{x_{loser}} \leftarrow \mathbf{x_{elite}}$
$\mathbf{x_{elite}} \leftarrow \mathbf{x_{offspring}}$
$\mathbf{x_{winner}} \leftarrow \mathbf{x_{offspring}}$
else
$\mathbf{x_{winner}} \leftarrow \mathbf{x_{elite}}$
$\mathbf{x_{loser}} \leftarrow \mathbf{x_{offspring}}$
end if
$\boldsymbol{\mu}^{(t+1)} = \boldsymbol{\mu}^{(t)} + \frac{1}{N_p}\left(\mathbf{x_{winner}} - \mathbf{x_{loser}}\right)$ ▷ Equation 3.31
$\boldsymbol{\sigma}^{2\ (t+1)} = \boldsymbol{\sigma}^{2\ (t)} + \boldsymbol{\mu} \circ \boldsymbol{\mu}^{\ (t)} - \boldsymbol{\mu} \circ \boldsymbol{\mu}^{\ (t+1)} + \frac{1}{N_p}\left(\mathbf{x_{winner}} \circ \mathbf{x_{winner}} - \mathbf{x_{loser}} \circ \mathbf{x_{loser}}\right)$ ▷ Equation 3.32
$t \leftarrow t + 1$ ▷ $\mathbf{PV}^{(t+1)}$ becomes the current $\mathbf{PV}$
$\mathbf{PV^{(t)}} = [\boldsymbol{\mu}^{(t)}, \boldsymbol{\sigma}^{(t)}]$ ▷ Update next generation $\mathbf{PV}$
end while
Output $\mathbf{x_{elite}}$

Chapter 4

Memetic Algorithms & Memetic Computing

This chapter is entirely devoted to the category of those optimisers which, rather than employing a stand-alone logic, make use of hybridisation, connection and interaction of multiple techniques and structures in order to optimise a specific given problem. These algorithms cover an important role in CIO, having great potential and being the subject of growing research. Most hybrid algorithms, designed by adding to general-purpose evolutionary frameworks (see Chapter 3.1), a fine refinement routine (see Chapter 2.2), can all be referred to as different manifestations of a more general class called MA. A further generalisation can be also made, by grouping any possible combination of different operators under the umbrella name MC.

The former term, MA, was introduced for the first time in CIO by Pablo Moscato in 1989 (Moscato 1989), referring to the concept of "*meme*" coined by Richard Dawkins as *the basic unit of cultural transmission, or imitation* (Dawkins 1976). Thus, the main metaphor behind MA, does not rely on biological evolution, but extends this concept, making use of Dawkins' Universal Darwinism theory, where evolution also exists in all those complex systems exhibiting the processes of inheritance, variation and selection, as it happens with elements of culture that pass on to new generations. As an idea, a promising meme can be shared, exchanged within a community, adapted or refined. Regardless of the underlying metaphor, the concept of exchanging information between different operators has been developed by researchers and MAs have become more and more popular. MA can now be considered as an independent discipline within CIO and this term has assumed a widely accepted meaning referring to a *population-based optimiser composed by an EA and a list of LS algorithms activated within the evolutionary framework* (Hart, Krasnogor & Smith 2004). For instance, Genetic Local Search Algorithms (GLSA) falls in this category, since processing some solutions in the population of a standard GA via hill-climber routines, or evolving micro-populations, i.e. small population sizes of about 5 individuals behave like a LS, see (García-Martínez & Lozano 2008). Similarly, Lamarckian-EAs make use of classic local searchers in order to modify the genotype of some individuals in the population so driving the evolution towards promising points, while in Baldwinian-EAs only the phenotype gets modified, see (Geoffry E. Hinton and 1987) and (Whitley, Gordon & Mathias 1994). To some extent, the Cultural Algorithm (CA) can also be seen as an MA. In this case the population space, e.g. of a GA, is flanked by a further "belief" space which is updated after each iteration by considering the best individuals in the population. The belief space is used to alter, by means of an "influence" function, the genotype of each offspring in order to speed up the

evolutionary process. The pattern EA + LS is not difficult to understand: general purpose algorithms are good at exploring the search space, but not precise in refining a solution, and vice-versa. Thus, the combination of these two components is beneficial. The local search algorithms help to find "super-fit" individuals that can enhance and speed-up the evolutionary mechanism, and at the same time round off the final solution with small adjustments in the surroundings of the optimum. In some ways, the presence of the local searchers brings some useful knowledge, which is employed for specialising the EA part to tackle the problem under study. How to balance the budget between exploratory and exploitative search is one of the difficulties to deal with while using this approach. Coordination plays an important role.

MC further extends the previous scheme. Generally speaking MC represents a broad slice of Computer Science dealing with different aspects of AI, through multiple interacting operators whose interaction actualises intelligent systems. In CIO, these interacting entities (memes) can be seen as *units of information encoded in computational representations for the purpose of problem solving* (Ong et al. 2010).

As an example of MA, in (Molina, Lozano, García-Martínez & Herrera 2010*a*) a powerful optimiser is introduced by combining two popular algorithms, namely GA and CMA-ES. The resulting algorithm performs excellently on low-dimensional problems. The main task of the GA is obviously exploration, while CMA-ES is used here for LS. It must be noticed that CMA-ES provides best results over mono-modal functions and by setting the initial step-size σ to a small value, it can indeed be seen as a proper LS routine. In order to guarantee a good exploration of the search space, the Steady State GA has been equipped with the negative assortative mating selection strategy described in Chapter 3.1.1, the BLX-α cross-over operator (Equation 3.1) and the BGA mutation (Equation 3.2). Cyclically, GA is run for n_{frec} fitness functional calls and then temporarily arrested in order to fill a set, $\mathbf{S_{LS}}$, with all those individuals that either have never been refined by CMA-ES or have gone through LS refinement obtaining a fitness improvement bigger than a predefined value δ_{LS}^{min}. The best individual from S_{LS} (or from the steady-state GA population in case $\mathbf{S_{LS}} = \emptyset$) becomes at this stage the mean value for the CMA-ES Gaussian distribution, while the distance between the mean value and its closest point in the steady-state GA population gives the step-size σ, and the LS can take place by running over I_{str} (Intensity stretch factor) fitness evaluations. I_{str} is worked out according the following formula: $n_{frec} = I_{str} \cdot \frac{1-r_{L/G}}{r_{L/G}}$. A good balance among exploration/exploitation can be reached by setting the Local/Global evaluations ratio ($r_{L/G}$) equal to 0.5, which gives the same amount of fitness evaluations to both the parts. In case the LS is applied on a new point, i.e. never being processed in the past generations, strategy parameter values and internal variables for CMA-ES have to be initialised as in (Hansen & Ostermeier 2001). This scenario is not likely to happen many times, since the steady-state configuration tends to keep good solutions in the population for a long time. It is therefore more probable that a promising solution, which has already shown an improvement greater than δ_{LS}^{min}, is selected to undergo refinement. In this case, the authors proposed to use the so called "LS chain" process, in which the final configuration reached by CMA-ES is used as the initial configuration for the next application. Despite its effectiveness, this approach is both computationally and memory expensive, since for each solution being processed an $n \times n$ covariance matrix must be evaluated and stored in memory. In LSOP this implementation could be too slow or even not applicable. For this reason, a second variant has been designed for handling large scale problems (Molina, Lozano & Herrera 2010), i.e. problem dimensionality bigger than

100, by simply replacing CMA-ES with a lighter Sub-grouping Solis Wets (Algorithm 11, Chapter 2.2.8). Potentially, this algorithm can be equipped with any LS routine and regardless of the chosen one, the generic framework can be addressed with the name Memetic Algorithm with Local Search Chain (MA-LSCh). For parameters setting of the two variants, refer to MA-LSCh-CMA and MA-LSCh-SSW (see Appendix D). Since MA-LSCh-CMA does not perform well on LSOP, and vice-versa MA-LSCh-SSW degrades its performances in low dimensional problems, in this thesis the first variant has been tested up to 100 dimensions, while the second for higher dimensionality. Numerical results have been then merged together and reported, see Appendix E, under the generic name MA-LSCh.

A borderline case is represented by Micro-Differential Evolution with extra moves along the Axes (μDEA), a computationally light and memory saving optimiser proposed in (Caraffini, Neri & Poikolainen 2013). Even though DE does not belong to the EA class, and so strictly speaking does not literally fulfil the former definition, it can still be seen as an MA since having a population based core hybridised with a LS. It must be said that the very first attempts of combining EAs with LS have been made when DE, PSO and other nowadays consolidated optimisers were still under development. As a consequence, the concept of MA has then implicitly been broadened assuming a more general meaning, and many memetic DEs and PSOs have been designed, see (Neri & Tirronen 2008) and (Wang, Moon, Yang & Wang 2012) respectively.

In the fashion of MAs, μDEA implements a classic DE/rand/1/exp with a population size narrowed to $M = 5$ individuals, equipped with the S operator, see Chapter 2.2.4, in order to provide a further move along the axis on the top to the one carried out by the DE scheme. After every generation, there is a certain (small) probability η that a pivot individual x_p, i.e. the current best point in the population, is selected to undergo the additional move. For a given number $Iter$ of iterations, the selected point is perturbed along the n axes according to the S operator logic, i.e. step forward and in case of failure backward for each design variable. At the beginning of the optimisation the exploratory radius δ, used to perform the deterministic search, is set to 40% of the search space size (basically it covers the entire search space), so acting more like a deterministic GS rather than a LS. As the optimisation process goes ahead , δ is subject to a shrinking procedure, see Algorithm 22 for implementation details, and the search gets limited in a smaller and smaller neighbourhood of the current pivot solution. After every execution of the extra move, the current value reached by $\boldsymbol{\delta}$ is stored in memory, as it happens in the LS Chain method, and then reloaded and used during the next step. It can be noticed that μDEA does not implement a proper LS Chain strategy, since the pivot individual can change and S is never reinitialised. In this way, during the optimisation process the extra move along the axis becomes more exploitative, so avoiding undesired stagnation, when the pivot solution is more likely to be close to optimal position. It must be noted that if such a small population size could lead to premature convergence in GAs, it has been conversely observed that the lack of a proper section strategy in DE keeps a certain degree of diversity also in μ-populations, see (Parsopoulos 2009). Since stagnation can anyway occur regardless of the population diversity level, the LS is mainly used for overcoming this problem. Moreover, even though a big population size has proven to provide a number of desirable features (Prügel-Bennett 2010), on the other hand a small one is preferable in the presence of memory limitations and so μDEs have been widely used in engineering applications (Rahnamayan & Tizhoosh 2008). for this reason, micro-population based algorithms are still being used, and is then worth improving their performances with light LS as in this case.

Algorithm 22 Micro-Differential Evolution with extra moves along the Axes

```
μPop ← randomSampling(M, n, 𝒟)
δ ← α(x^U − x^L)                                              ▷ α = 0.4
while Condition on budget do
    for j = 1 : M do
        select x_r, x_s, and x_t from μPop
        x_off ← x_t + F(x_r − x_s)                            ▷ Equation 3.14
        x_off ← XoverExp(x_j, x_off)                          ▷ Algorithm 19
        if f(x_off) ≤ f(x_j) then
            x_j ← x_off
        end if
    end for
    if 𝒰(0, 1) < η then
        extract the pivot individual x_p from μPop
        x_s ← x_p
        for k = 1 : Iter do
            improved ← false
            for i = 1 : n do
                x_s[i] ← x_p[i] − δ[i]
                if f(x_s) ≤ f(x_p) then
                    x_p ← x_s
                    improved ← true
                else
                    x_s[i] = x_p[i] + δ[i]/2
                    if f(x_s) ≤ f(x_p) then
                        x_p ← x_s
                        improved ← true
                    end if
                end if
                i ← i + 1
            end for
            if improved == false then
                δ ← δ/2
            end if
            k ← k + 1
        end for
    end if
end while
Output Best Individual
```

Table 4.1. Memetic Algorithms overview.

Algorithm	Population-based Meta-heuristic	Local Search Routines	Comments
MA-LSCH-CMA	*Steady-State* GA	CMA-ES	*-Lamarckian MA employing the LS Chain method. -Evolves several covariance matrices having* $m \sim \mathcal{O}\left(n^2\right)$.
MA-LSCH-SSW	*Steady-State* GA	*Subgrouping Solis-Wets*	*-Lamarckian MA employing the LS Chain method. -Suitable for Large Scale Optimisation.*
μDEA	*Micro*-DE	*S*	*-Simple and light Lamarckian MA. -LS never restarted. -Small memory footprint.* ✓
EPSDE-LS	EPSDE	*-Simplex -Powell -Rosenbrock*	*-Lamarckian MA applying multiple LS operators forming a pool for a given computational budget. -Operators are reinitialised at any application.*

Numerical results in (Caraffini, Neri & Poikolainen 2013) have shown the beneficial effect of S in μDEA, that has proven to improve upon ordinary μDEs and be competitive against powerful population-based state-of-the-art DEs, such as SADE, MDE-pBX and JADE. It is interesting to highlight that thanks to the extra move along the axes, a significant boost on the performance has been registered in particular on large scale problems, i.e. functions in 1000 dimensions from the suites Congress on Evolutionary Computation (CEC) 2008 and 2010. An explanation to this phenomenon will be given in Chapter 8.

Also multiple LS logics can be integrated in an MA. For instance, in (Iacca, Neri, Caraffini, & Suganthan 2014) another DE-based memetic optimiser is proposed by extending the EPSDE algorithm, described in Chapter 3.3, with an additional pool $\mathbf{P_{LS}}$ of diverse local search routines: Nelder-Mead simplex method (Algorithm 3, Chapter 2.2.2), Powell's conjugate direction method (Algorith 7, Chapter 2.2.6) and Rosenbrock's algorithm (Algorithm 2.2.5, Chapter 2.2.5). The main motivation behind this choice is that multiple and diverse LS help a general purpose algorithm deal with specific features such as ill-conditioning and separability, see (Caraffini, Neri, Iacca & Mol 2013) and (Iacca, Neri, Mininno, Ong & Lim 2012). Nelder-Mead contains, to some extent, global search features and thus has the potential of jumping outside a basin of attraction while Powell's and Rosenbrock's algorithms deterministically exploit the starting solution towards the closest optimum. Although the last two methods share a similar search, they differ since Rosenbrock performs a single diagonal move while Powell detects n conjugate direction where the fitness is optimised by means of a line-search method. With reference to Algorithm 23 it can be seen that after every F_{LS} generations of a standard EPSDE, one among the three LS methods is randomly selected from $\mathbf{P_{LS}}$ and applied to the best current individual in the population for B_{LS} fitness evaluations (in case of simplex algorithm being selected the best solution in the population takes the place of the first point of a randomly generated polytope). The proposed pseudo-code refers to a general scheme with desired mutations, parameters and local searchers, since this framework can be indeed easily extended with novel perturbation logic in order to fulfil the given necessities. The resulting algorithm, Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search (EPSDE-LS), improves upon its early version, i.e. EPSDE described in Chapter 3.3, and also outperforms state-of-the-art algorithms such as MDE-pBX, CCPSO2 and CMA-ES, see (Iacca, Neri, Caraffini, & Suganthan 2014).

Table 4.1 summarises the presented examples. It can be noticed that despite the common pattern, MAs can be quite different and display either extremely heavy structures, as in MA-LSCh-CMA, or light as for μDEA,

Algorithm 23 Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search

initialize a pool of mutation strategies $\mathbf{P_{mut}}$ *and crossover strategies* $\mathbf{P_{cross}}$
initialize a pool of scale factors $\mathbf{P_F}$ *and crossover probabilities* $\mathbf{P_{CR}}$
generate N_p *individuals of the initial population pseudo-randomly*
for $i = 1 : N_p$ **do**
  assign to $\mathbf{x_i}$ *random strategies/parameters from* $\{\mathbf{P_{mut}}, \mathbf{P_{cross}}, \mathbf{P_F}, \mathbf{P_{CR}}\}$
  compute $f(\mathbf{x_i})$
end for
$g = 1$
while *budget condition* **do**
  for $i = 1 : N_p$ **do**
    generate $\mathbf{x_m}$ *through mutation strategy/parameter associated to* $\mathbf{x_i}$
    generate $\mathbf{x_{off}}$ *through crossover strategy/parameter associated to* $\mathbf{x_i}$
    if $f(x_{off}) \leqslant f(x_i)$ **then**
      save index i for replacing $\mathbf{x_i} \leftarrow \mathbf{x_{off}}$ *(including mutation, crossover strategies as well as parameters)*
    else
      assign to $\mathbf{x_i}$ *new random strategies/parameters from the pools*
    end if
  end for
  perform replacements
  $g = g + 1$
  if $(g \bmod F_{LS}) = 0$ **then**
    select a local search algorithm from the pool $\mathbf{P_{LS}}$
    apply it to $\mathbf{x_{best}}$*, until the budget condition* B_{LS}
  end if
end while

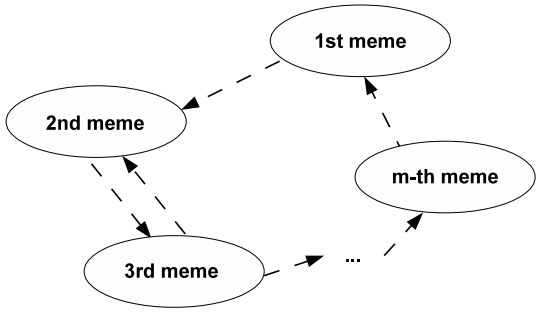


Figure. 4.1. MC cloud of generic operators.

employ multiple search as in EPSDE-LS or just minimal number of operators such as GA and Subgrouping Solis-Wets in MA-LSCh-SSW. All these algorithms can be also referred to in a more generic way as MC optimisers. Obviously the other way around would not be correct. MC has a very broad sense, including hybrid schemes, or multi-operator structures. To some extent, every single algorithm can be seen as an MC approach, as long as possible to disassemble and isolate its very basic operators, e.g. cross-over, perturbation, replacement, variable decomposition etc., and extract their coordination logic. So, this definition is very global and can be used to address all those techniques born during the last decades under different names, in a fairly young and still germinal discipline, which is becoming wider, more structured and consolidated. Since MC does not fix/specify any structure before an algorithmic analysis of the problem is performed, an evolutionary framework is not mandatory, as an efficient algorithm (for a given problem) can be based on multiple subsequent local searches (Molina, Lozano, García-Martínez & Herrera 2010*b*) or even by operators perturbing a single solution, see next section, thus not limiting the designer with standard choices. Figure 4.1 shows the most general scheme, consisting of m operators which potentially can have diverse structure and complexity.

For instance, the Multi-Strategy Coevolving Aging Particle (MS-CAP) algorithm (Iacca, Caraffini & Neri 2014) is an MC approach not employing a direct LS routine. Its structure can be seen as a simple two operators state machine, sharing and working on the same set of solutions, in this case addressed as a swarm of M "aging" particles. When the first operator stops improving, the second comes into help providing an alternative search move. The two operators complement each other, and can be seen as two separate algorithms, one deriving from the PSO family (Algorithm 25) and the second from DE (Algorithm 26). The main procedure is shown in Algorithm 24: the entire swarm is initialised with the same initial solution $\mathbf{x_{init}}$, each j-th particle $\mathbf{x_j}$ is associated to a lifetime $\mathbf{life}\,[j] = life_j$ (initially zero), and a velocity vector ($\mathbf{v_j} \in \mathbb{R}^n$). The main generational loop consists in alternating the two strategies in order to co-evolve the particles by dispersing and attracting them towards $\mathbf{x_{best}}$. The first logic starts as default operator, and performs random perturbation on each design variable, biased toward the best solution (see Line 7 and 6 in Algorithm 25), before evaluating the fitness value. The latter is activated when the first one fails at improving upon the best solution ($update$ condition) by applying multiple DE-like mutations, i.e. diagonal move though liner combination of particles, and cross-over strategies. In particular DE/rand/1 (Equation 3.14) and DE/rad/2 (Equation 3.16) provides with an exploratory search through $\mathcal{D}$, while DE/rand-to-best/2 (Equation 3.21) and cur-to-best/1 Equation 3.18) perform a biased move heading toward $\mathbf{x_{best}}$. As can be seen from Line 6 in Algorithm 25, unlike in classic PSO, the attraction

Algorithm 24 Multy-Strategy Coevolving Aging Particle

$\mathbf{x_{init}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
$n_{eval} \leftarrow 1$ ▷ Mmust be updated after evaluating the fitness value of every new generated point
$\mathbf{life} \leftarrow \emptyset$
$best \leftarrow 1$ ▷ Index of the best particle
for $j = 1 : M$ **do**
  $\mathbf{x_j} \leftarrow \mathbf{x_{init}}$
  for $i = 1 : n$ **do**
    $\mathbf{v_j}\,[i] \leftarrow \mathcal{U}\,(-1/2, 1/2) \cdot \left(\mathbf{x^U}\,[i] - \mathbf{x^L}\,[i]\right)$
    $i \leftarrow i + 1$
  end for
  $\mathbf{Swarm} \leftarrow \langle \mathbf{x_j}, \mathbf{life}\,[j]\,, \mathbf{v_j} \rangle$
  $j \leftarrow j + 1$
end for
while $n_{eval} \leq max_{eval}$ **do** ▷ max_{eval}: maximum computational budget allowed
  $update \leftarrow$CAP($\mathbf{Swarm}, \mathbf{x_{best}}, n_{eval}$) ▷ Algorithms 25
  if not $update$ **then**
    MSMR($\mathbf{Swarm}, \mathbf{x_{best}}, n_{eval}$) ▷ Algorithm 26
  end if
end while
Output $Best\,Individual$ $\mathbf{x_{best}}$

force toward the global best particle is not constant but progressively increases during the optimisation process. Thus, at the beginning the attraction is weak, i.e. large exploration pressure, whereas in later stages it gets stronger, i.e. local exploitation. Moreover, an "aging" logic considers the particle's lifetime to be set to zero after every improvement, or increased by one conversely. An exponential $decay = e^{-life_i}$ is then evaluated and, if smaller than a given threshold ε, the particle is replaced with another particle randomly chosen from the swarm, its lifetime is set to zero, and its velocity is reinitialised. If the decay is still larger ε, the perturbed particle $\mathbf{x_j}$ is instead reset to the old value $\mathbf{x_{old}}$. In this case, if $mod(life_i, 2) = 0$ holds true (i.e. the lifetime is even), the velocity is shrunk via the $decay$ factor in the opposite direction ($\mathbf{v_j} \leftarrow \mathbf{v_j}\,(-decay)$), otherwise (i.e. the lifetime is odd), velocity's magnitude is retained but its direction is changed ($\mathbf{v_j} \leftarrow -\mathbf{v_j}$). The aging mechanism plays an important role as it allows a natural refreshment of the particles, so avoiding undesired

stagnation, and it has been already successfully applied in PSO, see (Chen, Zhang, Lin, Chen, Zhan, Chung, Li & Shi 2013), for replacing unpromising solutions. In this case, it aims at refreshing the entire swarm while it is perturbed by two concurrent sets of perturbation rules, by acting like a sort of restart mechanism. Whenever

Algorithm 25 Coevolving Aging Particle

procedure CAP($\mathbf{Swarm}, best, n_{eval}$) ▷ $\mathbf{Swarm}$ contains particles, life-times and velocity vectors
  $update \leftarrow false$
  for $j = 1 : M$ **do**
    $\mathbf{x_j^{old}} \leftarrow \mathbf{x_j}$
    for $i = 1 : n$ **do**
      $\mathbf{v_j}[i] \leftarrow \mathbf{v_j}[i] + \mathcal{U}(0,1) \cdot \frac{n_{eval}}{max_{eval}} \cdot (\mathbf{x_{best}}[i] - \mathbf{x_j}[i])$ ▷ Line 6: velocity update
      $\mathbf{x_j}[i] \leftarrow \mathbf{x_j}[i] + \mathbf{v_j}[i]$ ▷ Line 7: position update
      $i \leftarrow i + 1$
    end for
    $n_{eval} \leftarrow n_{eval} + 1$
    if $f(\mathbf{x_j}) < f(\mathbf{x_{best}})$ **then**
      $update \leftarrow true$
      $best \leftarrow j$
    end if
    if $f(\mathbf{x_j}) < f\left(\mathbf{x_j^{old}}\right)$ **then**
      $life_i \leftarrow 0$
    else
      $life_i \leftarrow life_i + 1$
      $decay \leftarrow e^{-life_i}$
      if $decay < \varepsilon$ **then** ▷ $\varepsilon = 10^{-6}$ in (Iacca, Caraffini & Neri 2014) $\Rightarrow$ a maximum lifetime equal to 14
        $life_j \leftarrow 0$
        $r \leftarrow \mathcal{I}(0, M)\ ,\ r \neq j$
        $\mathbf{x_j} \leftarrow \mathbf{x_r}$ ▷ Reset to a random particle
        for $i = 1 : n$ **do**
          $\mathbf{v_j}[i] \leftarrow \mathcal{U}(-1/2, 1/2) \cdot (\mathbf{x^U}[i] - \mathbf{x^L}[i])$
          $i \leftarrow i + 1$
        end for
      else
        $\mathbf{x_j} \leftarrow \mathbf{x_j^{old}}$
        if $mod(life_j, 2) == 0$ **then**
          $\mathbf{v_j} \leftarrow \mathbf{v_j}(-decay)$
        else
          $\mathbf{v_j} \leftarrow -\mathbf{v_j}$
        end if
      end if
    end if
    $j \leftarrow j + 1$
  end for
  Output $update$
end procedure

the Coevolving Aging Particles fails at improving upon $\mathbf{x_{best}}$, Algorithm 26 is executed. As in EPSDE, it can execute four (previously mentioned) different mutation schemes, combined with two possible kind of cross-over: exponential (Algorithm 19) and binary (Algorithm 18). As for scale factor and cross-over rate for each j-th particle are drawn from random uniform distribution in $[0.1, 1)$ and $[0, 1)$ respectively. Also mutation and cross-over are randomly selected from two respective pools ($P_{mut} = \{mut_1, mut_1, mut_3, mut_4\}$ and $P_{xover} = \{xover_1, xover_2\}$). An offspring x_{xover} can thus be generated, compared with its parent x_i and, in case of improvement, inserted in the swarm according to the DE one-to-one spawning logic. After L repetitions of this sequence of operations, the particles which were updated are assigned a new velocity and lifetime. Beyond outstanding results on test-bed problems, also available on line[1], MS-CAP has been successfully used for training neural networks with multiple configurations (Iacca, Caraffini & Neri 2014).

[1] http://sites.google.com/site/facaraff/home/Downloads/MS-CAP_Detailed_Results.pdf

Algorithm 26 Multi-Strategy Mutation and Recombination

procedure MSMR($\mathbf{Swarm}, best, n_{eval}$)
  for $i = 1 : M$ **do**
    $changed_i \leftarrow false$
  end for
  for $l = 1 : L$ **do**
    for $j = 1 : M$ **do**
      $\mathbf{x}^{\mathbf{old}} \leftarrow \mathbf{x_j}$
      $j \leftarrow j + 1$
    end for
    for $j = 1 : M$ **do**
      $F_j = \mathcal{U}(0.1, 1)$
      $CR_j = \mathcal{U}(0, 1)$
      $m \leftarrow \mathcal{I}(1, 4)$
      $x \leftarrow \mathcal{I}(1, 2)$
      generate $\mathbf{x_{offspring}}$ *by applying* $mut_m + xover_x$ *to* $\mathbf{x_j}$ *with* F_j *and* CR_j
      $n_{eval} \leftarrow n_{eval} + 1$
      if $f(\mathbf{x_{offspring}}) < f\left(\mathbf{x_j^{old}}\right)$ **then**
        $\mathbf{x_j} \leftarrow \mathbf{x_{offspring}}$
        $changed_i \leftarrow true$
      end if
      $j \leftarrow j + 1$
    end for
    update best
    $l \leftarrow l + 1$
  end for
  for $j = 1 : M$ **do**
    if $changed_j$ **then**
      for $i = 1 : n$ **do**
        $\mathbf{v_j}[i] \leftarrow \mathcal{U}(-1/2, 1/2) \cdot \left(\mathbf{x^U}[i] - \mathbf{x^L}[i]\right)$
      end for
    end if
  end for
end procedure

4.1 The rationale behind MC

As previously discussed in Chapter 3, the search for the best optimiser has been stopped by the publication of the NFL theorem (Wolpert & Macready 1997), see Appendix B, which has had a significant impact on the CIO scientific community. Since then, it was clear that there was no longer a reason to argue which algorithm is universally better or worse, because the highest performance on a specific problem is given by the optimiser entirely designed around it. This concept is graphically represented in Figure 4.2: the sharp peak reaches the highest performance possible, and pays off its specialisation in dealing with a particular problem being unable to optimise any other different scenario. In this light, MAs/MC find their motivation, since allowing the designer to bend the flat line (general-purpose algorithm) towards a peak of interest. By adding a specific LS to the general optimisation framework, is it possible to "train" the algorithm so making it able to focus on a specific problem. Usually, see for example numerical results for μDEA in the previous section, once enhanced with LS routines the resulting algorithm globally improves upon their predecessor version (there is inevitably a loss of performance over some problems, see the ends of the thick grey line in Figure 4.2). Unfortunately, all the information required to designed a problem tailored algorithm is not always given. A more realistic scenario is the one where only a limited piece of information is available. Where it is not possible to design a specific solver, it is still possible to decompose the problem in multiple sub-problems, falling into well-known classes not depending on the nature of the application, but only on the fitness function modelling the optimisation

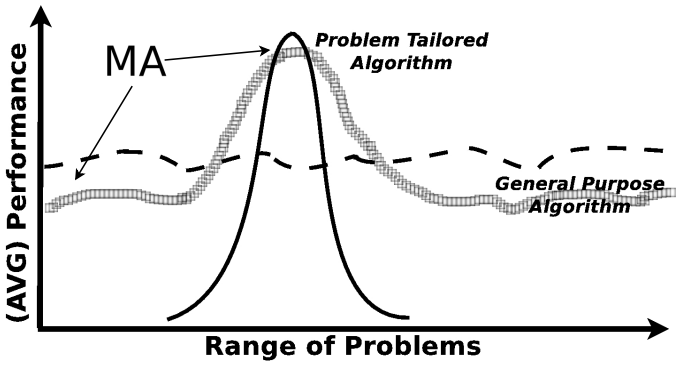


Figure. 4.2. Average performance for general-purpose meta-heuristics enriched with knowledge

problem. MA is the tool which gives us a way of handling such classes, e.g. is the problem separable?, is the fitness function ill-conditioned?, is it Multi-modal?, also dimensionality plays a crucial role (see Chapter 8), by adding specialised operators. It is known that some LS routines can better refine certain functions than others, e.g. S over separable functions or CMA-ES on ill-conditioned problems. Moreover, in many real-world problems, the use of a tailored algorithm for each problem could be extremely impractical as it would likely require the constant action of an optimisation expert to adjust the design and parameters to the industrial needs. In addition, the implicit non-stationary nature of optimisation processes imposes that there is no unique most suitable algorithm and parameter setting during the run-time. In order to tackle these issues, modern optimisation make use of multiple search operators that are coordinated with the aim of promptly reacting to the necessities of the optimisation processes and thus successfully addressing reasonable ranges of problem features. For this reason, MC has emerged and attracted a growing interest during the last years. It is so important to add operators so maintaining a diverse set of different moves (perturbation logics), and be able to address multiple scenarios. If adding many LS techniques can seem an easy way for handling all the problems, it must be also noted that this procedure could potentially lead to uselessly complex structures, difficult to coordinate and suffering of the opposite phenomen, i.e. performance decree. It must in fact be remembered that when applying optimisers to real-word problems, a prefixed (and usually restrictive) computational budget is allotted for the entire optimisation process, and the algorithm must perform in the best way without exceeding it: too complex would not work appropriately, and the design procedure has to be carried out carefully (see next section Chapter 4.2).

As a final consideration, it can be noted that the philosophy behind MC contains a visionary potential for the future in optimisation, see (Ong, Lim & Chen 2009) and (Ong et al. 2010). More specifically, if the design of an optimisation algorithm consists of selecting its suitable operators from a pool, as well as their interaction relationship, i.e. the links between the operators, this bottom-up design allows a straightforward implementation within a computational device, without relying on human experience, see Chapter 7.2. The idea of having a self-designing algorithm is not new in MC, for example (Meuth et al. 2009), but it has not been formalised in technical terms yet. The main aim of this thesis is to present a prototype framework for "automatic design" of optimisation algorithms. The proposed scheme, fully embraces the MC philosophy,

making use of a pool of operators which are selected only after a preliminary analysis of the fitness landscape, in terms of its fundamental features (such as separability), and tuned accordingly. In the following sections all the steps leading to the formulation of the proposed method are reported.

4.2 Ockham's Razor in MC: simple VS complex structures

A great deal of research has recently gone into the direction of MC, mostly combining and improving existing optimisers. The aforementioned algorithms are only a few examples of a plethora of designed variants. It is worth noting that after hybridisation, there is usually an increase on the performance of the algorithm accompanied by a growth of the complexity of the structure, that makes the application of the optimiser on real-world problems often infeasible, due to limited hardware resources, time restrictions and computational budget availability. EPSDE-LS is a good example of complex structure employing a high numbers of operators. MS-CAP as well relies on two operators which are *de facto* two stand-alone algorithms. By using the same logic even more complex structures have been designed in the past, having either many population based algorithm hybridised together or perturbation schemes grouped in a single framework, see (Peng et al. 2010) and (Montes de Oca et al. 2009) as significant examples but also Multicriteria Adaptive Differential Evolution (MADE) and Super-fit Multicriteria Adaptive Differential Evolution (SMADE). MA-LSCh-CMA is a different case of a performing algorithm, with no complex but rather heavy structure evolving a sub-population of covariance matrices. This feature makes it inapplicable over several problems, in particular LSOP. In a nutshell, if it is true that population based complex hybrid structures seem to be efficient, on the other hand simpler algorithms are preferable for a number of others reasons (i.e. allowing the designer to better understand which operator is actually working, inferior memory footprint and computational overhead etc.), and there is no theoretical proof they cannot compete with tangle optimisers. There is a number of successful simple implementations, e.g. jDE and μDEA, providing solutions as good as those of highly elaborate algorithms.

More formally, in (Iacca, Neri, Mininno, Ong & Lim 2012) the following research question has been raised and investigated:*"Is the algorithmic complexity in optimisation actually supported by the results? Is the complexity, in MC, an algorithmic feature which makes the algorithms any better?"*. This question, which will be further investigated and extended in the remainder of this thesis (Chapter 6), has been here addressed by designing a minimalistic algorithm, consisting of three basic memes. Each one of them has been specifically thought to deal with a particular phase of the optimisation process, so having a complete structure where every operator is as essential as the others, and free of redundancy. The proposed algorithm, 3SOME, relies on a simple bottom-up combination of the three entities based on a natural trial and error logic, by following the well-known *Ockhams Razor* principle[2]. Though the Keep it Simple, Stupid (KISS) principle is very common in Computer Science, the authors have re-adapted the Ockhams Razor concepts to shift the attention to another level. In fact, in CIO algorithms are sometimes inevitably complex due to the particular task to address or to the underlying idea and mathematical concept they are based on, but the design should be done so minimising

[2]"Pluralitas non est ponenda sine neccesitate" (plurality must not be considered without necessity), in other words there is no need to make things complex, when can be kept as simple as possible to reach the same goal.

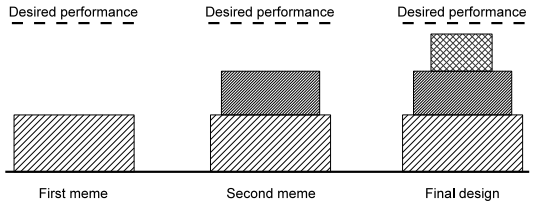


Figure. 4.3. Bottom-up design strategy.

the impact on their structure. In other words, unnecessary memes shouldn't be added if it makes the optimiser slower, heavier (memory-wise) and makes it perform redundant operations. An interesting consideration upon this principle is that, if well designed, there should be no need to add more components to the existing algorithm, but whether there is always a margin for improving by adding new features, what it should not be possible to do is to take operators off without performance loss. According to this reflection, the novel memetic structures proposed in this piece of research, and described in the next chapter, have been designed by invoking the Ockham's Razor principle for MC, making sure to have the least structure able to guarantee and reach a certain goal. In order to prove it, experiments have been carried out, so bringing evidence with numerical results on how simpler schemes would not work any better, see Chapter 7.1.1 and 7.2.5 for details.

The 3SOME algorithm is based on the idea that a satisfactory solution can be achieved by proceeding through 3 simple steps, focussing the search on progressively more exploitative levels. The algorithm is built from scratch, starting with a global search which ends up in an intermediate stage before undergoing local refinements. The operators are sequentially activated by means of a set of deterministic conditions, according to their exploratory/exploitative features: during the first stage "Long Distance Exploration" (Algorithm 28) no knowledge of the problem is known and a new random point is generated in $\mathcal{D}$ until an improvement upon the initial solution occurs. During the second stage, "Middle Distance Exploration" (Algorithm 29) the most promising basin of attraction is exploited in its neighbourhood until it is possible to improve upon it. As soon as the the second stage does no longer succeed at improving the passed solution, the resulting output is refined via the S operator (Algorithm 2.2.4) in the last stage: "short Distance Exploration". If S succeeds, the middle distance exploration is newly activated (the first stage is simply skipped) to further achieve improvements. Conversely, if the short distance exploration fails at improving the solution, the optimisation is started over with the long distance exploration. This logic, reported in Algorithm 27 is repeated until the computational budget expires. It can be noted that the three memes are not able to accurately optimise a given problem on their own, each one is specified at handling a given necessity and this bottom-up strategy, graphically explained in Figure 4.3, leads to the final goal. While S has been widely introduced in Chapter 2.2.4, the other two stages need further explanation, since both were designed ad-hoc for covering stage number 1 and 2 in (Iacca, Neri, Mininno, Ong & Lim 2012). Going into detail, during the Long distance exploration (L), a new trial solution $\mathbf{x_{trial}}$ is sampled within the entire decision space, and mated by means of the exponential crossover with the current elite solution $\mathbf{x_{elite}}$. As mentioned in the previous chapter, the elite solution is the one corresponding to the best fitness value found by the algorithm. In this case, $\mathbf{x_{elite}}$ is basically the only solution undergoing the optimisation process and, during the first stage, donates $\alpha\%$ of its design variables to each newly generated

Algorithm 27 Three Stage Optimal Memetic Exploration

$\mathbf{x_{elite}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
while *budget condition* **do**
  /**STAGE 1**/
  $\mathbf{x_{elite}} \leftarrow$L($\mathbf{x_{elite}}$) ▷ algorithm 28
  /**STAGE 2**/
  $\mathbf{x_{elite}} \leftarrow$M($\mathbf{x_{elite}}$) ▷ algorithm 29
  /**STAGE 3**/
  $\mathbf{x_{elite}} \leftarrow$S($\mathbf{x_{elite}}$) ▷ algorithm 5
  if $\mathbf{x_{elite}}$ *has improved* **then**
    skip next STAGE 1
  end if
end while
Output $\mathbf{x_{elite}}$

solution. In this way, on the top of performing the search within the entire decision space and thus attempting to detect unexplored promising basins of attraction, this operator also promotes retention of a small section of promising design variables. This kind of inheritance of some genes appears to be extremely beneficial in terms of performance with respect to a stochastic blind search. If the trial solution outperforms the elite, a replacement occurs. A replacement is performed also if the newly generated solution has the same performance of the elite. This is to promote the exploration of unexplored area and to prevent plateaus from jeopardising the search. In the middle distance exploration stage, a hyper-cube whose edge has a side width equal to δ is

Algorithm 28 Long Distance Exploration

procedure L($\mathbf{x_l}$)
  while $\mathbf{x_l}$ *has not improved* **do**
    $\mathbf{x_{trial}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
    $\mathbf{x_{trial}} \leftarrow$XOVEREXP($\mathbf{x_{trial}}, \mathbf{x_l}$) ▷ Algorithm 19, CR calculated as in Formula 3.22 (inheritance factor α)
    if $f\,(\mathbf{x_{trial}}) \leq f\,(\mathbf{x_l})$ **then**
      $\mathbf{x_l} \leftarrow \mathbf{x_{trial}}$
    end if
  end while
end procedure
Output $\mathbf{x_l}$

constructed around the elite solution $\mathbf{x_{elite}}$. Within this region, $k \times n$ trial points are stochastically generated by randomly perturbing the elite along a limited number of dimensions, thus making a randomised exploitation of the current elite solution. In other words, this stage attempts to focus the search around promising solutions in order to determine whether the current elite deserves further computational budget or other unexplored areas of the decision space must be explored. If the elite is outperformed, it is replaced. A replacement occurs also if one of the newly generated solutions has the same performance of the elite, in order to prevent the search getting trapped in some plateaus of the decision space. At the end of this stage, if the elite has been updated a new hypercube is constructed around the new elite and this mechanism is repeated. On the other hand, if the middle distance exploration does not lead to an improvement, an alternative search logic is applied, that is the deterministic logic of the short distance exploration. This algorithm has proven to be competitive against complex and powerful optimisers, such as JADE, MDE-pBX, CLPSO , MA-LSCh and cDE (see Table 7.5 and 6.4), and extremely versatile and suitable for implementation in embedded systems (Iacca, Caraffini & Neri 2013). As well as a low algorithmic computational overhead, see Figure 6.15, 3SOME requires a small amount of memory to be executed, since working only over two memory slots, one for the elite and one for trial solution. These features make it appropriate for both embedded and real-time applications.

Algorithm 29 Middle Distance Exploration

procedure M($\mathbf{x_m}$)
  do
    generate an hypercube $\mathcal{HC}$ centred in $\mathbf{x_m}$ with size width Δ
    for $j = 1 : k \cdot n$ **do**
      $\mathbf{x_{trial}} \leftarrow randomSampling\,(1, n, \mathcal{HC})$
      $\mathbf{x_{trial}} \leftarrow$XOVEREXP($\mathbf{x_{trial}}, \mathbf{x_m}$) ▷ Algorithm 19, CR calculated as in Formula 3.22 (but $\alpha^{'} = 1 - \alpha$)
      if $f\,(\mathbf{x_{trial}}) \leq f\,(\mathbf{x_m})$ **then**
        $\mathbf{x_m} \leftarrow \mathbf{x_{trial}}$
      end if
      $j \leftarrow j + 1$
    end for
  while $\mathbf{x_m}$ *improves*
end procedure
Output $\mathbf{x_m}$

Chapter 5

Technical research methods: experimental set-up and data presentation

In order to address and answer to the research questions motivating this piece of work, extensive experiments have been performed. Statistic tests have also been reported in detailed tables, accompanied with comments and interpretations. Real-world applications have been omitted for avoiding an excessive length of this thesis[1], but results from a significantly wide range of benchmark problems are displayed to support, with numerical evidence, every step taken and achievement claimed. The research methodologies adopted are summarised here before going into the "experimentation" part of this work. The experimental set-up used for addressing the research questions is also described on a separate section (Chapter 5.1). A clarification on the methodology used for finding the suggested set of parameters of the proposed algorithms is given in Chapter 5.2. For the sake of clarity, a list reporting the complete parameters setting for all the algorithms ran in this piece of research can be found in Appendix D, in order to allow the reproduction of the presented conclusions.

Intense experiments over a large set of problems have been performed in order to be able to know the general behaviour of the proposed optimisers, and also find out both potentialities and limitations. This approach guarantees a good understanding of what the real achievements are, and allows to spot flaws and undesired scenarios for a given class of problems. Anyway, ad-hoc experimentations have also been set-up in order to prove specific points (see for examples the Lennard-Jones Potential (LJP) application in Chapter 7.2.6 and 8.4.1). Moreover, in order to perform a fair comparison with the already existing optimisers, a set of 14 algorithms, representing the actual state-of-the-art, have been considered to compare with. Comparisons against the proposed algorithms and some of their variants, or against their predecessor algorithms, have also been included in order to show that the proposed version is the one that actually presents better performances. Numerical data have been obtained by using well-known benchmark suites in the filed, such as those realised for the Congress on Evolutionary Computation, but also the Black-Box Optimisation Benchmark (BBOB) 2010 suite. The set of functions used in this work is vary, and has been kept constant apart from small variations due to new annual releases of newer versions for the CEC benchmarks, and due to the requests made by the reviewers during the peer-review process. It must be remarked that the numerical data presented in this

[1] One can read the original papers listed in Appendix C for details regarding the real-world engineering applications.

thesis, apart from those displayed in the last chapter (Chapter 8) have been published in the papers listed in Appendix C. Finally, the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimisation (SISC) 2010 test suite has been also considered in Chapter 8. Obviously, this choice was justified by the fact that the latter benchmark is fully scalable, and extremely suitable for studying the same problem in increasing dimensionality values. For the same reason, the test-bed problem number 2 (LJP) from CEC2011 (Swagatam & Suganthan 2010) has been used in Chapter 7 and 8.

More formally, the experimental set-up used in this thesis consist of:

- The CEC2005 benchmark (25 test problems) described in (Suganthan, Hansen, Liang, Deb, Chen, Auger & Tiwari 2005) in 30 dimensions;
- The CEC2008 benchmark (7 test problems) described in (Tang, Yao, Suganthan, MacNish, Chen, Chen & Yang 2007) in 1000 dimensions
- The BBOB2010 benchmark (24 test problems) described in (Hansen, Auger, Finck, Ros et al. 2010) in 10, 20, 40 and 100 dimensions;
- The SISC2010 benchmark (19 test problems) described in (Lozano, Molina & Herrera 2011) in 10, 30, 50, 100, 500 and 1000 dimensions;
- The CEC2010 benchmark (20 test problems) described in (Hansen, Auger, Finck & Ros 2010) in 1000 dimensions ;
- The CEC2013 benchmark (28 test problems) described in (Liang, Qu, Suganthan & Hernndez-Daz 2013) in 10, 30 and 50 dimensions.

Thus, a diverse set of test-bed functions, having different and multiple features, have been considered in this work. For each algorithm considered 100 runs have been performed, each one continued for $5000 \times n$ fitness evaluations.

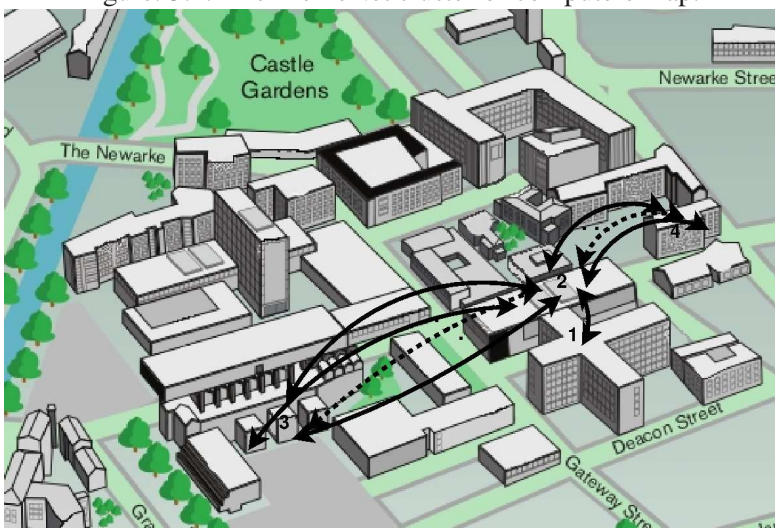
All the results were reported in tables, mainly grouped in Appendix E, showing data in terms of average fitness or average fitness error (with respect to the target fitness) and its standard deviation. Furthermore, in order to perform a fair comparison, the statistical significance of the results has been strengthened thanks to the Wilcoxon Rank-Sum test, see Appendix A.1, displayed with the following notation: the symbol "=" indicates a statistically equivalent performance of the reference algorithm when compared, in terms of Wilcoxon Rank-Sum test, with the algorithm in the column label. Conversely, since the two algorithms have a different statistic behaviour, a "+" is used to point out an average better performance of the reference algorithm with respect to the comparison algorithm, while a "-" if the comparison algorithm average performance has not been outperformed. A further statistic test, the Holm-Bonferroni procedure described in Appendix A.2, was reported on a separate table. The use of these two statistic tests, together with other information such as median, mean value and standard deviation of multiple runs, represents the most modern and more appropriate way of comparing optimisation algorithms, see (García, Molina, Lozano & Herrera 2009). Numerical results

were displayed in scientific notation and worked out with double precision. For the sake of readability, tables show an approximation for the mantissa limited to 3 decimal places, but comparisons take place before the approximation. In order to effectively display the highest performance for each problem and each algorithm, the best average fitness values are highlighted in boldface.

5.1 Hardware and software setup: MemeNet

Extensive testing over complete benchmarks, especially in high dimensionality values, requires particular software/hardware resources to be carried out in a reasonable amount of time. The large amount of numerical data produced during my PhD could have not been achieved without an appropriate "infrastructure" for testing optimisation algorithms. Moreover, also the algorithmic design involves a number of revisions and refinements, implying experiments over diverse fitness landscapes. In order to face this issues, huge experiments can be split up into sub-experiments, i.e. single run (single algorithm over a single problem), and executed in parallel if multiple computational cores are available. Since high performance multiprocessors systems are too expensive, a high performance computational cluster was set-up at the first stage of my PhD, by making use of existing computational resources at De Montfort Univeristy.

Figure. 5.1. The MemeNet cluster of computers map.



In details, all the required software, i.e. algorithms, benchmarks problems, and even the software for collecting data coming from the cluster, applying statistic tests and eventually displaying results as LaTeX tables, has been coded in Java and then integrated within the software platform Kimeme (Cyber Dyne Srl Home Page 2012). Thanks to this tool, a cluster of 182 computers all over the university campus has been "carved" from the resources already available for solving optimisation problems. Kimeme has a modular structure so that each component can be placed on a different machine, and via few settings they can easily establish a network communication. Every machine involved has to be kept up-to-date with a package containing algorithms and

problems, and has to be remotely monitored in order to avoid either a loss of computational cores available during heavy experiments or an excessive use of the computational power when they are used for didactic purposes. In fact, in order to optimise the usage of the available hardware resources, the same machines used by the students are at the same time exploited for research. The optimisation process runs in background, i.e. the students don't realise that the machine they are using is producing a little portion of data for a bigger experiment, and thanks to a set of bash scripts the number of employed computational cores can be reduced during laboratory activities. The impact of this processing on the activity of the students has been tested and estimated to be fairly low so as to not compromise teaching activities. In the same way, the machines forming the remote nodes of the cluster, named "MemeNet", can be remotely updated, and reinserted into the "optimisation network", e.g. some students may need to restart the machine so dropping down the available computational power. With reference to Figure 5.1, it can be seen that presently, MemeNet reaches out of 4 buildings of the campus. Experiments are launched from a working station in Gateway House (building number 1 in Figure 5.1), that is arranged to receive back, store and process a large amount of data. An xml file, containing for each algorithm parameters setting, problems to be optimised, and the number of repetitions and allotted computational budget, is first sent towards a central node located in Eric Wood (Building number 2) before being partitioned in sub-experiments, which are despatched to the remote nodes according their number of available cores. MemeNet is so configured as a star network, managed by a head-node handling the communication with the remote nodes, collecting and sending back to the working station partial results from every single core. Remote nodes are mostly located in different laboratories, i.e. different sub-networks of De Montfort Univeristy's network, in Queens and Clephan buildings (number 3 and 4 in Figure 5.1), but some of them also belong to a private network accessible only through the head-node, i.e. thus explaining why this specific machine has to be used as the central node. The head-node can offer 8 computational cores, and is able to access 32 more quad-core processors in the private network. The latter share a single file-system, together with the head-node, so this machine can actually be used as a single supercomputer running 136 computational cores. Machines in the Queens building have to be linked one by one, for a total of 21 remote nodes with 8 cores each (168 cores), as well as those in Clephan. Resource in the last building comes from multiple labs featuring 19 12-core machines, plus 97 8-core and 33 dual-core computers (682 computational cores). Thus, MemeNet provides a peak computational power of 986 cores, or, in other words, can potentially optimise 986 test-bed problems at the same time.

5.2 Algorithmic design and parameters tuning

The use of MemeNet had a major impact also during the algorithmic design. With the aforementioned hardware/software set-up was possible to test multiple variants of the same algorithm in a short amount of time. Tables containing statistic tests come automatically once the data are ready and the most promising variant cab be easily chosen while the other discarded, and then further refined with successive rounds. The algorithmic design takes usually place through a series of intermediate steps, where little modifications are maid and need to be tasted in order to understand which one is the most suitable and whether the designer is going toward

the right direction or not. Being able to perform these steps over many problems simultaneously, in a seriously short amount of time, is an advance that prevent from successive meaningless experimentations. In this study, all the problems from CEC2005 in 10 and 30 dimensions and BBOB2010 in 100 have been considered during the algorithmic design, since 100 runs over this two benchmark configuration only requires few minutes. Thus, was possible to empirically study the behaviour of prototypes on diverse problems, e.g. separable, mono-modal, etc., in different dimension values.

A similar strategy has been adopted also for the tuning of the parameters suggested for the optimiser introduced in this work. It is worth noting that the optimal tuning of parameters is a problematic procedure that has been investigated by researchers and it is still under study. The computational cluster for testing optimisation algorithms built during my doctoral studies represents a precious tool for addressing this problem. In effect, amongst the issues for performing the tuning, the first concern regards the time consuming aspect of testing multiple combinations of parameters, that are supposed to be optimised at the same time since a bed choice of the first one could affect the value of the last one, see (Eiben, Michalewicz, Schoenauer & Smith 2007). According to (Eiben et al. 2007), also neglecting the linkages among the parameters, i.e. tuning them separately one at time, would be too time consuming, and for this reason also on-the-fly tuning methods were proposed . Anyway, this issue can be easily overtaken thanks to the computational speed of MemeNet.

Furthermore, a more interesting observation is that the optimal set of parameters exists for a specific problem, while is impossible to find the set of parameters guaranteeing best results over all the possible problems (Nannen, Smit & Eiben 2008). This can be seen as a consequence of the NFL theorem for optimisation. So, while tuning the parameters on a single real-world problem could make sense, it would be worthless for tests over complete benchmark suites. Moreover, trying to change the parameters for each specific benchmark function, will turn into an unfair comparison against the comparison algorithms, which should be appropriately tuned as well. For this reason, in this thesis, the suggested set of parameters for the newly designed algorithms has been obtained by always using the same set of functions, i.e. CEC2005 in 10 and 30 and BBOB2010 in 100 dimensions, trying to find a good set of value that keep the algorithm quite versatile and general purpose. Ad-hoc tuning has to be made, obviously, in real world specific scenarios, see (Davis et al. 1991) and (Eiben, Hinterding & Michalewicz 1999), but in order to perform a fair comparison no particular tuning has been carried out on selected functions, and no cherry-picked problem has been used for testing. Conversely, a very diverse and large set of functions, up to 132 problems at time, have been optimised on each experiment, and most of them were not used for the parameters tuning and vice-versa. Finally, it must be said that parameters for comparison algorithm came from the suggestions made from the authors in the original publications.

Chapter 6

Again on Ockham's Razor for MC: further investigations

This chapter contains all the preliminary studies laying the foundations for the "Novel Memetic Structures" presented in the remainder of this thesis. Starting with a study on 3SOME's bottom-up structure, the importance of each operator's role within the algorithmic structure of a MC approach, as well as the importance of the coordination logic of its memes, are discussed and investigated. Novel simple, but competitive, algorithms are then proposed and tested on several benchmark problems in order to show their efficiency, also when compared against complex well-known algorithms. This investigation has been organised in three consecutive sections, each one addressing **IRQ I**, **IRQ II** and **IRQ III** respectively.

6.1 Studying and altering 3SOME's structure

As soon as the very first implementation of 3SOME was published, slightly different variants have followed, invoking the same principle and attempting to enhance the former framework, which indeed presents a general-purpose scheme that can be easily made more specific in order to improve upon a specific class of problems. As a first attempt, is worth mentioning the Meta-Lamarckian-3SOME (ML-3SOME), where the trial-and-error coordination for the 3 memes has been replaced with the well-known Meta-Lamarckian Learning method (Ong & Keane 2004), as described in (Neri et al. 2012). It must be stated that this variant does not modify any of the memes, but only affects their coordination by exchange the old deterministic logic with a probabilistic selection scheme, considering the performance history of each operator during its previous activations. The advantages of the adaptive coordination are evident for low dimensional problems, n up to 40 as shown in detail in Table E.1(a) and E.1(b) (Appendix E). Conversely, this logic is not able to improve upon the classic 3SOME on higher dimensions, apart from few cases as shown in Figure 6.1 and 6.2 for f_{10} and f_{15} from the suite CEC2010.

Conversely, in (Poikolainen et al. 2013) a novel operator for the second stage is introduced, comporting a

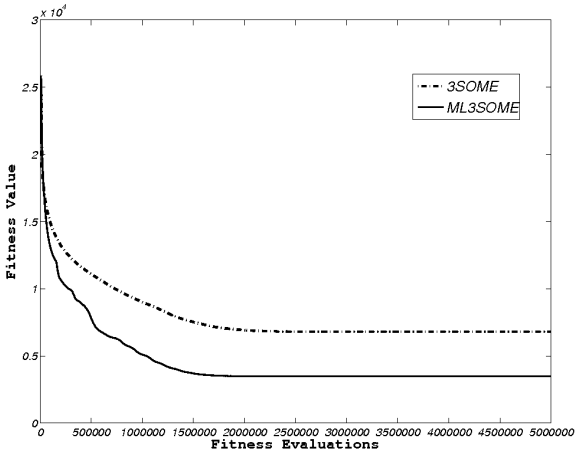


Figure. 6.1. Average fitness trend for ML-3SOME against 3SOME on f_{10} from CEC2010 in 1000 dimensions.

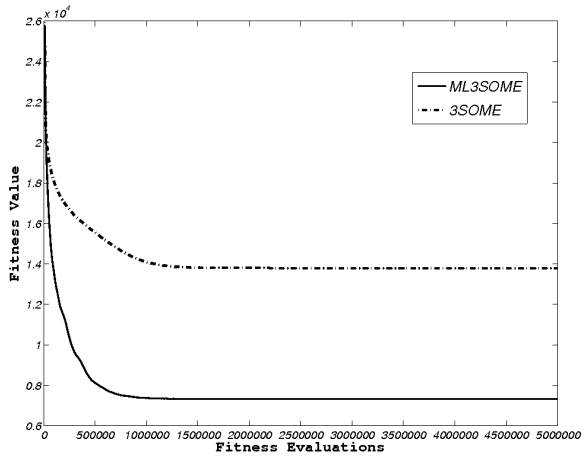


Figure. 6.2. Average fitness trend for ML-3SOME against 3SOME on f_{15} from CEC2010 in 1000 dimensions.

significant improvement on the performance over large scale problems, see Table 6.1. The proposed algorithm, Shrinking-3SOME (S-3SOME), keeps the same coordination logic of its predecessor 3SOME while replacing the middle distance operator with a more exploitative meme, see Algorithm 30. This exploration move attempts

Algorithm 30 Stochastic Short Distance Exploration

procedure SHRINKING($\mathbf{x_s}$)
  generate a hypercube $\mathcal{HC}$ *around* $\mathbf{x_s}$ *with a hyper-volume* 20% *of that of* $\mathcal{D}$
  while *the hyper-volume is bigger than* 0.0001% *of* $\mathcal{D}$ **do**
    for $j = 1 : n$ **do**
      $\mathbf{x_{trial}} \leftarrow randomSampling\,(1, n, \mathcal{HC})$
      if $f\,(\mathbf{x_{trial}}) \leq f\,(\mathbf{x_s})$ **then**
        $\mathbf{x_s} \leftarrow \mathbf{x_{trial}}$;
        centre the hypercube around $\mathbf{x_s}$
      end if
      $j \leftarrow j + 1$
    end for
    if *no elite update occurred* **then**
      halve the hyper-volume
    end if
  end while
end procedure
Output $\mathbf{x_s}$

to detect promising areas of the decision space by making use of a stochastic logic, in contrast with the deterministic search performed by S during the last stage. In a nutshell, when the long distance exploration detects a new promising solution, the stochastic short distance exploration generates a hypercube centred in the newly detected solution having a hyper-volume which is 20% of the decision space $\mathcal{D}$. This exploration samples a trial solution for n times (n being the dimensionality of the problem), attempting to outperform previous point. If an improvement occurs, the initial solution is updated and the hyper-volume is generated. If, after n comparisons, at least one replacement occurred, n new attempts are scheduled in the same hyper-volume, and comparisons performed accordingly. On the contrary, if all the n comparisons led to no improvements, the n new samplings are scheduled in a new hyper-volume, obtained by halving the current one. This shrinking mechanism is repeated until the hyper-volume is smaller than 0.0001% of the total hyper-volume. When this

Table 6.1. Average Fitness $\pm$ Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = S-3SOME) for S-3SOME against 3SOME on CEC2010 in 1000 dimensions.

	S3SOME	3SOME	
f_1	$\mathbf{6.03e-04 \pm 4.71e-04}$	$2.15e-02 \pm 6.88e-02$	+
f_2	$\mathbf{3.99e-02 \pm 9.13e-03}$	$1.36e+01 \pm 1.91e+01$	+
f_3	$\mathbf{7.56e-03 \pm 6.83e-04}$	$4.81e-01 \pm 4.94e-01$	+
f_4	$2.08e+13 \pm 6.90e+12$	$\mathbf{8.08e+12 \pm 3.04e+12}$	-
f_5	$\mathbf{4.27e+08 \pm 1.16e+08}$	$7.26e+08 \pm 1.38e+08$	+
f_6	$\mathbf{1.54e+07 \pm 5.39e+06}$	$1.98e+07 \pm 9.20e+04$	+
f_7	$1.07e+10 \pm 2.83e+09$	$\mathbf{1.48e+09 \pm 3.73e+08}$	-
f_8	$1.78e+09 \pm 2.68e+09$	$\mathbf{8.68e+08 \pm 2.62e+09}$	-
f_9	$\mathbf{3.54e+08 \pm 7.89e+07}$	$4.24e+08 \pm 6.35e+07$	+
f_{10}	$\mathbf{5.12e+03 \pm 2.63e+02}$	$6.75e+03 \pm 3.69e+02$	+
f_{11}	$\mathbf{1.97e+02 \pm 4.27e+00}$	$1.99e+02 \pm 5.28e-01$	+
f_{12}	$\mathbf{8.74e+04 \pm 2.12e+04}$	$1.57e+05 \pm 7.74e+04$	+
f_{13}	$5.61e+05 \pm 7.34e+05$	$\mathbf{1.48e+04 \pm 6.61e+03}$	-
f_{14}	$\mathbf{8.79e+07 \pm 2.66e+06}$	$1.12e+08 \pm 2.13e+07$	+
f_{15}	$\mathbf{1.33e+04 \pm 2.40e+03}$	$1.37e+04 \pm 6.36e+02$	=
f_{16}	$\mathbf{1.60e+02 \pm 1.14e+02}$	$3.81e+02 \pm 6.13e+01$	+
f_{17}	$\mathbf{6.31e+04 \pm 8.74e+03}$	$2.78e+05 \pm 2.26e+05$	+
f_{18}	$\mathbf{3.88e+03 \pm 4.66e+03}$	$2.27e+04 \pm 1.47e+04$	+
f_{19}	$1.16e+06 \pm 8.91e+04$	$\mathbf{1.44e+05 \pm 1.83e+04}$	-
f_{20}	$1.20e+03 \pm 1.93e+02$	$\mathbf{1.14e+03 \pm 1.51e+02}$	=

condition occurs, the current value of $\mathbf{x_s}$ is passed to the following operator (S) for further improvements. The difference between the original implementation in 3SOME is that edge length of the hypercube created is not fixed value but also shrinks over time. This modification allows scaling along the dimensionality of the problem and being more exploratory at the start and turning into a more exploitative search operator towards the end. Numerical results in (Poikolainen et al. 2013) show that S-3SOME is equivalent to its predecessor in low dimension value and significantly better for high dimensions, being able to overtake also powerful single solution and compact algorithms already in 100 dimensions, as well as more complex popular population-based algorithms, see Figure 6.3 and 6.4. Its strength in tackling LSOP in 1000 dimensions it is even more evident, see Figure 6.5 and 6.6, but also detailed results on the full suite CEC2010 reported in Table E.2 E.3.

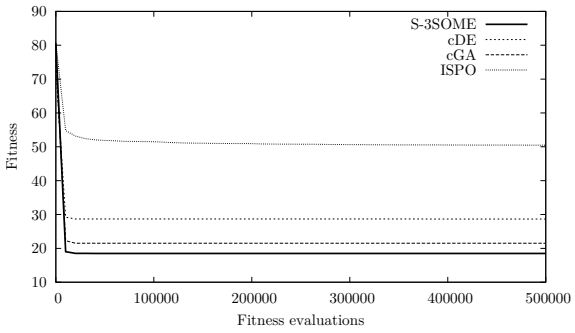


Figure. 6.3. Average fitness trend for S-3SOME against memory-saving algorithms on f_{16} from BBOB2010 in 100 dimensions.

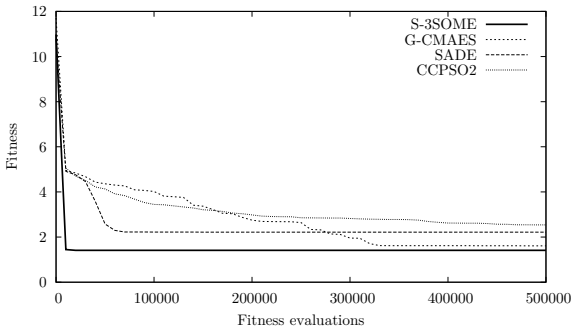


Figure. 6.4. Average fitness trend for S-3SOME against population-based algorithms on f_{23} from BBOB2010 in 100 dimensions.

The main drawback of the 3SOME algorithm is that the non-separability is no explicitly addressed and thus the algorithm displayed a not so good performance in some cases. A first effort in order to tackle this problem has been taken in (Poikolainen et al. 2012), where the exponential cross-over in Algorithm 29 has been replaced

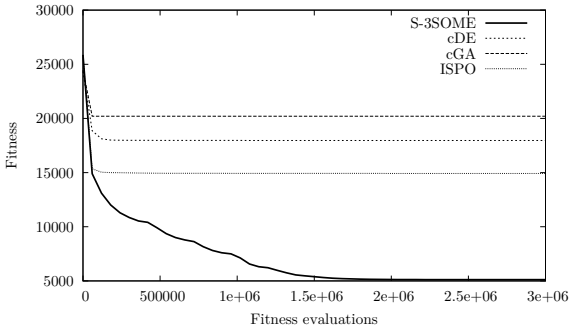


Figure. 6.5. Average fitness trend for S-3SOME against memory-saving algorithms on f_{10} from CEC2010 in 1000 dimensions.

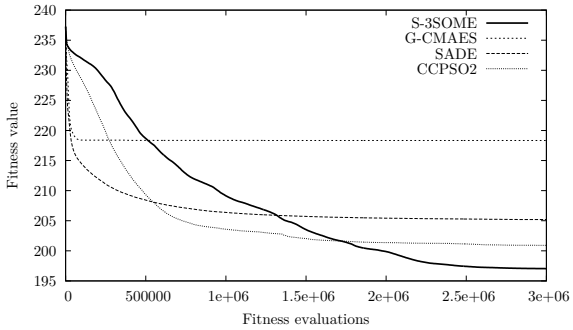


Figure. 6.6. Average fitness trend for S-3SOME against population-based algorithms on f_{11} from CEC2010 in 1000 dimensions.

with the rotational invariant DE/current-to-rand/1 perturbation scheme (Equation 3.19), thus the name Rotation Invariant-3SOME (RI-3SOME). As soon as the algorithm gets in to the second stage, the trial solution is not longer randomly generated but obtained via a linear combination of three points $\mathbf{x_r}$, $\mathbf{x_s}$, and $\mathbf{x_v}$, respectively randomly sampled within $\mathcal{D}$ and then combined according to the popular DE scheme. Since an improvement on the performance of the modified algorithm, on those non-separable benchmark functions affected by rotations, was noted, this concept has then been further investigated in (Caraffini, Iacca, Neri & Mininno 2012*b*).

In particular, three variants for handling non-separability, with increasing algorithmic complexity, were designed, tested and compared. A careful analysis of the 3SOME structure suggests that the long and the short distance operators are somehow algorithmically necessary to properly balance exploration and exploitation. Thus, the simpler idea in order to improve upon 3SOME is to slightly modify its original structure by replacing the middle search operator with an operator specifically tailored for handling non-separability. In this light, the first two modifications involve the second stage only. The first variant, extends what has already been done with RI-3SOME by mixing its logic with the shrinking operator (Algorithm 30) of S-3SOME. In this case the trial solution is also generated via DE/current-to-rand/1 but $\mathbf{x_r}$, $\mathbf{x_s}$, and $\mathbf{x_v}$ are sampled within an hypercube collapsing toward a promising basin of attraction, see Algorithm 31. This operator provides a LS diagonal move which, being stochastic still contains a certain degree of inheritance, i.e. the three randomly sampled points are linearly combined with the elite solution. The coordination logic for the operators in Rotation Invariant Shrinking-3SOME (RIS-3SOME) is graphically given by Figure 6.7, where the algorithm is described as a composition of states, each one corresponding to a single meme processing and returning the (hopefully improved) elite solution. The operator can be said to "succeed" if it is able to improve upon the incoming elite, otherwise it can be said to "fail". With reference to figure 6.7, the arrows represent the interaction amongst memes. The "S" and "F", represent success and failure, respectively, of the meme, while the condition on the search volume in the shrinking component is labelled explicitly. The rotational invariant perturbation in Equation 3.19 is still used on the second variant, but applied in a more classical way by replacing the original middle distance exploration with a micro-population Differential Evolution. The operating principle of this algorithm, named Micro-population Differential Evolution-3SOME (μDE-3SOME), is the following. Whenever the long (or short) distance exploration returns a new elite, a micro-population of $M = 5$ individuals is sampled within a hypercube centred around $\mathbf{x_{elite}}$, whose volume has been empirically set equal to 40% of the volume of the entire search space. The worst individual of the micro-population is then replaced with the current

Algorithm 31 Stochastic Diagonal Short Distance Exploration

procedure DIAG-SHRINKING($\mathbf{x_{scr}}$)
  generate a hypercube $\mathcal{HC}$ *around* $\mathbf{x_{scr}}$ *with a hyper-volume* 20% *of that of* $\mathcal{D}$
  while *the hyper-volume is bigger than* 0.0001% *of* $\mathcal{D}$ **do**
    for $j = 1 : n$ **do**
      $\mathbf{x_r} \leftarrow randomSampling\,(1, n, \mathcal{HC})$
      $\mathbf{x_v} \leftarrow randomSampling\,(1, n, \mathcal{HC})$
      $\mathbf{x_s} \leftarrow randomSampling\,(1, n, \mathcal{HC})$
      $\mathbf{x_{trial}} = \mathbf{x_{scr}} + K\,(\mathbf{x_v} - \mathbf{x_{scr}}) + F'\,(\mathbf{x_r} - \mathbf{x_s})$ ▷ Equation 3.19
      if $f\,(\mathbf{x_{trial}}) \leq f\,(\mathbf{x_{scr}})$ **then**
        $\mathbf{x_{scr}} \leftarrow \mathbf{x_{trial}}$;
        centre the hypercube around $\mathbf{x_s}$
      end if
      $j \leftarrow j + 1$
    end for
    if *no elite update occurred* **then**
      halve the hyper-volume
    end if
  end while
end procedure
Output $\mathbf{x_{scr}}$

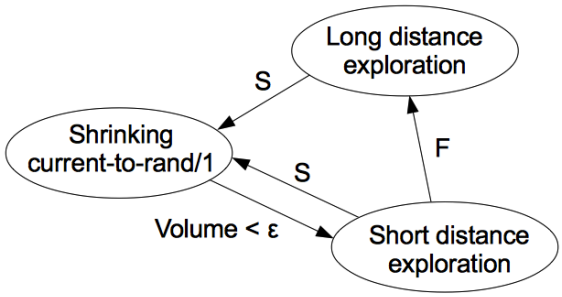


Figure. 6.7. Coordination of the operators for RIS-3SOME.

elite. Subsequently, for a fixed number of iterations, a run of DE/current-to-rand/1 with exponential crossover is executed over the micro-population. When the given budget allotted to the μDE operator is reached, if an improvement is found, a new hypercube is constructed around the new elite and μDE is repeated. Otherwise, the short distance exploration is activated. Compared to 3SOME and RIS-3SOME, the only difference in the inter-operator coordination logics is that, in order to force a more frequent activation of the μDE operator, thus guaranteeing its convergence, a budget limit equal to 5% of the total budget (in terms of fitness evaluations) is imposed over each activation of the long distance exploration, see figure 6.8. After this limit is reached, μDE is activated regardless of whether the long distance exploration has improved upon the current elite or not. This additional control also guarantees a balance in the activation of each of the three operators similar to that one of 3SOME and RIS-3SOME. It should be noted that, similar to RIS-3SOME, the μDE meme naturally embeds a form of "shrinking" over the most promising search region. However, compared to 3SOME and RIS-3SOME this variant is slightly more expensive on a memory viewpoint, because it needs M additional memory slots to store the micro-population. On the other hand, the computational cost is comparable to the two previous algorithms. Conversely, the third scheme relies on a different and heavier approach, since employing (1+1)-CMA-ES (Algorithm 15 described in Chapter 3.1.2.2) during the second stage. The resulting combination of 3SOME with $(1 + 1)$ Covariance Matrix Adaptation Evolution Strategy ((1+1)-CMA-ES-3SOME) makes use

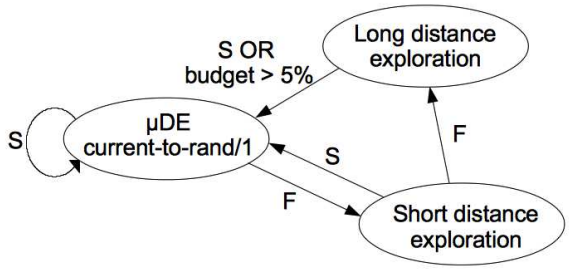


Figure. 6.8. Coordination of the operators for μDE-3SOME

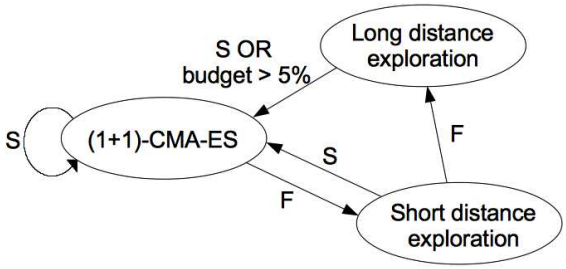


Figure. 6.9. Coordination of the operators for (1+1)-CMA-ES-3SOME

of the same coordination scheme of 3SOME, in which a run of (1+1)-CMA-ES is executed in place of the middle distance search operator, for a fixed budget (see Figure 6.9). Similarly to μDE-3SOME, each activation of the long distance stage is given a maximum budget equal to 5% of the total budget of the algorithm. It must be said that, due to the covariance matrix, this approach becomes more memory consuming with respect to the first two strategies. In order to understand the algorithmic contribution provided by each of the three variants described above, they have been compared with the original implementation of 3SOME on the entire noiseless BBOB2010, consisting of 24 test functions with different properties in terms of modality, separability, and ill-conditioning. To test the scalability of the proposed approaches, the whole benchmark has been run in 10, 20, 40 and 100 dimensions, thus considering $24 \times 4 = 96$ functions in total. Numerical results have been displayed in detailed tables grouped in Appendix E.1. In 10 dimensions, see Table E.9, RIS-3SOME is able to improve upon 3SOME in 10 out of 24 functions, while in 11 cases they are statistically equivalent, and only in 3 cases RIS-3SOME degrades the original performance of 3SOME: two of these three functions, namely f_3 and f_4, are indeed separable. In general, RIS-3SOME seems to performs better than 3SOME especially on non-separable multi-modal functions, in particularly the group of functions f_{15}-f_{19}. As for μDE-3SOME, it outperforms 3SOME in 15 cases, it is outperformed only in 3 uni-modal cases (f_3, f_{10}, and f_{12}), and it shows a similar performance in the remaining 6 cases. Thus μDE-3SOME seems to consistently and regularly improve upon 3SOME, especially on non-separable multi-modal functions. Similarly, (1+1)-CMA-ES-3SOME outperforms 3SOME in 15 cases, it is outperformed in 4 cases (2 separable functions), and it is equivalent in 5 cases. In this case the improvement provided by the CMA-ES scheme seems to be less focused on a specific group of functions, but rather "structural", since it displays a better performance both on separable (e.g. f_1 and f_2) and

non-separable functions, particularly uni-modal (f_{11}, f_{12}, f_{14}). With reference to Table E.10, in 20 dimensions, a similar trend emerges. RIS-3SOME improves upon 3SOME in 11 cases (again on non-separable multi-modal functions with global structure), while it is outperformed in 6 cases and it equals 3SOME in the remaining 7 cases. Similarly, μDE-3SOME outperforms 3SOME in 11 cases, it is outperformed on 5 functions, and it shows the same performance as 3SOME on 8 cases. In general μDE-3SOME seems to be indeed better suited than 3SOME for non-separable functions (both uni-modal and multi-modal, especially with global structure). Also (1+1)-CMA-ES-3SOME clearly outperforms 3SOME: in 17 cases out of 24 it obtains a better result, while it degrades the 3SOME performance only in 4 cases. Again, (1+1)-CMA-ES-3SOME seems to be globally better than 3SOME, although it seems to be extremely good especially on uni-modal functions, both separable and non-separable (e.g. f_1, f_2 and the function group f_{10}-f_{14}). Similar results were also obtained in 40 dimensions (Table E.11). RIS-3SOME displays a better performance than 3SOME in 12 cases (especially non-separable multi-modal functions with global structure), and a worse performance in only 4 cases (among which again f_3 and f_4). μDE-3SOME outperforms 3SOME in 11 cases, while it is outperformed in 6 cases: once again it seems to obtain better results especially on non-separable functions, both uni-modal and multi-modal. (1+1)-CMA-ES-3SOME instead outperforms 3SOME on 15 test functions, with different properties in terms of modality and separability, and it is outperformed in 6 cases (either separable, see f_3 and f_4, or not, see f_8 and f_9). These results are confirmed even in 100 dimensions, although in this case the advantages obtained modifying the original structure of 3SOME appear less prominent, see Table E.12. In particular, RIS-3SOME outperforms 3SOME in 9 cases, it is outperformed in 7 cases, and it equals 3SOME in the remaining 8 cases: similarly to lower dimensions, the improvements are more evident on non-separable multi-modal functions, but in this case only on those having an adequate global structure (function group f_{15}-f_{19}). μDE-3SOME performs better than 3SOME in 11 cases, while it is outperformed in 8 cases. Also in this case the pattern suggests the μDE-3SOME is better suited for non-separable multi-modal functions with global structure. Finally, (1+1)-CMA-ES-3SOME displays a better performance than 3SOME in 13 cases (either separable or non-separable), while 3SOME is more promising in 9 other cases. However, in this case there is no clear evidence of a global scheme, except that (1+1)-CMA-ES-3SOME seems to outperform 3SOME especially on non-separable uni-modal functions (f_{10}-f_{14}).

From the numerical results summarised above a few conclusions can be drawn. First of all, it is quite evident that the three 3SOME variants proposed here are all able to improve, sometimes remarkably, upon 3SOME. This is specially true for non-separable functions at low dimensions (from 10 to 40), while on semi-large scale problems (100 dimensions) the performance improvement is relatively limited. On lower dimensions, the results obtained in this study also show some more specific trends. Referring to the property taxonomy used to structure the BBOB2010 benchmark, it seems that the two variants based on DE/current-to-rand/1, namely RIS-3SOME and μDE-3SOME, are able to better exploit the global structure of some landscapes, and in general they show similar performances on the whole benchmark from 10 to 40 dimensions, tending to outperform 3SOME on non-separable functions, especially multi-modal. In a nutshell, these two variants can be considered equivalent in terms of global performance: this can be explained considering that, although their coordination scheme is different, both RIS-3SOME and μDE-3SOME rely on the same DE/current-to-rand/1 mutation scheme. On the other hand, the combination of 3SOME with the (1+1)-CMA-ES structure seems to produce the best global results on lower dimensions, both on separable and non-separable functions. However,

compared to the first two simpler schemes, it leads to minor improvements. A possible interpretation of these results is that, even though its robustness and mathematical elegance, (1+1)-CMA-ES is still prone to converge to local optima, especially on highly multi-modal problems. Thus its application within the 3SOME structure appears to be beneficial only on those landscapes whose number of optima does not grow with the problem dimension. In other words, especially on high dimensional problems, simpler approaches like 3SOME or RIS-3SOME are already successful without adding more complexity. This finding is in line with the Ockham's Razor (Iacca, Neri, Mininno, Ong & Lim 2012) .

Moreover, the fact that the three variants have similar performances raises the point that not only the choice of the operator makes the difference in MC, but also the algorithmic structure plays an important role, see next section (Chapter 6.2) for a deep investigation on this topic.

6.2 The importance of being structured

In order to formalise what is suggested in (Neri et al. 2012) and (Caraffini, Iacca, Neri & Mininno 2012*b*) is reported here an analysis on the structure of 3SOME (Caraffini, Iacca, Neri & Mininno 2012*a*). This analysis can be generalised to all the other MC optimisers and allows one to observe that, besides the components of an algorithm, the structure combining them also plays a crucially important role. On the contrary, most of the memetic design focuses on the components while the structure is unfortunately often overlooked. This tendency sometimes turns into a poor understanding of the algorithmic behaviour, and misleads the interpretation of results and algorithm potentials. A first attempt to analyse the rationales behind the choice of the algorithmic structure in MC has already been done in (Iacca, Neri, Mininno, Ong & Lim 2012) by introducing a bottom-up approach, i.e. the algorithm is designed from scratch, adding the minimum amount of as simple as possible components, each one with a well-defined algorithmic role. The main idea is that if each role is clear, a proper operator can be chosen accordingly within a certain class of operators, that regardless of their complexity, will provide similar global performances once integrated into the main algorithm. This idea also suggests that operators can be grouped into classes in order to address specific aspects, and automatically selected to build a structure displaying a better performance, see Chapter 7.2. Regarding 3SOME, further experiments on the coordination of the memes has been carried out in (Neri et al. 2012), and the second stage has been used to handle non-separable problems in (Caraffini, Iacca, Neri & Mininno 2012*b*), but little has been done for the final stage.

In order to demonstrate that a memetic structure is important and that the 3SOME structure is well-designed, the algorithmic operators have been modified within the same structure. More specifically, 3SOME versions where S is replaced by the Rosenbrock method (Algorithm 6, as described in Chapter 2.2.5 and Powell's Direction Set Method (Algorithm 7) with the GSS line minimisation method, see Chapter 2.2.6, have been tested and compared. In order to evaluate both the versions, 3SOME equipped with Rosenbrock local search (3SOME-Rosenbrock) and 3SOME equipped with Powell local search (3SOME-Powell), upon the original version, the three algorithms have been run over two complete benchmarks, i.e. BBOB2010 (in 10, 40 and 100 dimensions) and CEC2010 (in 1000 dimensions).

Numerical results show that, regardless of the choice of the specific memes, as long as the 3SOME structure contains memes which perform long, middle, and short distance explorations a similar performance is achieved. In 10, 20 and 40 dimensions, Table E.4, E.5 and E.6, in particular all the variants act the same way. As the problem dimension grows however, see Table E.7 for 100 dimensions results and Table E.8 for 1000, 3SOME outperforms its Powell-based variant, while 3SOME-Rosenbrock seems to offer a slightly better global behaviour. In general, the use of the Powell method appears beneficial especially on non-separable multi-modal functions with weak or adequate global structure and low dimensionality, while it can be detrimental in large scale problems. 3SOME-Rosenbrock shows better performances on separable uni modal functions and non-separable problems in 100 dimensions. It must be noted that these three LS methods perform different moves through $\mathcal{D}$, so small variations over different kind of problems are to be expected.

A careful analysis of the dynamic behaviour of the three algorithms with different problem dimensions was also performed. Tables 6.2(a)-6.2(c) display the memes activation, in terms of percentage of the total budget consumed by each meme, averaged over 100 runs on a subset of the BBOB2010 test functions in 10, 40 and 100 dimensions. In order to assess if the algorithmic dynamics depends on the optimisation problem, the subset was chosen selecting the first functions of each of the five subgroups of the BBOB2010, which differ in terms of separability, multi-modality, and ill-conditioning. It can be seen that, for each algorithm, the coordination scheme scales up nicely with the problem dimensionality, since the activation percentages seem to be almost constant (apart from natural stochastic fluctuations) as the number of dimensions grows. This behaviour is likely to be a consequence of the serial nature of the 3SOME structure (and its variants), which guarantees the same budget allocation regardless of the dimensionality. On the other hand, the comparison among the three algorithms shows that their behaviour is very similar (see f_1, f_{10} apart from 3SOME-Powell in 10 dimensions, and f_{15}). Two interesting exceptions are f_6 and f_{20}, where the three algorithms show different trends, especially in larger dimensions. In particular, while 3SOME and 3SOME-Rosenbrock seem to have similar dynamics, 3SOME-Powell uses a longer budget in the long exploration stage. A possible explanation of this phenomenon is that on some peculiar non-separable multi-modal landscapes the short distance search and the Rosenbrock method tend to consume more budget than the Powell algorithm, thus guaranteeing a better budget balance. The same analysis was also performed on theCEC2010 benchmark, with similar results. For the sake of brevity, numerical results on memes activation have not been reported for this test suite. The fitness trends on two of the test functions from the benchmark, see Figure 6.10 and 6.11, are instead reported. It is interesting to notice that, except f_{11} and f_{16} where the three algorithms show remarkably different dynamics (with 3SOME outperforming its two variants), in the other cases the fitness trends are specular. Apart from these two cases, similar trends were obtained in all the remaining 18 functions of the CEC2010 benchmark.

In summary, these results confirm that, despite the three algorithms use different exploitative components (with different budget conditions), their global behaviour is almost completely ruled by the structure, that is the coordination scheme.

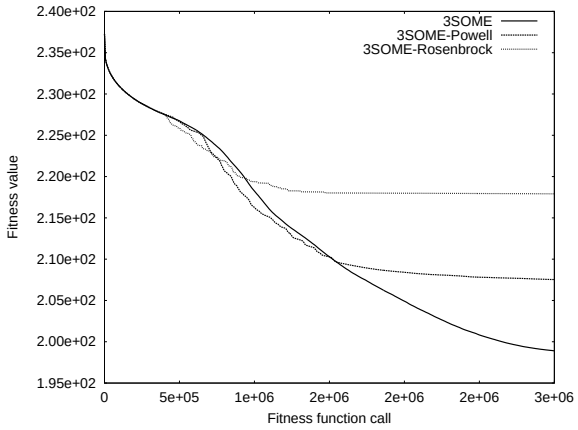


Figure. 6.10. Average fitness trend for 3SOME, 3SOME-Powell and 3SOME-Rosenbrock on f_{11} from CEC2010 in 1000 dimensions.

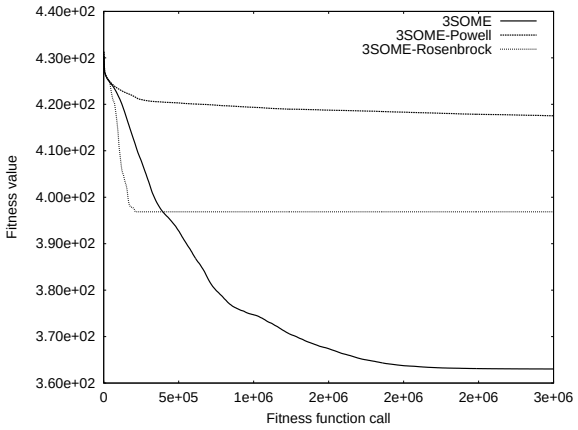


Figure. 6.11. Average fitness trend for 3SOME, 3SOME-Powell and 3SOME-Rosenbrock on f_{16} from CEC2010 in 1000 dimensions.

Table 6.2. Memes activation on BBOB2010 in 10 (a), 40 (b) and 100 (c) dimensions.

(a)

	3SOME	3SOME-Powell	3SOME-Rosenbrock
f_1	STAGE-1 = 90.85% STAGE-2 = 0.91% STAGE-3 = 8.24%	STAGE-1 = 97.36% STAGE-2 = 1.04% STAGE-3 = 1.6%	STAGE-1 = 90.53% STAGE-2 = 1.34% STAGE-3 = 8.13%
f_6	STAGE-1 = 14.19% STAGE-2 = 2.59% STAGE = 83.22%	STAGE-1 = 83.97% STAGE-2 = 4.88% STAGE = 11.15%	STAGE-1 = 0.005% STAGE-2 = 3.593% STAGE = 96.402%
f_{10}	STAGE-1 = 0.01% STAGE-2 = 2.15% STAGE-3 = 97.84%	STAGE-1 = 23.24% STAGE-2 = 5.33% STAGE-3 = 71.43%	STAGE-1 = 0.008% STAGE-2 = 1.624% STAGE-3 = 98.368%
f_{15}	STAGE-1 = 74.77% STAGE-2 = 0.95% STAGE-3 = 24.28%	STAGE-1 = 82.22% STAGE-2 = 1.84% STAGE-3 = 15.94%	STAGE-1 = 80.01% STAGE-2 = 1.64% STAGE-3 = 18.35%
f_{20}	STAGE-1 = 75.36% STAGE-2 = 1.25% STAGE-3 = 23.39%	STAGE-1 = 90.63% STAGE-2 = 2.17% STAGE-3 = 7.2%	STAGE-1 = 54.47% STAGE-2 = 2.38% STAGE-3 = 43.15%

(b)

	3SOME	3SOME-Powell	3SOME-Rosenbrock
f_1	STAGE-1 = 85.31% STAGE-2 =1.64% STAGE-3 = 13.05%	STAGE-1 = 95.36% STAGE-2 = 2.24% STAGE-3 = 2.4%	STAGE-1= 95.96% STAGE-2 = 1.52% STAGE-3 = 2.52%
f_6	STAGE-1 = 29.08% STAGE-2 =3.74% STAGE-3 = 67.18%	STAGE-1 = 80.67% STAGE-2 = 3.93% STAGE-3 = 15.4%	STAGE-1 = 0.002% STAGE-2 = 2.624% STAGE-3 = 97.374%
f_{10}	STAGE-1 = 0.01% STAGE-2 = 4.1% STAGE-3 = 95.89%	STAGE-1 = 0.001% STAGE-2 = 4.72% STAGE-3 = 95.279%	STAGE-1 = 0.001% STAGE-2 = 2.032% STAGE-3 = 97.967%
f_{15}	STAGE-1 = 64.64% STAGE-2 = 1.99% STAGE-3 = 33.37%	STAGE-1 = 63.85% STAGE-2 = 7,69% STAGE-3 = 28.46%	STAGE-1 = 78.65% STAGE-2 = 3.01% STAGE-3 = 18.34%
f_{20}	STAGE-1 = 47.81% STAGE-2 = 2.45% STAGE-3 = 49.74%	STAGE-1 = 82.8% STAGE-2 = 5.2% STAGE-3 = 12%	STAGE-1= 42.07% STAGE-2 = 3.36% STAGE-3 = 54.57%

(c)

	3SOME	3SOME-Powell	3SOME-Rosenbrock
f_1	STAGE-1 = 85.31% STAGE-2 =1.64% STAGE-3 = 13.05%	STAGE-1 = 95.36% STAGE-2 = 2.24% STAGE-3 = 2.4%	STAGE-1 = 95.96% STAGE-2 = 1.52% STAGE-3 = 2.52%
f_6	STAGE-1= 29.08% STAGE-2 =3.74% STAGE-3 = 67.18%	STAGE-1 = 80.67% STAGE-2 = 3.93% STAGE-3 = 15.4%	STAGE-1 = 0.002% STAGE-2 = 2.624% STAGE-3 = 97.374%
f_{10}	STAGE-1 = 0.01% STAGE-2 = 4.1% STAGE-3 = 95.89%	STAGE-1 = 0.001% STAGE-2 = 4.72% STAGE-3 = 95.279%	STAGE-1 = 0.001% STAGE-2 = 2.032% STAGE-3 = 97.967%
f_{15}	STAGE-1 = 64.64% STAGE-2 = 1.99% STAGE-3 = 33.37%	STAGE-1 = 63.85% STAGE-2 = 7,69% STAGE-3 = 28.46%	STAGE-1 = 78.65% STAGE-2 = 3.01% STAGE-3 = 18.34%
f_{20}	STAGE-1 = 47.81% STAGE-2 = 2.45% STAGE-3 = 49.74%	STAGE-1 = 82.8% STAGE-2 = 5.2% STAGE-3 = 12%	STAGE-1 = 42.07% STAGE-2 = 3.36% STAGE-3 = 54.57%

6.3 Seriously simple MC structures with high performances

As previously mentioned, 3SOME's structure appears to have two main irreplaceable stages, i.e. long and short distance exploration, plus an intermediate search to help adjust the final move, taking the most out of the deterministic LS. Considering the study in (Caraffini, Iacca, Neri & Mininno 2012*a*), re-proposed in Chapter 6.2, the middle distance exploration seems to be of help in few cases, while it is activated only for a small portion of the total computational budget over the majority of the problems under study. The numerous variants described in this thesis tried and enhanced this stage, e.g. making it able to address non-separability or performing over LSOP. On the other hand, the successful use of 3SOME in real-world memory restrictive and real-time applications (Iacca, Caraffini & Neri 2013) originated the idea that in many real-world cases a further simplification (instead of potentiating the old operators) could be beneficial, and that in the real-world case simpler structure are preferable and a trade-off can be accepted. Keeping this in mind, a minimalistic optimiser has been proposed in (Caraffini, Neri, Gongora & Passow 2013) by removing the middle search stage, and simplifying the long distance search to the essential. The resulting algorithm, Re-sampled Search (RS) turned out to be not only seriously simple and light, but also extremely efficient over a large set of test-bed functions, displaying astonishing results improving upon the regular 3SOME implementation (Table 6.4).

This optimiser makes use of only two operators that progressively perturb a single point, the elite solution. The first operator implements a re-sampling mechanism which is supposed to generate a solution far away from the current elite, while the second implements a LS that exploits the area of the decision space suggested by the re-sampling operator. In this sense, RS is only a simple, but performing, implementation of a multi-start local search. However, the proposed scheme is thought as a global optimisation algorithm in the fashion of MC, as a degenerative case of the 3SOME algorithm. In fact, the LS is committed to the S operator, as in Algorithm 5, since showing to be able to perform well over a wide range of problems. The re-sampling, in its simplicity, plays a crucial role since the lack of the Long distance search could turn in premature convergence due to the poor exploration of the landscape. Moreover, it helps improve S performances, since as well as the others LS deterministic techniques, its search strongly depends on the choice of the initial point. Despite this scheme has shown to be promising, it has then been further tested and improved thanks to a slight modification (Caraffini, Neri, Passow & Iacca 2013*a*), which has been justified empirically though a large series of experiments. Going into details, it has been observed that adding to random restart procedure appears to beneficially affect the performance of the RS algorithm. The resulting Re-sampled Inheritance Search (RIS) performs better in high dimensions, but has also been proved to be at least as good as its variant without inheritance and, in some cases, significantly more promising also in low dimensional problems (see Table 6.3 for results in 30 dimensions on CEC2005). The inheritance mechanism assures that a part of the genotype of the most promising candidate solution is used to enhance its performance via exponential cross-over (as in Algorithm 19) between the newly sampled point and the current best solution, i.e. the elite solution, being a single solution algorithm. Thanks to this mechanism, a trial solution $\mathbf{x_t}$ is generated, and its fitness instantly compared with that of the elite. If the newly generated solution outperforms the elite, an elite replacement occurs, but regardless off its fitness value the trial solution is then passed as input for the S operator, for local refinements. The working principle of RIS is shown in algorithm 32. It can be observed that after the LS procedure, a second comparison is

Table 6.3. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = RIS) for RIS against its variant without inheritance, RS, on CEC2005 in 30 dimensions

	RIS	RS	
f_1	$\mathbf{-4.50e+02 \pm 1.31e-13}$	$-4.50e+02 \pm 1.54e-13$	+
f_2	$-4.50e+02 \pm 1.23e-10$	$\mathbf{-4.50e+02 \pm 1.29e-10}$	=
f_3	$\mathbf{2.03e+05 \pm 1.25e+05}$	$2.30e+05 \pm 1.39e+05$	=
f_4	$\mathbf{1.88e+04 \pm 5.57e+03}$	$3.42e+04 \pm 6.04e+03$	+
f_5	$\mathbf{3.48e+03 \pm 7.70e+02}$	$4.04e+03 \pm 8.22e+02$	+
f_6	$6.91e+02 \pm 4.39e+02$	$\mathbf{6.45e+02 \pm 6.52e+02}$	=
f_7	$-1.80e+02 \pm 3.38e-03$	$\mathbf{-1.80e+02 \pm 3.00e-03}$	=
f_8	$\mathbf{-1.20e+02 \pm 4.05e-04}$	$-1.20e+02 \pm 7.51e-03$	=
f_9	$\mathbf{-1.18e+02 \pm 1.63e-05}$	$-1.17e+02 \pm 3.69e-01$	+
f_{10}	$\mathbf{2.14e+02 \pm 1.52e+01}$	$2.15e+02 \pm 1.49e+01$	=
f_{11}	$\mathbf{1.09e+02 \pm 1.90e+00}$	$1.10e+02 \pm 1.76e+00$	=
f_{12}	$1.90e+02 \pm 1.41e+03$	$\mathbf{1.25e+02 \pm 1.20e+03}$	=
f_{13}	$-1.21e+02 \pm 1.92e+00$	$\mathbf{-1.22e+02 \pm 1.76e+00}$	=
f_{14}	$-2.86e+02 \pm 2.79e-01$	$\mathbf{-2.86e+02 \pm 3.43e-01}$	=
f_{15}	$\mathbf{1.44e+03 \pm 3.89e-01}$	$1.45e+03 \pm 5.00e-01$	+
f_{16}	$\mathbf{1.55e+03 \pm 7.51e+00}$	$1.55e+03 \pm 6.53e+00$	+
f_{17}	$\mathbf{1.66e+03 \pm 1.54e+01}$	$1.70e+03 \pm 1.30e+01$	+
f_{18}	$\mathbf{9.10e+02 \pm 4.58e-10}$	$9.10e+02 \pm 4.47e-10$	=
f_{19}	$9.10e+02 \pm 4.52e-10$	$\mathbf{9.10e+02 \pm 4.72e-10}$	=
f_{20}	$\mathbf{9.10e+02 \pm 4.29e-10}$	$9.10e+02 \pm 4.79e-10$	=
f_{21}	$\mathbf{1.69e+03 \pm 4.33e+00}$	$1.70e+03 \pm 4.06e+00$	+
f_{22}	$\mathbf{2.42e+03 \pm 2.92e+01}$	$2.45e+03 \pm 2.98e+01$	+
f_{23}	$\mathbf{1.72e+03 \pm 6.16e+00}$	$1.73e+03 \pm 5.51e+00$	+
f_{24}	$\mathbf{1.68e+03 \pm 6.83e+00}$	$1.68e+03 \pm 6.59e+00$	+
f_{25}	$\mathbf{1.43e+03 \pm 3.74e+02}$	$1.44e+03 \pm 2.64e+02$	+

needed to verify whether the new point has to replace the current elite solution or not. In case of failure, this point is just discarded and another sampling occurs. For the sake of completeness, it must be said that the S operator is here used (for both RS and RIS) with the stop criterion given in Equation 2.5, and not continued for a fixed budget. In this way, the local refinement is continued as long as the exploratory radius does not allow any improvements. Although the re-sampling is an operation that is performed only occasionally, and thus basically leaving all the budget to the LS, the transmission of some variables from the most promising solution to a newly sampled point appears to have a certain impact on the global performance of the algorithm, so being able to successfully replace the more budget requiring long distance operator of 3SOME. A graphic representation of RIS sequential structures, i.e. two states passing a single solution to each other, is given in Figure 7.5 (Chapter 7). Both the variants, RS and RIS have been tested over 76 test problems from CEC2005 (30

Algorithm 32 Re-sampled Inheritance Search

$\mathbf{x_{elite}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
while *budget condition* **do**
  $\mathbf{x_{trial}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
  $\mathbf{x_{trial}} \leftarrow$ XOVEREXP($\mathbf{x_{trial}}, \mathbf{x_{elite}}$) ▷ Algorithm 19
  if $f\,(\mathbf{x_{trial}}) < f\,(\mathbf{x_{elite}})$ **then**
    $\mathbf{x_{elite}} \leftarrow \mathbf{x_{trial}}$
  end if
  $\mathbf{x_{trial}} \leftarrow$ S($\mathbf{x_{trial}}$) ▷ Algorithm 5 with Equation 2.5
  if $f\,(\mathbf{x_{trial}}) < f\,(\mathbf{x_{elite}})$ **then**
    $\mathbf{x_{elite}} \leftarrow \mathbf{x_{trial}}$
  end if
end while
Output $\mathbf{x_{elite}}$

dimensions), BBOB2010 (100 dimensions), CEC2008 and CEC2010 (1000 dimensions), and compared against

Table 6.4. Holm-Bonferroni procedure on the algorithms under consideration (reference algorithm: RIS, Rank= $6.46e+00$).

j	Optimizer	$Rank$	z_j	p_j	δ/j	Hypothesis
1	PMS	6.58e+00	2.67e-01	6.05e-01	5.00e-02	Accepted
2	CCPSO2	6.58e+00	2.67e-01	6.05e-01	2.50e-02	Accepted
3	RS	6.09e+00	-8.29e-01	2.03e-01	1.67e-02	Accepted
4	MA-LSCh	5.74e+00	-1.63e+00	5.17e-02	1.25e-02	Accepted
5	MDE-pBX	5.74e+00	-1.63e+00	5.17e-02	1.00e-02	Accepted
6	3SOME	5.70e+00	-1.72e+00	4.29e-02	8.33e-03	Accepted
7	CLPSO	5.64e+00	-1.84e+00	3.32e-02	7.14e-03	Accepted
8	JADE	4.62e+00	-4.15e+00	1.69e-05	6.25e-03	Rejected
9	cDE	1.79e+00	-1.05e+01	3.72e-26	5.56e-03	Rejected

8 performing algorithms as listed in Table 6.4, displaying the overall results in term or Holm-Bonferroni test. Since RIS has proven to provide better results than RS (also the rank value in Table 6.4 is slightly higher), extended numerical results on each single problem functions, see Appendix E.2, have been referenced to RIS while RS has not been reported in this work. For the sake of brevity, the tables for the PMS algorithm (displaying the highest rank over this test-bed), have also not been reported. Moreover, PMS has not been described yet in this work, as it deserves a specific attention, Chapter 7.1. For a comparison of RIS against PMS and RS one can read (Caraffini, Neri, Passow & Iacca 2013*a*). As shown in Table 6.4, RIS is ranked third amongst all the algorithms considered in this study. In accordance with the Ockham's Razor in MC, this is one further confirmation that the performance of simple, properly designed algorithms can be as good as (or even better than) the performance of modern complex algorithms, composed of multiple and computationally expensive components, such as MA-LSCh, MDE-pBX, JADE, 3SOME and CLPSO. A full set of detailed tables have been placed in Appendix E.2 in order to be able to draw more precise conclusions on specific functions, with different properties and dimensions. These results have been divided into groups. Tables E.13, E.14, E.15, and E.16 show the comparison against popular meta-heuristic, i.e. 3SOME, CLPSO, andJADE, for the four benchmarks under consideration. Tables E.17, E.18, E.19, and E.20 show the comparison against recent (state-of-the-art) algorithms, i.e. CCPSO2, MA-LSCh and MDE-pBX. Despite its simple structure, the RIS algorithm outperforms both the popular meta-heuristics under consideration. In particular, RIS overtakes CLPSO and JADE in 30 and 100 dimensions, while in 1000 dimensions CLPSO is competitive over the test-bed CEC2010. With reference to Table E.13, it can be seen that for the full set of benchmark under consideration, RIS is outperformed by all the other algorithms only in function f_4, which, being subjected to a Gaussian noise ($\mathcal{N}(0,1)$), is more suited to a population-based algorithm. It can also be observed that RIS achieves the best performance in 9 cases, 3SOME also in 9 cases, JADE in 8 case while CLPSO in only one case. Nonetheless, regarding problems f_1, f_2, f_9, f_{18}, f_{19}, and f_{20}, despite the fact that 3SOME is possibly slightly more robust, RIS and 3SOME detect very similar solutions. On the contrary, in most of the cases where the RIS algorithm appears to be promising against the 3SOME scheme, there is an important margin of difference in terms of final fitness value. Equally relevant results are displayed in Table E.14, for BBOB2010 in 100 dimensions, and in Table E.15 for CEC2008 in 1000 dimensions. In particular, Table E.15 highlights an extremely good behaviour of the proposed algorithm over large-scale separable problems. In fact, RIS widely outperforms JADE on a regular basis and is significantly outperformed by CLPSO in only one case (see function f_4). The comparison against MDE-pBX shows that RIS displays a better performance for all the groups of dimensionality values

considered in this article. Regarding the comparison against CCPSO2, RIS tends to outperform it at 30 and 100 dimension. For large scale problems, CCPSO2 displays a good performance and slightly outperforms RIS. A reversed situation occurs for the MA-LSCh-CMA algorithm. The latter algorithm is very efficient in 30 and 100 dimensions, where it clearly outperforms RIS. On the other hand, this trend is not confirmed in high dimensions, since RIS statistically outperforms MA-LSCh-SSW for all the problems in 1000 dimensions. With respect to the 3SOME algorithm, RIS appears to detect better solutions in most of the analysed cases. For the sake of brevity extended results against PMS and cDE are not reported in Appendix E, since the PMS algorithm will be extensively treated in the next chapter and cDE does not perform well against RIS (see Holm-Bonferroni procedure in Table 6.4). These numerical data can be found in (Caraffini, Neri, Passow & Iacca 2013*a*), while for the sake of completeness, Figure 6.12 (f_{25} of CEC2005 in 30 dimensions), 6.13 (f_{24} of BBOB2010 in 100 dimensions) and 6.14 (f_{11} of CEC2010 in 1000 dimensions) have been reported to graphically show the average (over 100 runs) performance trends for three significant optimisation problems amongst the 76 under consideration.

All in all, it can be noted that RIS appears to be especially efficient in tackling large scale problems. This can be seen as a consequence of exploitative approaches being likely to be more successful than exploratory ones in high dimensions. Moreover, it once again suggests that LSOP can be tackled with moves along the axis, see Chapter 8. As a final remark, the fact must be stressed that among the algorithms tested in this study, RIS is one of the most suitable for engineering applications, since its negligible computational overhead (Figure 6.15) and memory footprint make it compatible for integration in embedded systems. As instance, a real world application, i.e. optimal tuning of a Proportional Integral Derivative (PID) regulator, has also been successfully optimised in (Caraffini, Neri, Passow & Iacca 2013*a*). Furthermore, it must be said that the concept of restarting the search is not new in optimisation, see for example the Population-based Iterated Local Search (Thierens 2004) for combinatorial domains, or more recent CMA-ES restart variants for continuous problems (Loshchilov, Schoenauer & Sebag 2012). Nonetheless, the implementation proposed in this study presents features, such as modest algorithmic temporal overhead, minimal memory footprint and the capability of providing high quality solutions, which make it preferable to modern complex multi-start algorithms in many scenarios.

More in detail, Figure 6.15 displays the average (over 30 runs) computational overhead, i.e. time of a run without the time required to perform the fitness evaluations, depending on the problem dimensionality n of the algorithms under examination. Each run has been continued until 10000 fitness evaluations. It must be mentioned that an enhanced version of RIS also exists (Caraffini, Iacca, Neri, Picinali & Mininno 2013). This algorithm, employs a super-fit scheme by first applying the CMA-ES framework in order to find a promising initial solution for starting the re-sampled search (RIS). This variant, CMA-ES super-fit scheme for the re-sampled inheritance search (CMA-ES-RIS), has shown good results on complex test functions, but on the other hand the use of CMA-ES for the preliminary search for the super-fit initial point, makes it inappropriate for many real-time real-world applications.

A further confirmation of the fact that simple structures employing a simple perturbation logic, e.g. exploration of each design variable at time, can be as good as complex structures, especially in LSOP, is

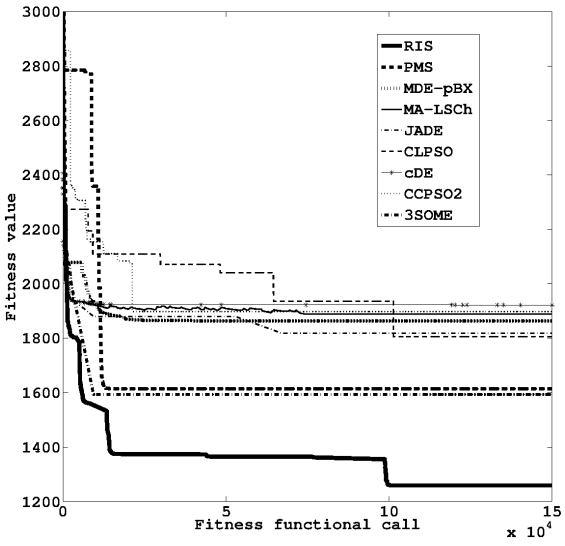


Figure. 6.12. Average fitness trend for RIS against popular meta-heuristics on f_{25} from CEC2005 in 30 dimensions.

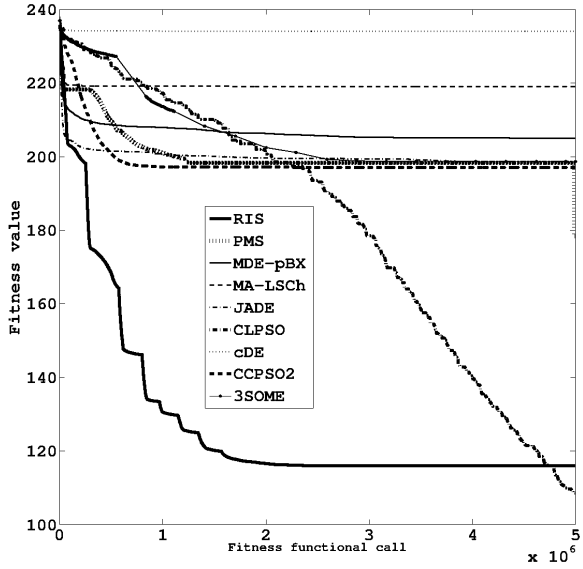


Figure. 6.14. Average fitness trend for RIS against popular meta-heuristics on f_{11} from CEC2010 in 1000 dimensions.

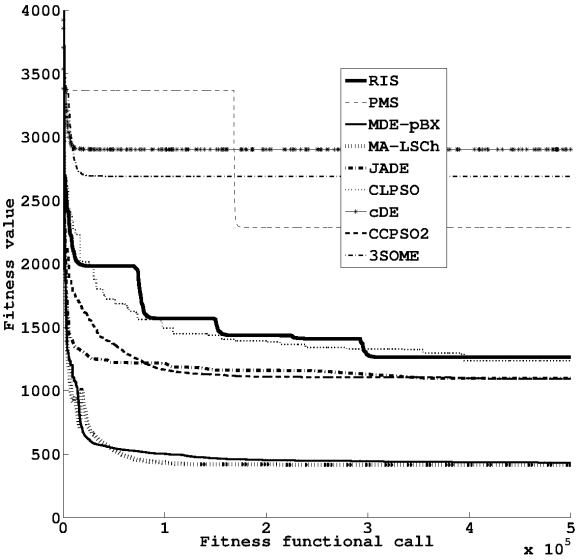


Figure. 6.13. Average fitness trend for RIS against popular meta-heuristics on f_{24} from BBOB2005 in 100 dimensions.

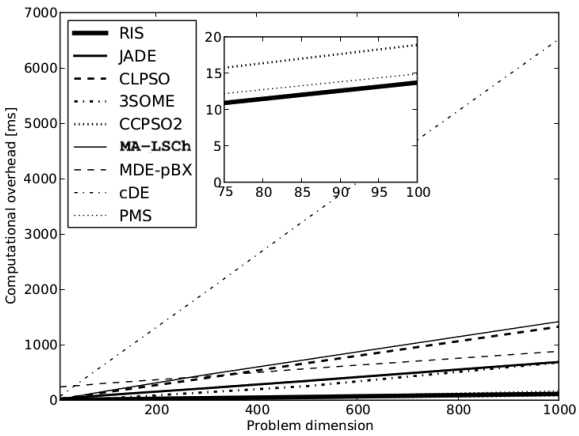


Figure. 6.15. Average computational complexity (overhead vs dimensionality) for RIS against other popular meta-heuristics.

given in (Iacca, Caraffini, Neri & Mininno 2013). The ISPO algorithm described in Chapter 3.2.1, is enhanced here with a "learning mechanism" and, in the fashion of MC, with one more operator implementing a "restart procedure". The learning process has been here introduced in order to estimate the level of separability of the optimisation problem at hand and change the perturbation order of the variables accordingly, so improving the old scheme ISPO that presents some flaws in dealing with non separable problems. The re-start procedure is instead activated in case a full iteration of the basic perturbation scheme, i.e. all of the n design variables must be scanned and perturbed, has been unsuccessful. In this case some of the design variables of the processed particle are randomly chosen within $\mathcal{D}$ with the aim of providing the next iteration of the algorithm with a new starting particle, hopefully located in a better zone of the search space. It must be noted, see Algorithm 33, that the restart routine actually replaces the old design variable only if the newly sampled bring an improvement, otherwise the old value is restored. Moreover, the number of design variables affected by the restart procedure depends on the progress of the optimisation process, i.e. higher (exploration) at the beginning and lower towards the end (exploitation), since the CR factor in Algorithm 33 is dynamically allocated considering the number of the elapsed functional calls. These two new modifications add to the old Intelligent Single Particle Optimisation the ability of adapting to the problem and jumping out from an unfavourable basin of attraction and so making it, to some extent, smarter than its predecessor, thus the name Very Intelligent Single Solution Optimisation (VISPO). Going into detail, with reference to Algorithm 34, it can be seen that during each macro-iteration

Algorithm 33 VISPO Restart Routine

procedure RESTART($(\mathbf{x}, N_{eval}, n_{eval})$)
  $CR \leftarrow \left(\frac{N_{eval}-n_{eval}}{N_{eval}}\right)^2$
  for $i = 1 : n$ **do**
    if $\mathcal{U}(0,1) < CR$ **then**
      $\mathbf{x}^*[i] \leftarrow \mathcal{U}\left(\mathbf{x}^{\mathbf{L}}[i], \mathbf{x}^{\mathbf{U}}[i]\right)$
    else
      $\mathbf{x}^*[i] \leftarrow \mathbf{x}[i]$
    end if
    $i \leftarrow i + 1$
  end for
  if $f(\mathbf{x}^*) < f(\mathbf{x})$ **then**
    $\mathbf{x}^* \leftarrow \mathbf{x}$
  end if
  Output $\mathbf{x}$
end procedure

each i-th variable is perturbed H times, where H is a parameter of the algorithm. The velocity update has been simplified with respect to the one used in ISPO as follows:

$$v = \mathbf{A}[i] \cdot \mathcal{U}(-0.5, 0.5), \ i = 1, 2 \ldots, n \tag{6.1}$$

The velocity v must be thus calculated at each i-th step and then added to the i-th variable $\mathbf{x}$ at time and then evaluated. In case the particle prior to the perturbation was better than the perturbed one, v is halved. This way incrementally smaller perturbations are attempted. On the other hand, if the perturbation produces a fitness improvement, the same velocity is used for the next perturbation and a counter s ("success"), which is initialized to zero at the beginning of the optimisation, is incremented. The counter s is used, after LP (learning period) "macro-iterations" (consisting of $n \times H$ trials), to learn, empirically, the level of separability

of the fitness function. More specifically, if the quantity $s/(n \cdot LP)$ is smaller than a given threshold, the number of perturbations H, which will be applied in the next stages of the algorithm, is set to one, otherwise the initial value of H is retained. This mechanism can be explained as follows. If the percentage of successful perturbations, averaged over the problem dimension n and the number of learning periods LP, is smaller than a threshold, then the problem can be considered, with some degree of approximation, non-separable. In this case, it makes sense to attempt, in later stages of the algorithm, a single perturbation for each dimension. On the other hand, if this percentage is bigger than the given threshold (meaning that during the learning periods an averagely high number of successful perturbations was obtained per each variable) the level of separability of the problem can be considered higher. Thus it is worthwhile to keep on trying multiple perturbations per dimension, aiming at exploiting the separability of the fitness function. This mechanism is rather minimalistic, and based on empirical rules. It represents a simple attempt to implement a learning rule which is able to determine the level of separability of a given fitness function. More complex and rigorous methods can be envisioned and are currently under investigation, with the final goal of creating an intelligent agent capable of automatically designing an algorithm based on the characteristics of the optimization problem at hand. Still, despite the simplicity of the mechanism implemented in VISPO, numerical results show that it provides a performance improvement with respect to ISPO (Iacca, Caraffini, Neri & Mininno 2013). From the MC point of view, this algorithm can be seen as a deterministic sequential repetition of two logics. The first one (referred as VISPO Perturbation Scheme, VPS, in Table 7.12), further simplifies the old ISPO perturbation scheme, but acts in a more performing way, being able to dynamically change the number of consecutive perturbations on the same design variable relying on the "learning mechanism". The second performs a restart procedure which randomly samples a portion of the design variables of the particle within D.

From the numerical data, extended table are grouped in Appendix E.3, it can be immediately see that the proposed optimiser aoutper ISPO on a regular basis, in both low dimensional problems, e.g. function from CEC2005 and BBOB2010 in 30 and 100 dimension values respectively, and LSOP as those in CEC2010. Figure 6.16 graphically depict this situation for f_4 from BBOB2010 in 100 dimensions. A general overview across the three aforementioned test suites can be see in Table 6.5, where ISPO displays, in terms of Holm-Bonferroni procedure, the worst rank and the comparison against VISPO presents the rejection of the null-hypothesis.

Moreover, comparisons with CLPSO and JADE on the CEC2005 benchmark in 30 dimensions, display a respectable performance for VISPO, see Table E.21. Especially against CLPSO, which is a rather complex PSO algorithm, VISPO shows that its simple structure is superior in 15 cases out of 25. On the other hand, the comparison with JADE shows a substantially equivalent performance (12 "+" and 13 "-"). This situation is reversed when the problem dimensionality increases, especially on BBOB in 100 dimensions, where VISPO performs better than JADE, particularly on separable and uni-modal functions, and shows the same performance as CLPSO (Table E.22). A similar trend can also be observed in 1000 dimensions for the CEC2010 benchmark, Table E.23, where once again VISPO is able to deal with the curse of dimensionality at least as well as CLPSO and JADE, see for example the trend for the "Shifted Elliptic" function in Figure 6.17.

Algorithm 34 Very Intelligent Single Particle Optimisation

$\mathbf{x} \leftarrow randomSampling\,(1, n, \mathcal{D})$
for $i = 1 : n$ **do**
  $\mathbf{A}[i] \leftarrow \left(\mathbf{x^U}[i] - \mathbf{x^L}[i]\right)$
  $i \leftarrow i + 1$
end for
$s \leftarrow 0$
$k \leftarrow 0$
while $n_{eval} \leq N_{eval}$ **do**
  $\mathbf{x}^* \leftarrow \mathbf{x}$
  $\mathbf{x_{old}} \leftarrow \mathbf{x}$
  for $i = 1 : n$ **do**
    $v \leftarrow \mathbf{A}[i] \cdot \mathcal{U}\,(-0.5, 0.5)$ ▷ Equation 6.1
    for $j = 1 : H$ **do**
      $\mathbf{x_{old}}[i] \leftarrow \mathbf{x}[i]$
      $\mathbf{x}[i] \leftarrow \mathbf{x}[i] + v$
      if $f\,(\mathbf{x}) < f\,(\mathbf{x_{old}})$ **then**
        $s \leftarrow s + 1$
      else
        $v \leftarrow \frac{v}{2}$
        $\mathbf{x}[i] \leftarrow \mathbf{x_{old}}[i]$
      end if
      $j \leftarrow j + 1$
    end for
    $i \leftarrow i + 1$
    $n_{eval} \leftarrow n_{eval} + 1$
  end for
  if $k == LP$ **then**
    if $s/(n \cdot LP) < threshold$ **then**
      $H \leftarrow 1$
    end if
  end if
  if $f\,(\mathbf{x}^*) == f\,(\mathbf{x})$ **then**
    $\mathbf{x} \leftarrow$RESTART$(\mathbf{x}, N_{eval}, n_{eval})$ ▷ Algorithm33
  end if
  $k \leftarrow k + 1$
end while
Output $\mathbf{x}$

Table 6.5. Holm-Bonferroni procedure on the algorithms under consideration (reference algorithm: VISPO, Rank$= 2.68e + 00$).

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	CLPSO	2.75e+00	4.26e-01	6.65e-01	5.00e-02	Accepted
2	JADE	2.43e+00	-1.45e+00	7.39e-02	2.50e-02	Accepted
3	ISPO	2.12e+00	-3.32e+00	4.50e-04	1.67e-02	Rejected

6.4 Chapter remarks

This chapter answered to all the intermediate research questions **IRQ I**, **IRQ II** and **IRQ III**.

In particular, multiple 3SOME variants have been first proposed in order to address **IRQ I** (Chapter 6.1). Amongst them, S-3SOME has proven to be particularly efficient on LSOP, rising the point that in large scale optimisation, algorithm employing exploitative operators are to be preferred, but also the idea that the use of operators perturbing the design variables along the axes are suitable for high-dimensional problems. Moreover, the fact that the efficiency of the three 3SOME variants for handling non separability decreases while the problem dimensionality grows, open a new interrogative. This behaviour makes one think that at high dimensions, non separable functions can be optimised with simple operators like S. This result represents a turning point, and has to be taken into consideration while addressing a precise problem with an optimisation

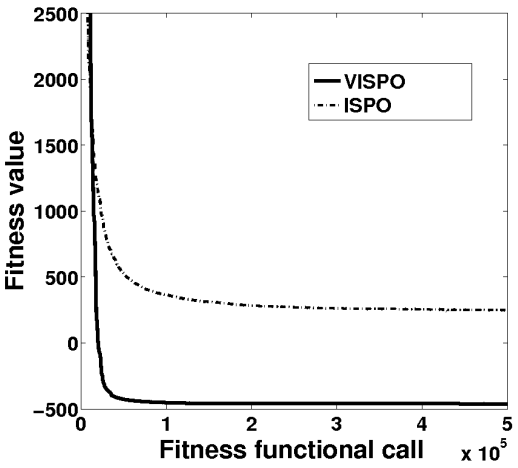


Figure. 6.16. Average fitness trend for VISPO against ISPO on f_4 from BBOB2010 in 100 dimensions.

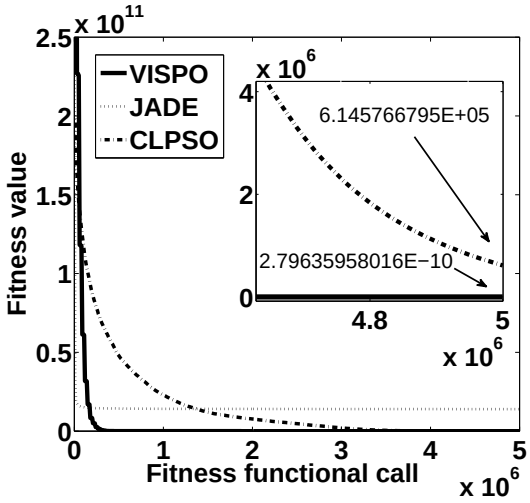


Figure. 6.17. Average fitness trend for VISPO against popular meta-heuristics on f_1 from CEC2010 in 100 dimensions.

algorithm. To some extent, this outcome suggests us that in large scale optimisation, also non-separable functions can be success fully addressed by simply perturbing one design variable at a time (in contrast with the definition of non-separable function).

This observation is also confirmed by the numerical results obtained with RIS and VISPO. Both of them have shown good performances on LSOP, and both of them are simple structures performing move along the axes. In this light, the study reported in Chapter 6.3 not only answers to **IRQ III**, but also empirically shows that simple and exploitative algorithms, with a search logic operating along the axes, perform well on LSOP . Chapter 8 will get back to this phenomenon, tying to give a formal explanation for it.

The (**IRQ II** is also addressed in a study focusing on the importance of the algorithmic structure with respect to the choice of the single component, see Chapter 6.2), which has been conducted by testing three different LS routines within the 3SOME framework. Numerical data have shown that in a well designed optimiser, the algorithmic structure is as important as the components forming it. This concept will be extended and formally generalised in Chapter 7.

Chapter 7

Novel memetic structures

While the study of coordination schemes in MAs is nowadays quite a mature research area, see for example the excellent tutorial (Krasnogor & Smith 2005), the taxonomy proposed in (Ong, Lim, Zhu & Wong 2006) and the theoretical analysis performed in (Sudholt 2009), almost no work has been done on the structures used in MC. Although some of the ideas successfully applied in MAs can also be extended to modern MC, this research line still presents many unresolved issues. MC being a much broader (and more recent) area than MAs, it is not trivial to perform a general, conceptual analysis of all possible coordination schemes, and their influence on the algorithmic performance. In other words, the definition of a *grammar* of structures (whose syntactic elements are memes) is yet to come. In order to tackle **IRQ IV**, a re-organisation of the concept of the optimisation algorithm and "algorithmic structure" is here given, proposing a general notation to address all the existing algorithms from a MC point of view. The proposed notation introduces the concept of sequence/parallel structure, that will be then implemented in Chapter 7.1 with the name PMS. This implementation provide the software framework for the prototype for the automatic design of optimisers in Chapter 7.2.

Optimisation algorithms can be seen, at an abstract level, as mathematical procedures that address optimisation problems by combining operations of two kinds: search operations where one or more solutions (a population) are generated/computed within the search space, and selection operations, when one or more solutions are selected and retained, see (Neri et al. 2013). This definition is valid regardless of whether the algorithm is an exact or a heuristic method. In general, we can consider all the algorithms as population-based where single-solution algorithms are a special case of them (population size 1). At the generic step t, a set of candidate solutions are stored into a memory structure, namely population $\mathbf{Pop^t}$, which contains the current solutions. In order to improve upon the available candidate solutions, a sub-operator $\mho$ processes the population $\mathbf{Pop^t}$, by applying one/multiple strategies from a given pool, and returns a new population of (trial) candidate solutions $\mathbf{Trials}$ after having applied the search logic of the algorithm:

$$\mathbf{Trials} = \mho\left(\mathbf{Pop^t}\right). \tag{7.1}$$

Subsequently, the selection sub-operator $\mathcal{S}$ processes both the populations $\mathbf{Pop^t}$ and $\mathbf{Trials}$ and returns a new

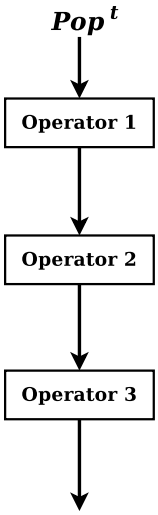


Figure. 7.1. Graphical representation of Sequential Memetic Structure.

current population for the step $t + 1$:

$$\mathbf{Pop^{t+1}} = \mathcal{S}\left(\mathbf{Pop^{t}}, \mathbf{Trials}\right). \tag{7.2}$$

The selection sub-operator $\mathcal{S}$ obviously requires the fitness evaluation of the candidate solutions. For evolutionary algorithms, the above-mentioned formulas are already explicitly formulated as $\mho$ is the application of variation operators (cross-over and mutation) while $\mathcal{S}$ is the selection strategy which can be, for example, the selection of parents and generational replacement for classical GAs or the so called "plus" strategy in ESs. Within an MC view, paired sub-operators $\mho$ and $\mathcal{S}$ (for a certain amount of steps t) compose an algorithmic operator, while optimisation algorithms are seen as connected structures of operators. For the sake of clarity, it must be said that most of the Modern optimisation algorithms employ multiple perturbation logics that could be seen as multiple operators, e.g. EPSDE or MDE-pBX. In order to avoid ambiguity, it is worth stressing that according to the proposed notation, all the basic strategies performed with the aim of providing a new population to be evaluated, are implemented within the $\mho$ sub-operator. In this light, algorithms such EPSDE or MDE-pBX would be forming a single meme, consisting of a single sequence $\mho \rightarrow \mathcal{S}$. LS methods also form a single operator. Even though they often require intermediate steps to work out a new solution, also performing fitness evaluations, all the processing within a single iteration is carried out by $\mho$. Each operator is associated with an input and output $\mathbf{Pop}$, and iterates for a given number of iterations until a stop condition is verified. MC structures can be complex and composed of multiple basic entities, Elementary Memetic Structure (EMS). The first EMS is called *Sequential Memetic Structure* and consists of multiple operators that sequentially process a population $\mathbf{Pop}$, i.e. the output of an operator is univocally the input of the following operator, e.g μDEA, RS, and RIS. Figure 7.1 gives a graphical representation of a sequential structure. Instead of having an unequivocal path, the output population from an operator could be further processed by a decisional component that selects the subsequent operator which will attempt to improve upon $\mathbf{Pop}$. The selection criteria can be based on a probability or a fitness-based rule (e.g. the success of the previous operator). This decisional component is

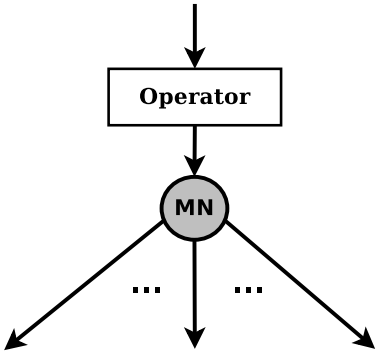


Figure. 7.2. Graphical representation of Memetic Node

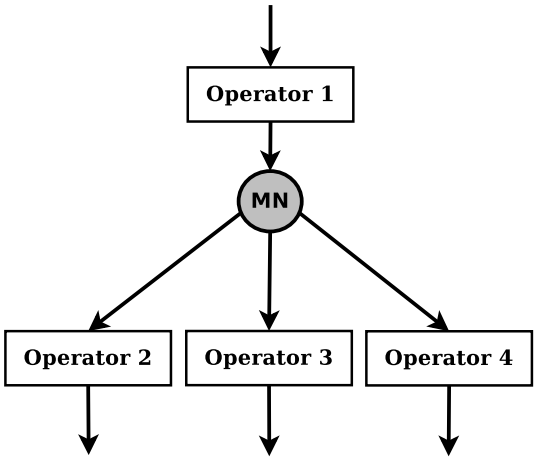


Figure. 7.3. Graphical representation of Parallel Memetic Structure

called Memetic Node (MN) and is defined as a decisional mechanism that allows the selection of one meme (operator) amongst multiple choices, see (Caraffini, Neri, Iacca & Mol 2013). A graphical representation of MN is given in Figure 7.2. Analogous to an electrical node, an MN can be seen as the connection of at least three memes/operators; one is the operator that has just been activated, and two (or more) of them represent the list of memes from which the subsequent operator should be selected. Operators that concur to the same MN are said to be *in parallel*. A sub-structure containing only one MN is here called *Parallel Memetic Structure*, and this is the second EMS here taken into account. Figure 7.3 shows a graphical representation of a Parallel Memetic Structure where operators 2, 3, and 4 are in parallel, and the sub-structure composed of the MN, operators 2, 3, 4 and respective links is a parallel structure. It must be said that the concepts of parallel structure reported here from (Caraffini, Neri, Iacca & Mol 2013), does not refer to the parallel execution of the memes, which are started one at time, but only to the uniqueness of the path, that can be graphically represented like an electric circuit. As in a circuit, where a higher percentage of electrons flow thorough the resistor with the lower resistance, $\mathbf{Pop^t}$ is more likely to undergo a meme rather than another. At the end of the optimisation process the overall flow of functional calls is then re-parted with a different proportion on each branch of the parallel path, but in contrast with electricity the two parallel branches are not run at the same

time, unless the algorithm has specifically been designed for parallel-computing systems (Tan & Zhou 2011). In this light, it is not possible to forecast which meme would next be applied in the optimisation sequence, but it is only possible to draw a static on the activation of each meme (as for example has been done in Chapter 6.2). Circuit-like graphical representation helps us to understand that some structures intrinsically perform an adaptive mechanism, e.g. in 3SOME the last two memes are more likely to be selected over a certain problem due to the improvement made to the fitness, while others systems, allot a fixed probability for activating each branch, thus guaranteeing versatility (e.g. EPSDE-LS). This methodology of representing algorithms has been first defined in (Caraffini, Neri, Iacca & Mol 2013), while proposing a novel algorithm consisting of a single MN, thus the name PMS, which is described in the next section (Chapter 7.1). This optimiser can be seen as an evolution of the 3SOME algorithm deprived of its second stage, which has shown to cover a marginal role (see Chapter 6.2), but empowered with a new component and a new "probabilistic" meme selection strategy, in contrast with the deterministic way that 3SOME uses to sequentially apply each meme. For the sake of clarity, it must be said that the definitions given in (Caraffini, Neri, Iacca & Mol 2013) have then been better formalised in (Caraffini et al. 2014) for every possible MC algorithm, and here further refined with respect to the early studies in a new and more coherent manner. In this light, despite some substantial differences (further details in Chapter 7.1) both 3SOME and PMS share a parallel structure. In fact, see Figure 7.4, 3SOME makes use of a single MN for deciding, according to the actual value of the fitness function after local refinement, which one is the best stage to get in to. Despite the deterministic condition on the decision, is not possible to know the exact sequence of memes activation in advance, since it is not possible to know whether S will improve. In addition, S i more likely to improve upon a certain class of problems, and its performance heavily depends on the initial point. This is provided by a stochastic move performed by the middle exploration stage, so, by looking at the graphical representation it is possible to understand why 3SOME performances are biased on a class of problems. Even though 3SOME intrinsically adapt to the problem, the MN will be able to supervise successfully those problems where M and S perform well, while it won't be able to handle other landscapes, since it is forced to restart over from the first and the second stage, whose exploration is mainly global. Conversely, PMS employs a probabilistic criterion (equal probability for each branch) for selecting a specific LS, making it more robust (in accordance with the NFL theorem). The main difference between the two approaches is that while 3SOME behaviour cannot be altered without changing its structure, those systems relying on a stochastic selection of the operators can be tuned by simply varying a threshold in the MN.

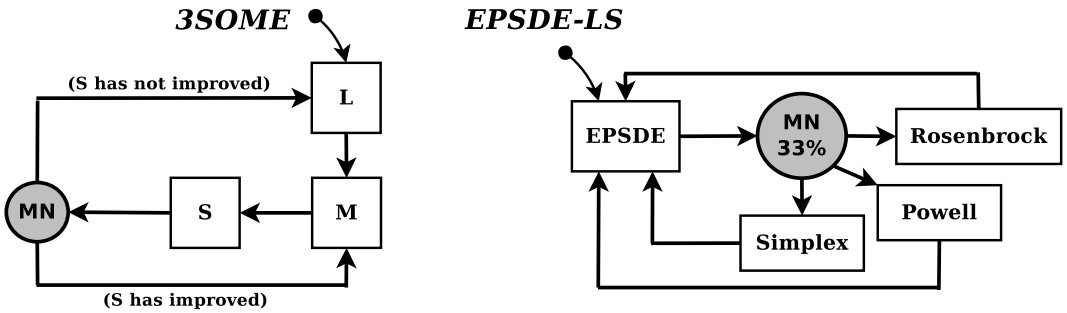


Figure. 7.4. Graphical representation of 3SOME (left) and EPSDE-LS (right) Parallel Memetic Structures.

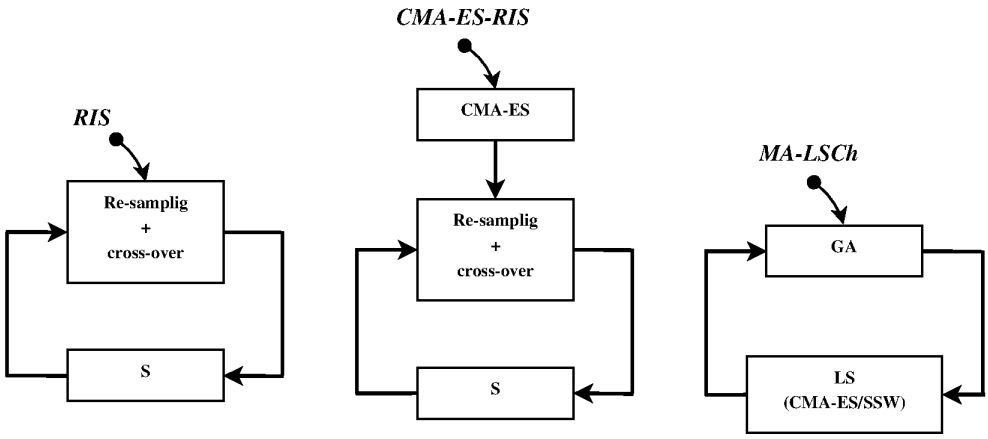


Figure. 7.5. Graphical representation of RIS (left), CMA-ES-RIS (middle) and MA-LSCh (right) Sequential Memetic Structures.

Similarly to PMS, EPSDE-LS also includes a parallel structure, see Figure 7.4. As a counter example, RIS, CMA-ES-RIS and MA-LSCh, Figure 7.5, displays a sequential structure. As a final remark is interesting to notice that a complex algorithm such as MS-CAP also employs a simple sequential structure. This algorithm is in fact based on two rather complex EMS, based on PSO based perturbations (the first one) and multiple DE schemes (the second) but still executed sequentially. In particular, the first meme is executed for a pre-fixed number of functional calls (depending on the dimensionality of the problems) and then further continued according to the stop criterion, i.e. either no improvement has occurred or the maximum budget allowed has expired. So, the current population $\mathbf{Pop^t}$ is being updated with the same sub-operators for a certain amount of time, and then where the computational budget allows a transition, the second meme is run for a given amount of functional calls. There is no ambiguity in the path for this optimiser, which will always alternate two different memes. The same explanation holds for μDEA and VISPO, with the only difference is that while μDEA switches on a simple condition on the budget, VISPO (and MS-CAP) makes its "decision" according to the success on improving upon the fitness value. If an improvement has occurred the same operator keeps iterating, otherwise (stop criterion) there is a transition to the next one. There is no ambiguity on the path and the succession of the operators forms a deterministic sequence. A complete taxonomy considering all the MC approaches presented in this thesis, Chapters 4 and 7, is given in Table 7.12.

7.1 The Parallel Memetic Structure algorithm

The PMS algorithm shares the same motivation explained in (Iacca, Neri, Mininno, Ong & Lim 2012), i.e. the design of a simple MC approach that can compete with the complex population-based algorithms presented in the literature and the definition of an algorithmic structure that can be easily integrated within an automatic designer. The proposed algorithm (as well as that in the following section) have a pure single-solution structure, hence there is no population of solutions involved in the algorithm. Moreover, unlike the EDAs, PMS does not

make use of any explicit probabilistic model of the population. However, it must remarked that as all the meta-heuristics, PMS can be seen as a sample and test procedure. At the machine level, PMS samples points from a probability distribution function which are then improved by deterministic strategies and further tested when their fitness value is calculated. This general scheme is valid all the optimisation algorithm, regardless they are population-based, single-solution, or EDAs. While in 3SOME the attention was mostly focused on the bottom-up logic, so tackling optimisation problems through specialised stages, in PMS a particular care has been given to providing the algorithm with a rich set of diverse search logics, maintaining, at the same time, the algorithmic structure as minimal as possible. As a result, the PMS algorithm still makes use of three memes, but only two stages are considered: the long and short exploration. If in 3SOME the short exploration can be followed by another local refinement, starting from the middle operator, PMS similarly to RIS always performs the two stages sequentially, i.e. the long search is necessarily followed by a local search and then back to the first stage. However, an MC node is here used to switch among two possible LS strategies, leaving the possibility of changing a perturbation move within the neighbourhood of the elite solution, which is refined by the joint action of two operators exchanging $\mathbf{x_{elite}}$ with each other.

In detail, PMS is a combination of the Long Distance Exploration (Algorithm 28, Chapter 4.2), or simply L, the Short Distance Exploration S (Algorithm 5, Chapter 2.2.4) and the Rosenbrock method (Algorithm 6, Chapter 2.2.5), here addressed as R. Briefly, L is a stochastic global search which has the role of detecting interesting areas of the decision space. On the contrary S and R are both exploitative components which attempt to locally improve upon the elite solution. However S and R present very different features. While S performs movements along the axes, R by means of its rotation matrix, performs diagonal perturbations and attempts to follow the gradient of the fitness landscape. In this sense, L can be seen as a global search algorithm while S and R are local search algorithms which cooperatively/competitively exploit the results achieved by L. The two local search logics characterising S and R can be graphically visualised by means of a metaphor: S and R locally explore portions of the decision space like different pieces on a chess board. However, while S behaves like a rook as it performs vertical and horizontal moves, R behaves like a bishop as it moves diagonally. A graphical representation of this metaphor is shown in Figure 7.6. The alternated activation of the two local search logics is supposed to offer a globally robust behaviour, since S and R complete and compensate each other. More specifically, the vertical and horizontal S moves are likely to be an efficient strategy for separable and moderately non-separable functions as they correspond to perturbation along each axis separately. The R action, by exploiting the local gradient can be efficient as a local optimiser on problems that are locally non-separable with high conditioning. However, it must be noted that a strict characterisation of the role of a meme is in practice not always valid. For example, S (or another approach that perturbs the function along the axes separately) can in some cases be more efficient than R on non-separable problems, see e.g. f_{15} in Table 7.2. In consideration of the roles of the memes, the proposed algorithmic structure is organized in the following way. An initial solution is randomly sampled within the decision space $\mathcal{D}$ and given as input for L. According to the description given in Chapter 4.2, the perturbation by L continues until a trial solution outperforms the elite $\mathbf{x_{elite}}$. In order to avoid using all the budget on L an additional stop criterion is included: if L uses 5% of the budget without improving upon $\mathbf{x_{elite}}$, L is stopped and the optimisation is continued by another meme. The elite resulting from L is then processed by an MN to select the subsequent meme. After the MN, S and R have been structured in parallel. In accordance with the Ockham's Razor in MC, the simplest possible MN has

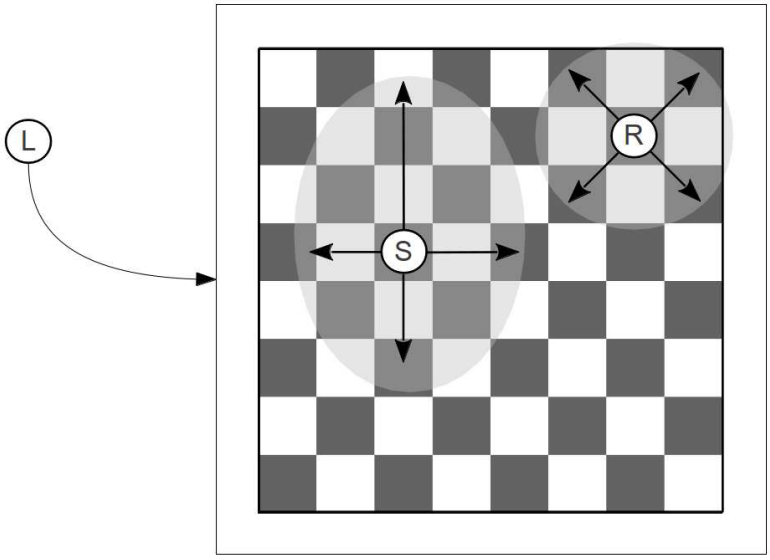


Figure. 7.6. A graphical representation of the roles of the memes in PMS.

been implemented: the solution $\mathbf{x_{elite}}$ is processed either by S or R with an equal probability 0.5. The solution processed by one of the memes in parallel (S or R) is then processed by L anew and the operations above are repeated until the allocated computational budget is used. The structure of PMS is represented by means of pseudo-code in Algorithm 35 and graphically in Figure 7.7. In the graphical representation, the solid line represents the fact that the data flow has only one possible path while the dashed line represents that multiple options are offered to the data flow.

It must be remarked that the proposed PMS could be seen as similar to hyper-heuristics (Burke et al. 2010) as it uses multiple search algorithms coordinated by a (simple) supervisor. However, PMS design does not share with hyper-heuristic the design philosophy. While hyper-heuristics are thought as collections of heterogeneous algorithms selected from a list, PMS, as MC structure, is built up by applying a bottom up logic, see . The PMS algorithm can also be seen as a related structure with respect to the variable neighbourhood search, see (Hansen & Mladenović 2001), (Pérez, Hansen & Mladenovic 2004) and (Hansen, Mladenovi & Moreno Prez 2008) . However, the latter has been specifically designed for combinatorial problem as the concept of neighbourhood take in the discrete space a radically different meaning with respect to what happens in the continuous domain. One specific implementation of variable neighbourhood search for continuous problems exist but it is ultimately a very different algorithm with respect to PMS. Finally, as a member of the MC group, PMS could be seen as a specific implementation of broader concepts and more specifically can be seen as a multimemetic algorithm with population size one and mutation rate 0.5 (Krasnogor & Smith 2005). Nonetheless, the context of PMS is different and its important lays on the fact that it attempts to think of algorithms as structures composed of elementary structures (the parallel would be an elementary structure). This is a fundamental fact to move towards the automatic design of optimisation algorithms.

The PMS algorithm has been run over four whole sets of test problems, namely CEC2005 (30 dimensions), BBOB2010 (100 dimensions), CEC2008 and CEC2010 (both in 1000 dimensions), for a total of 76 test problems considered in this study, as well as being compared against several popular and recent (state-of-

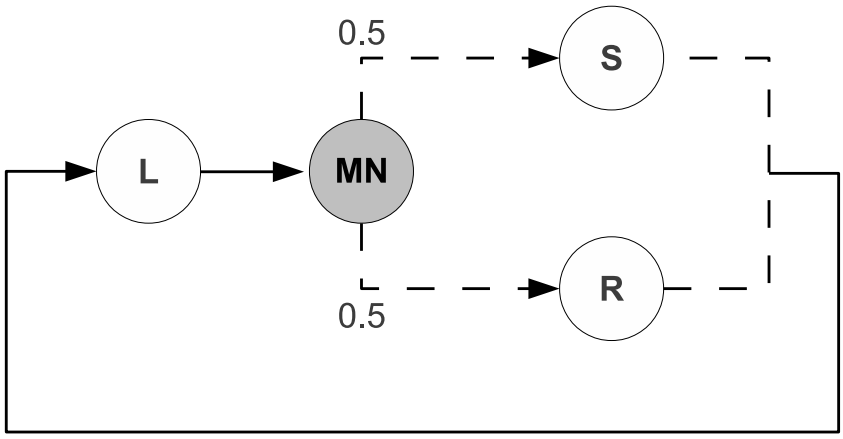


Figure. 7.7. A graphical representation of the PMS algorithms highlighting the parallel structure.

Algorithm 35 Parallel Memetic Structure

$\mathbf{x_{elite}} \leftarrow randomSampling\,(1, n, \mathcal{D})$
while *budget condition* **do**
  */**STAGE 1**/*
  $\mathbf{x_{elite}} \leftarrow \mathrm{L}(\mathbf{x_{elite}})$ ▷ Algorithm 28 with consecutive functional calls limit (5% of the budget)
  */**STAGE 2**/*
  if $\mathcal{U}\,(0, 1) < 0.5$ **then**
    $\mathbf{x_{elite}} \leftarrow \mathrm{S}(\mathbf{x_{elite}})$ ▷ Algorithm 5
  else
    $\mathbf{x_{elite}} \leftarrow \mathrm{R}(\mathbf{x_{elite}})$ ▷ Algorithm 6
  end if
end while
Output $\mathbf{x_{elite}}$

the-art) algorithms. Before commenting on the numerical results, a further test is reported, performed to show the benefits of the parallel structure and the advantages of having two different local components.

7.1.1 The importance of diversity on the chessboard

In this subsection, the intuition that a meme diversity is required to have a high performance is experimentally justified. This aim is pursued by comparing the PMS algorithm with two "degenerate" variants of it: one where MN allows the data flow exclusively to S, let us call it LS, and one where MN allows the data flow exclusively to R, namely LR. Tables 7.1, 7.2, 7.3, and 7.4 show the numerical results of this experiment over the four benchmarks under consideration. Numerical results show that the advantages due to the employment of diverse local search components are not so obvious for the low dimensional problems in this study. On the other hand, PMS appears to offer higher quality performance for large scale problems with respect to its simple variants. More specifically, in 30 dimensions, Table 7.1 shows that performance values are very similar to each other. However, it can be observed that PMS appears more promising than LR and displays a performance similar to that of LS. It must be observed that PMS displays a good performance for highly conditioned functions, such as f_3. Table 7.2 shows that for the 100-dimensional test-bed, PMS confirms its capability to handle functions with a high degree of conditioning (10^6 for f_{11} and f_{12}) and outperforms LS on

Table 7.1. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against LS and LR on CEC2005 in 30 dimensions.

	PMS	LS		LR	
f_1	$\mathbf{-4.50e+02 \pm 6.97e-14}$	$-4.50e+02 \pm 7.97e-14$	=	$-4.50e+02 \pm 3.52e-11$	+
f_2	$-4.50e+02 \pm 3.79e-10$	$\mathbf{-4.50e+02 \pm 2.45e-12}$	=	$-4.50e+02 \pm 6.62e-08$	+
f_3	$\mathbf{1.97e+05 \pm 1.50e+05}$	$2.15e+05 \pm 1.36e+05$	+	$2.10e+05 \pm 1.23e+05$	+
f_4	$2.16e+04 \pm 7.94e+03$	$2.58e+04 \pm 9.76e+03$	+	$\mathbf{9.79e+03 \pm 3.24e+03}$	-
f_5	$1.08e+04 \pm 6.01e+03$	$\mathbf{7.20e+03 \pm 2.33e+03}$	-	$1.55e+04 \pm 3.72e+03$	+
f_6	$\mathbf{5.28e+02 \pm 2.71e+02}$	$7.04e+02 \pm 4.77e+02$	+	$6.59e+02 \pm 3.75e+02$	+
f_7	$-1.80e+02 \pm 1.43e-02$	$\mathbf{-1.80e+02 \pm 1.26e-02}$	=	$-1.80e+02 \pm 1.39e-02$	=
f_8	$-1.20e+02 \pm 1.88e-03$	$\mathbf{-1.20e+02 \pm 9.89e-04}$	=	$-1.20e+02 \pm 6.91e-03$	=
f_9	$-1.18e+02 \pm 5.17e-01$	$-1.18e+02 \pm 7.73e-01$	=	$\mathbf{-1.18e+02 \pm 2.51e-01}$	=
f_{10}	$2.76e+02 \pm 2.45e+01$	$\mathbf{2.69e+02 \pm 2.89e+01}$	-	$2.91e+02 \pm 2.81e+01$	+
f_{11}	$1.23e+02 \pm 5.69e+00$	$\mathbf{1.19e+02 \pm 4.36e+00}$	-	$1.25e+02 \pm 3.64e+00$	+
f_{12}	$1.02e+03 \pm 2.18e+03$	$\mathbf{6.76e+02 \pm 1.53e+03}$	-	$1.53e+03 \pm 2.97e+03$	+
f_{13}	$-1.19e+02 \pm 4.66e+00$	$\mathbf{-1.24e+02 \pm 1.09e+00}$	-	$-1.15e+02 \pm 4.63e+00$	+
f_{14}	$-2.86e+02 \pm 5.60e-01$	$\mathbf{-2.87e+02 \pm 3.06e-01}$	=	$-2.85e+02 \pm 2.58e-01$	=
f_{15}	$1.44e+03 \pm 8.47e-01$	$1.44e+03 \pm 1.36e+00$	=	$\mathbf{1.44e+03 \pm 5.27e-01}$	=
f_{16}	$1.60e+03 \pm 2.52e+01$	$\mathbf{1.59e+03 \pm 1.52e+01}$	=	$1.62e+03 \pm 2.13e+01$	+
f_{17}	$1.67e+03 \pm 1.90e+01$	$1.68e+03 \pm 1.80e+01$	+	$\mathbf{1.64e+03 \pm 1.60e+01}$	-
f_{18}	$9.10e+02 \pm 5.70e-12$	$\mathbf{9.10e+02 \pm 5.75e-12}$	=	$9.10e+02 \pm 3.40e-10$	+
f_{19}	$9.10e+02 \pm 5.63e-12$	$\mathbf{9.10e+02 \pm 5.24e-12}$	=	$9.10e+02 \pm 2.62e-10$	+
f_{20}	$9.10e+02 \pm 5.52e-12$	$\mathbf{9.10e+02 \pm 5.12e-12}$	=	$9.10e+02 \pm 2.02e-10$	+
f_{21}	$1.72e+03 \pm 1.51e+01$	$\mathbf{1.72e+03 \pm 1.27e+01}$	=	$1.73e+03 \pm 9.11e+00$	+
f_{22}	$2.62e+03 \pm 8.11e+01$	$2.63e+03 \pm 7.93e+01$	=	$\mathbf{2.62e+03 \pm 6.85e+01}$	=
f_{23}	$\mathbf{1.72e+03 \pm 1.07e+01}$	$1.73e+03 \pm 1.21e+01$	+	$1.72e+03 \pm 1.11e+01$	=
f_{24}	$1.72e+03 \pm 1.39e+01$	$1.71e+03 \pm 1.32e+01$	=	$\mathbf{1.71e+03 \pm 1.17e+01}$	=
f_{25}	$1.65e+03 \pm 2.55e+02$	$\mathbf{1.61e+03 \pm 2.65e+02}$	-	$1.72e+03 \pm 1.92e+02$	+

Table 7.2. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against LS and LR on BBOB2010 in 100 dimensions.

	PMS	LS		LR	
f_1	$\mathbf{7.95e+01 \pm 3.65e-14}$	$7.95e+01 \pm 3.90e-14$	=	$7.95e+01 \pm 8.14e-13$	=
f_2	$-2.10e+02 \pm 7.13e-14$	$\mathbf{-2.10e+02 \pm 5.37e-14}$	=	$-2.10e+02 \pm 2.21e-09$	+
f_3	$\mathbf{-3.67e+02 \pm 1.66e+01}$	$-3.50e+02 \pm 1.89e+01$	+	$-1.62e+02 \pm 6.15e+01$	+
f_4	$\mathbf{-3.40e+02 \pm 2.05e+01}$	$-3.20e+02 \pm 2.13e+01$	+	$-8.42e+01 \pm 7.15e+01$	+
f_5	$\mathbf{-9.21e+00 \pm 3.51e-12}$	$-9.21e+00 \pm 4.39e-12$	=	$-9.21e+00 \pm 5.94e-12$	=
f_6	$4.05e+01 \pm 4.61e+00$	$\mathbf{3.59e+01 \pm 1.29e-07}$	-	$4.40e+01 \pm 6.60e+00$	+
f_7	$\mathbf{3.70e+02 \pm 8.80e+01}$	$3.88e+02 \pm 1.10e+02$	+	$1.09e+03 \pm 2.08e+02$	+
f_8	$2.21e+02 \pm 5.96e+01$	$\mathbf{1.82e+02 \pm 3.96e+01}$	-	$2.15e+02 \pm 6.08e+01$	-
f_9	$\mathbf{1.72e+02 \pm 2.30e+01}$	$1.86e+02 \pm 3.16e+01$	+	$1.74e+02 \pm 1.29e+01$	=
f_{10}	$2.37e+03 \pm 6.64e+02$	$3.22e+03 \pm 6.91e+02$	+	$\mathbf{2.17e+03 \pm 6.19e+02}$	-
f_{11}	$\mathbf{6.51e+02 \pm 8.17e+01}$	$7.32e+02 \pm 8.70e+01$	+	$6.83e+02 \pm 7.76e+01$	+
f_{12}	$-6.11e+02 \pm 1.52e+01$	$\mathbf{-6.18e+02 \pm 4.24e+00}$	-	$-6.08e+02 \pm 2.23e+01$	=
f_{13}	$3.60e+01 \pm 6.01e+00$	$3.64e+01 \pm 4.68e+00$	=	$\mathbf{3.37e+01 \pm 5.08e+00}$	-
f_{14}	$-5.23e+01 \pm 1.80e-05$	$-5.23e+01 \pm 5.39e-05$	=	$\mathbf{-5.23e+01 \pm 1.65e-05}$	=
f_{15}	$2.39e+03 \pm 6.32e+02$	$\mathbf{2.22e+03 \pm 2.38e+02}$	=	$4.36e+03 \pm 4.36e+02$	+
f_{16}	$9.23e+01 \pm 9.50e+00$	$\mathbf{8.83e+01 \pm 3.55e+00}$	=	$1.31e+02 \pm 7.97e+00$	+
f_{17}	$-1.96e+00 \pm 9.61e+00$	$\mathbf{-8.79e+00 \pm 1.60e+00}$	-	$1.08e+02 \pm 4.63e+00$	+
f_{18}	$4.89e+01 \pm 4.41e+01$	$\mathbf{1.58e+01 \pm 6.48e+00}$	-	$8.17e+01 \pm 3.15e+01$	+
f_{19}	$-5.08e+01 \pm 4.81e+01$	$\mathbf{-9.30e+01 \pm 2.21e+00}$	-	$3.21e+00 \pm 2.18e+01$	+
f_{20}	$-5.45e+02 \pm 1.28e-01$	$\mathbf{-5.45e+02 \pm 1.03e-01}$	=	$-5.45e+02 \pm 1.28e-01$	=
f_{21}	$\mathbf{5.05e+01 \pm 1.12e+01}$	$5.10e+01 \pm 8.90e+00$	=	$5.44e+01 \pm 1.42e+01$	+
f_{22}	$-9.83e+02 \pm 1.29e+01$	$\mathbf{-9.85e+02 \pm 1.45e+01}$	=	$-9.83e+02 \pm 1.35e+01$	=
f_{23}	$8.70e+00 \pm 8.16e-01$	$\mathbf{8.27e+00 \pm 5.46e-01}$	-	$9.32e+00 \pm 6.52e-01$	+
f_{24}	$2.14e+03 \pm 6.37e+02$	$\mathbf{1.75e+03 \pm 3.83e+02}$	-	$2.63e+03 \pm 2.85e+02$	+

separable functions (f_1 to f_5). Conversely, LS appears to be more effective on multi-modal problems with a weak global structure. In 1000 dimensions, PMS appears to outperform both its variants.

In addition, PMS has been compared with its predecessor 3SOME and two popular population based meta-heuristics, namely CLPSO and JADE. Moreover, a final comparison has been carried out against recent

Table 7.3. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference= PMS) for PMS against LR and LR on CEC2008 in 1000 dimensions.

	PMS	LS		LR	
f_1	$\mathbf{-4.50e+02 \pm 4.20e-13}$	$-4.50e+02 \pm 2.64e-07$	+	$-4.50e+02 \pm 6.64e-13$	+
f_2	$-3.88e+02 \pm 5.17e+01$	$\mathbf{-4.50e+02 \pm 4.46e-02}$	-	$-2.82e+02 \pm 3.58e+00$	+
f_3	$\mathbf{1.34e+03 \pm 5.37e+02}$	$1.40e+03 \pm 1.02e+02$	+	$1.40e+03 \pm 5.46e+02$	+
f_4	$\mathbf{-3.30e+02 \pm 1.33e-12}$	$-3.30e+02 \pm 1.52e-05$	+	$1.03e+04 \pm 4.45e+02$	+
f_5	$1.80e+02 \pm 1.50e-02$	$\mathbf{-1.80e+02 \pm 8.07e-03}$	=	$-1.80e+02 \pm 3.55e-02$	=
f_6	$-1.38e+02 \pm 6.36e+00$	$\mathbf{-1.40e+02 \pm 5.90e-06}$	-	$-1.20e+02 \pm 1.82e-02$	+
f_7	$-1.37e+04 \pm 2.80e+02$	$\mathbf{-1.40e+04 \pm 4.82e+01}$	-	$-1.22e+04 \pm 1.19e+02$	+

Table 7.4. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against LS an LR on CEC2010 in 1000 dimensions.

	PMS	LS		LR	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$2.45e-12 \pm 2.21e-12$	+	$4.84e-12 \pm 2.54e-11$	+
f_2	$\mathbf{1.48e-13 \pm 2.79e-13}$	$4.05e-05 \pm 2.99e-05$	+	$1.00e+04 \pm 4.24e+02$	+
f_3	$4.52e-01 \pm 2.76e+00$	$\mathbf{5.89e-05 \pm 7.43e-06}$	-	$1.98e+01 \pm 1.65e-02$	+
f_4	$5.27e+11 \pm 2.72e+11$	$7.73e+13 \pm 2.63e+13$	+	$\mathbf{4.39e+11 \pm 2.61e+11}$	-
f_5	$4.78e+08 \pm 1.40e+08$	$\mathbf{4.13e+08 \pm 9.35e+07}$	-	$6.15e+08 \pm 1.38e+08$	+
f_6	$1.92e+07 \pm 2.24e+06$	$\mathbf{7.15e+06 \pm 6.04e+06}$	-	$1.98e+07 \pm 9.29e+04$	+
f_7	$1.02e+08 \pm 2.57e+08$	$1.85e+10 \pm 6.71e+09$	+	$\mathbf{9.01e+07 \pm 2.28e+08}$	-
f_8	$1.17e+08 \pm 1.30e+08$	$1.91e+09 \pm 1.89e+09$	+	$\mathbf{1.08e+08 \pm 1.23e+08}$	-
f_9	$6.19e+06 \pm 2.80e+06$	$1.93e+08 \pm 2.92e+07$	+	$\mathbf{4.55e+06 \pm 8.03e+05}$	-
f_{10}	$5.25e+03 \pm 2.09e+03$	$\mathbf{3.52e+03 \pm 2.09e+02}$	-	$1.31e+04 \pm 4.74e+02$	+
f_{11}	$1.85e+02 \pm 3.03e+01$	$\mathbf{1.29e+02 \pm 5.17e+01}$	-	$2.18e+02 \pm 2.40e-01$	+
f_{12}	$\mathbf{1.06e+03 \pm 7.23e+02}$	$4.37e+04 \pm 1.03e+04$	+	$1.39e+03 \pm 7.74e+02$	+
f_{13}	$\mathbf{1.18e+03 \pm 6.33e+02}$	$3.73e+03 \pm 2.62e+03$	+	$1.42e+03 \pm 6.37e+02$	+
f_{14}	$1.44e+07 \pm 5.47e+06$	$5.95e+07 \pm 2.60e+06$	+	$\mathbf{9.41e+06 \pm 9.85e+05}$	-
f_{15}	$1.20e+04 \pm 3.87e+03$	$\mathbf{7.33e+03 \pm 3.41e+02}$	-	$1.52e+04 \pm 5.14e+02$	+
f_{16}	$3.27e+02 \pm 9.13e+01$	$\mathbf{1.14e+02 \pm 4.67e+01}$	-	$3.97e+02 \pm 3.07e-01$	+
f_{17}	$1.37e+03 \pm 8.23e+02$	$3.33e+04 \pm 6.64e+03$	+	$\mathbf{1.36e+03 \pm 1.17e+03}$	=
f_{18}	$2.41e+03 \pm 1.00e+03$	$\mathbf{2.07e+03 \pm 3.08e+03}$	-	$2.64e+03 \pm 8.21e+02$	+
f_{19}	$1.47e+05 \pm 4.93e+04$	$2.31e+06 \pm 2.24e+05$	+	$\mathbf{1.25e+05 \pm 1.98e+04}$	-
f_{20}	$\mathbf{9.57e+02 \pm 5.37e+02}$	$1.05e+03 \pm 1.69e+02$	+	$1.07e+03 \pm 5.41e+02$	+

algorithms, representing the state-of-the-art of modern optimisation:CCPSO2, MA-LSCh and MDE-pBX. Due to limitations with space, large tables have been reported in Appendix E.4. Numerical results against popular algorithms show that in 30, Table E.24, and 100 dimensions, Table E.25, PMS offers a competitive performance with respect to 3SOME, CLPSO and JADE. More specifically, PMS for both these two dimensionality values displays a performance whose quality is very similar to that of 3SOME. The comparison with CLPSO shows that PMS is slightly more promising over CEC2005 and slightly less promising on BBOB2010. Conversely, PMS tends to be slightly outperformed by JADE for the problems in CEC2005 while it outperforms JADE for those in BBOB2010. It must be said that both CLPSO and JADE were originally designed and tested to tackle problems similar to those in the two test-beds considered in Tables E.24 and E.25. For the high dimensional problems, PMS seems to be slightly outperformed by 3SOME for the test problems in Table E.26. In all the other cases (i.e. the majority) PMS appears to offer an extraordinarily high performance. This fact can be seen from Tables E.26 and E.27, where it is shown that PMS tends to regularly outperform the popular meta-heuristics used for comparison in this study. Numerical results obtained with the state-of-the-art algorithms show that for low dimensional problems (Table E.28 in 30 and Table E.29 for 100 dimensions), PMS is competitive with CCPSO2 and MDE-pBX. On the contrary, for these benchmarks MA-LSCh-CMA appears to have a superior performance with respect to the other algorithms considered in this paper. The fact that CCPSO2 does not show a very high performance in low dimensions was expected as it was specifically designed

Table 7.5. Holm-Bonferroni procedure on the algorithms under consideration (reference algorithm: PMS, Rank= 5.75).

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	CCPSO2	5.8	1.18e-01	5.47e-01	5.00e-02	Accepted
2	LS	5.71	-8.89e-02	4.65e-01	2.50e-02	Accepted
3	3SOME	5.26	-1.10e+00	1.37e-01	1.67e-02	Accepted
4	MA-LSCh	5.22	-1.18e+00	1.18e-01	1.25e-02	Accepted
5	MDE-pBX	4.78	-2.19e+00	1.42e-02	1.00e-02	Accepted
6	LR	4.32	-3.23e+00	6.23e-04	8.33e-03	Rejected
7	CLPSO	4.28	-3.32e+00	4.55e-04	7.14e-03	Rejected
8	JADE	3.9	-4.21e+00	1.30e-05	6.25e-03	Rejected

for large scale problems. On the contrary, MDE-pBX and MA-LSCh-CMA were designed for relatively low dimensional problems. As a consequence, the latter two algorithms (especially MA-LSCh-CMA) display a good performance. For the set of large scale problems, the ranking among these algorithms dramatically changes. In 1000 dimensions MDE-pBX and MA-LSCh-SSW display a performance that is not so good when compared to that of PMS and CCPSO2. The comparison between PMS and CCPSO2 shows that these two algorithms perform equally well over the set of problems of CEC2008 while, in the other test-bed, CEC2010, PMS outperforms all the algorithms including CCPSO2. According to these results, the two local search components contained in the PMS exploit the decision space and achieve, by means of different search logics, interesting areas of multi-variate fitness landscapes. The benefit of these components is evident in large scale problems as an exploitative action that allows fast improvement of the candidate solutions. On the other hand, for low dimensional problems, population-based systems appear to better explore the decision space and thus allow a more efficient detection of solutions that are located close to the optimum. Finally, this study confirms that, while as a general rule population-based systems containing multiple local search activations are capable of efficiently tackling optimisation problems, seeing their implementation in high dimensions may require an excessive effort in terms of computational resources and computational budget before high quality solutions are detected. With reference to Table 7.5, it is possible to have a more global view over the full set of 76 problems considered in this study thanks to the Holm-Bonferroni procedure. A first consideration is that PMS displays a better rank than LR and LS. This fact confirms the intuition about diversity of local search moves in parallel structures with multiple memes (diversity of pieces on a chessboard) as represented in Figure 7.6. At a general level, the idea that a proper combination of diverse memes can outperform each of the single memes making up this combination has already been mentioned in the literature, e.g. in (Krasnogor 2004) and (Hart, Krasnogor & Smith 2005). This case study shows how the combination of clearly different search strategies can lead to a robust structure displaying a high performance. As shown, PMS, despite its simplicity, is ranked as the second best algorithm in this study. It must be observed that PMS displays an overall better performance with respect to both classical and state-of-the-art algorithms. Although PMS statistically outperforms only CLPSO and JADE, it appears to be more promising than the modern and complex algorithms based on populations and multiple local search activations, such as MA-LSCh and MDE-pBX. The comparison between PMS and CCPSO2 shows interesting discussion potentials. CCPSO2 is ranked slightly better than PMS. However, CCPSO2 tends to have a slightly better performance than PMS in low dimensions and slightly worse in high dimensions. This finding is important because the variable decomposition of CCPSO2 was originally designed for large scale problems to decompose the dimensionality of the problem. Although efficient in high dimensions, this operation is likely to be suboptimal since it can be outperformed by a much simpler single solution structure. On the other hand, this

operation appears to be efficient in low dimensions. Further study could consider a meme that decomposes the dimensionality of the problem. The success of PMS for large scale problems can be justified when considering that the size of a search space grows exponentially with the amount of variables, see (Caponio, Kononova & Neri 2010). In other words, in a high dimensional space, given an initial candidate solution, the chance of improving upon this solution is much higher than low dimensional cases. In this light, exploitative schemes tend to be more efficient in high dimensions than algorithms that attempt to perform an extensive exploration. Since PMS perturbs only a single solution, it can be seen as an exploitative search. On the other hand, small populations (and thus also single solution algorithms) suffer from premature convergence, or, more generally, are likely to be unable to detect new promising solutions after an initial success. This effect is usually evident for large scale problems where the high dimensionality can be a challenge for the search operators. The power of PMS is the employment of two diverse operators for exploring the landscapes from different perspectives, see Figure 7.6, together with a global search that essentially restarts the search without a massive exploration. The features of high exploitation, diversity of operators, and a non-destructive restarting mechanism (part of the old solution is inherited by the output of L) seem to be the key-points for an algorithmic success in high dimensions.

In general, the numerical results reported for the 76 test problems belonging to four popular test beds, as well as the real-world application in (Caraffini, Neri, Iacca & Mol 2013), show that PMS is an extremely simple structure (in accordance with Ockham's Razor in MC) that can display a performance that is as good, if not better, than complex modern algorithms. This preliminary set of results on parallel structures opens a perspective for further studies, where other memes are considered, as well as adaptive data flow distribution within an MN. The latter aspect is investigated in the following section.

7.2 Towards an automatic design: an analysis on separability

The topic of automatic design of optimisation algorithms is currently intensively discussed within the MC community, see (Meuth et al. 2009) and (Zhu et al. 2010), as well as in related fields where similar concepts and issues are analysed from slightly different perspectives, see e.g. (Wu et al. 2012) and (Ren et al. 2012) in the field of hyper-heuristics. However, due to its complexity, an efficient and generally valid solution has not yet been found and, perhaps, will still require several years of research in mathematics and computer science. As discussed in Chapter 4, and in the previous sections, MC appears to be particularly prone, because of its methodology of combining operators in memetic structures capable of problem solving, to providing a potential tool for the automatic design of optimisation algorithms. To some extent, some of the works previously described in this thesis, such as (Iacca, Neri, Mininno, Ong & Lim 2012) and (Caraffini, Neri, Iacca & Mol 2013), represent the first trial steps in this direction. The latest, more significant, step was then made in (Caraffini et al. 2014), providing the implementation of a fully functional computational prototype for continuous optimisation problems. The general proposed procedure consists of two phases. At first a problem analyser detects the features of the problem. Subsequently these features are used to select the operators and

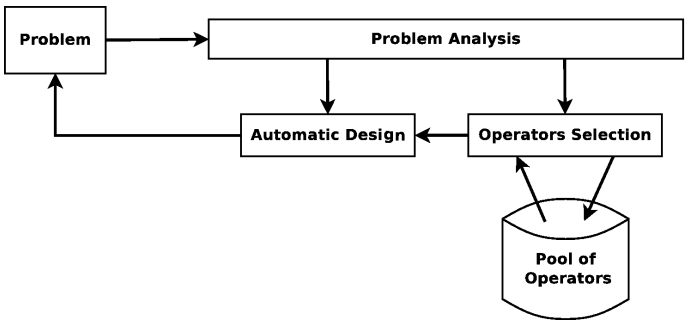


Figure. 7.8. Graphical representation of the general idea of automatic design platform with problem analyser

their links, thus performing the algorithmic design automatically. The implemented problem analyser estimates the separability of the problem and extracts an index indicating the so-called *degree of separability*, see Chapter 7.2.1. This index is then used to balance the action of an operator that moves along the axes with respect to another operator that performs diagonal moves by estimating the local gradient. The proposed implementation will be referred to here as the SPAM. Being a prototype, only one feature (separability) amongst other important landscape properties, e.g. dimensionality, ill-conditioning, multi-modality, has been investigated. Nonetheless, more testing is already in progress, and considering the results obtained in (Caraffini et al. 2014), this framework seems to be promising and easily extendible. Finally, it must be said that this topic has a major impact within the entire AI field in the mid and long term. More specifically, a machine that is able to "understand" what is the suitable solver in every circumstance is a machine capable of critically making decisions. In other words, since intelligence can be seen as the talent of performing the "right" choice on a regular basis, a machine that builds up the reasoning to perform the best choice by itself has a level of automation superior to that currently present in computational machines.

Before entering into the implementation details of the SPAM algorithm, a more detailed description of the general idea and software platform is given. The algorithmic design is automatically performed by the machine after a problem analysis. More specifically, an initial portion of the budget is used to analyse the problem and extract its features. For each feature, an index will be assigned. These indices will then be used for selecting and combining those operators that, according to the algorithmic designer, perform an action that addresses the problem features detected during the problem analysis. A graphical representation of the general idea of the automatic design platform is given in Figure 7.8. The present implementation is a prototype restricted to the separability, forming the first component of the problem analyser.

7.2.1 Separability Analysis

A function f of n independent variables is said to be separable if it can be expressed as a sum of n functions, each of them depending on one variable only. This definition implies that, from an optimisation point of view, separable functions are relatively easy to handle as the optimisation problem in n variables can be tackled

efficiently by separately perturbing each variable. However, real-world applications are often (if not always) characterized by non-separable fitness functions, i.e. functions in which there is some degree of non-linear inter-variable interaction. In contrast, non-separable problems cannot be solved by performing moves along the axes, but require simultaneous perturbations of multiple variables (diagonal moves). Whilst, according to rigorous mathematics, an interaction between a pair of variables only is enough to make the problem non-separable, in computational science this is not entirely true. According to the number of interacting variables, a function (and thus the associated problem) can be considered separable to a certain degree, being e.g. fully separable, moderately separable, moderately non-separable, or fully non-separable. Modern test beds tend to classify test problems according to similar criteria, e.g. see (Hansen, Auger, Finck, Ros et al. 2010). On the basis of this consideration, an estimation of the degree of separability (in a fuzzy logic fashion) has been defined here. A set of λ candidate solutions is sampled within $\mathbf{D}$. For a limited portion of the budget the CMA-ES with rank-μ-update and weighted recombination, see (Hansen et al. 2003), is applied. A detailed description of this optimiser has been given in Chapter 3.1.2.1, providing the mathematical background and implementation details. For the sake of clarity, a brief reminder is given here about the working principle of CMA-ES. The main idea behind this algorithm is to sample λ points from a multivariate distribution, compute their fitness values and update the shape of the distribution in order to (locally) progressively adapt to the problem. After a certain amount of generations, the matrix $\mathbf{C}$ evolves and reliably approximates the (theoretical) covariance matrix. A covariance matrix is a correlation matrix, i.e. a matrix that describes the correlation between pairs of variables. For the estimated covariance matrix $\mathbf{C}$, indicating the generic term of the matrix $C_{i,j}$, the following matrix transformation is applied:

$$\rho_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i}C_{j,j}}}. \tag{7.3}$$

The operation in Equation 8.2 describes the Pearson correlation coefficient and the matrix ρ is the Pearson correlation matrix. These coefficients vary between -1 and 1 and measure the linear correlation between pairs of variables. When $\rho_{i,j} = 0$ there is no correlation at all between the i^{th} and j^{th} variables. When $|\rho_{i,j}| = 1$ there is a perfect correlation between the variables. More specifically, when $\rho_{i,j} = 1$ it means that an increase of the i^{th} variable corresponds to the same (linear) increase of the j^{th} variable, when $\rho_{i,j} = -1$, it means that an increase of the i^{th} variable corresponds to the same (linear) decrease of the j^{th} variable. The Pearson correlation matrix has been chosen instead of the covariance matrix directly because its elements are limited and normalized within the $[-1, 1]$ interval, and thus allow an immediate interpretation for the purposes of this problem analyser. If there is no correlation between any pair of variables, the corresponding optimisation problem can be solved by perturbing each variable separately. Conversely, if the variables are correlated, search moves in optimisation require a simultaneous perturbation of multiple variables. Thus, as shown in (Lin & Cheng 2011), although there is no rigorous mathematical equivalence, the Pearson correlation matrix ρ can be viewed as a description of the separability of the optimisation problem. In order to use this description with the aim of designing an algorithm, the absolute value of the Pearson correlation matrix $|\rho|$ is computed, as there is no interest in distinguishing between positive and negative correlation, since they would both result in the application of a simultaneous perturbation of the variables. The resulting matrix is symmetrical and exhibits

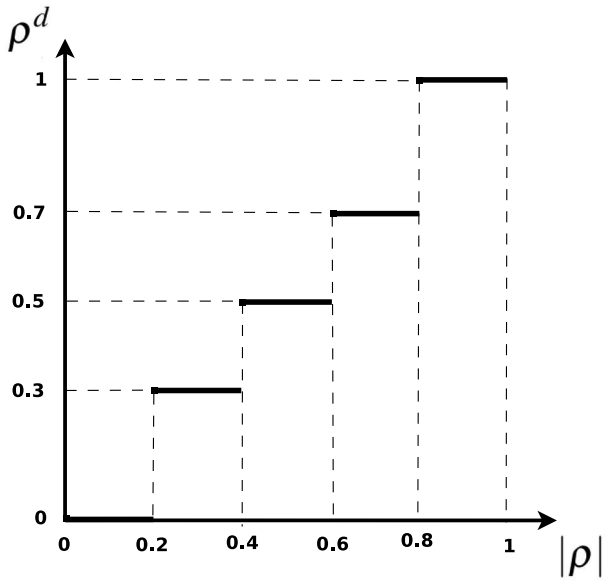


Figure. 7.9. Rounding procedure of the correlation coefficients

ones on the diagonal:

$$|\rho| = \begin{vmatrix} 1 & |\rho_{1,2}| & |\rho_{1,3}| & ... & |\rho_{1,n}| \\ X & 1 & |\rho_{2,3}| & ... & |\rho_{2,n}| \\ X & X & 1 & ... & |\rho_{3,n}| \\ ... & ... & ... & ... & ... \\ X & X & X & X & 1 \end{vmatrix}.$$

Thus, only $\frac{(n^2-n)}{2}$ elements of the matrix $|\rho|$ are of interest for evaluating the degree of separability of the problem. In order to extract an index that estimates the separability of a problem, an average of the elements of the matrix $|\rho|$ is computed. However, since the initial budget to evolve the matrix $\mathbf{C}$ is limited, the Pearson correlation matrix normally contains approximation errors. These errors could jeopardise the reliability of the separability index. In order to mitigate this effect, the matrix $|\rho|$ is processed by a rounding procedure that approximates a value $|\rho_{i,j}| \in [0, 0.2[$ with 0, $|\rho_{i,j}| \in [0.2, 0.4[$ with 0.3, $|\rho_{i,j}| \in [0.4, 0.6[$ with 0.5 , $|\rho_{i,j}| \in [0.6, 0.8[$ with 0.7, and $|\rho_{i,j}| \in [0.8, 1]$ with 1. The discrete values resulting from this process are indicated as $\rho^d_{i,j}$. A graphical scheme representing this rounding procedure is shown in Figure 7.9. Finally, the average of these values is calculated:

$$\varsigma = \frac{2}{(n^2-n)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \rho^d_{i,j}. \tag{7.4}$$

Thus, the separability index ς is an estimation of the degree of separability of the optimisation problem under examination. When this index $\varsigma = 0$, the problem is considered fully separable; when $\varsigma = 1$, the problem is considered fully non-separable. In the remaining cases the problem is considered to have intermediate features. In order to experimentally demonstrate the validity of the proposed approach. The popular CEC2005 test-bed

Table 7.6. Separability Index (ς) for CEC2005 in 30 dimensions.

	Separability Index (ς)	Separable
f_1	0.043	YES
f_2	0.080	–
f_3	0.399	–
f_4	0.189	–
f_5	0.226	–
f_6	0.104	–
f_7	0.067	–
f_8	0.195	–
f_9	0.058	YES
f_{10}	0.257	–
f_{11}	0.183	–
f_{12}	0.672	–
f_{13}	0.404	–
f_{14}	0.510	–
f_{15}	0.046	YES
f_{16}	0.121	–
f_{17}	0.186	–
f_{18}	0.164	–
f_{19}	0.143	–
f_{20}	0.165	–
f_{21}	0.073	–
f_{22}	0.175	–
f_{23}	0.155	–
f_{24}	0.050	–
f_{25}	0.190	–

in 30 dimensions, see (Suganthan et al. 2005), has been considered. For each problem contained in the test-bed, the problem analysis described above has been performed and the separability index ς has been calculated. Table 7.6 displays, for each optimisation problem, the corresponding ς value and the indication on separability originally reported in (Suganthan et al. 2005). As shown in Table 7.6, the proposed coefficient ς reliably estimates the separability of problems f_1, f_9, and f_{15}. In the remaining cases, the value of the index ς is on average larger than in the separable cases. In addition it can be seen that the separability degree varies over the problem. While in the case of a sphere (f_1) the problem is clearly separable and thus the index $\varsigma \approx 0$, in other cases, such as f_2 the problem (Shifted Schwefels Problem 1.2) is only weakly non-separable. It can be observed that in f_2 it is mathematically non-separable, but can still be optimised by taking into consideration the variables one-by-one. More specifically, the mathematical expression of f_2 is (apart from shift and bias) of the form $\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$. Although being non-separable, this problem can be solved by optimising at first x_1^2 (and obtaining 0), then substituting in $x_1^2 + x_2^2$ and solving with respect to x_2 and so on. If we are under the hypothesis that the problem is a black box and thus we ignore its analytical properties, this problem greatly benefits from an algorithm that perturbs the variables one by one. In other words, from an optimisation perspective this problem is "almost separable". Another interesting case is f_{24}, that albeit being a complex composition function, it appears to be nearly separable. While it is not easy to show that this function can be optimised by considering the variables separately, numerical results will show that the application of search moves along the axes lead to an excellent performance.

In order to formalise the notation used, the analysis of separability can be seen as an operator, a meme, whose stop condition is set to 20% of the entire computational budget. In term of software implementation,

it can be seen as a routine, SEPARABILITYANALYSER(), returning the elite solution $\mathbf{x_{elite}}$ (this portion of the budget is not taken off the optimisation process, the analyser also performs optimisation) and the separability index ς. At this stage of this study, the SEPARABILITYANALYSER() routine implements a CMA-ES, since it is one of the most elegant (and based on a solid mathematical background) way to estimate a covariance matrix, while performing an efficient optimisation. However, this routine is kept general on purpose, since it could be subject to variation in the future. In particular, some research is already trying to replace CMA-ES with a lighter method. A potential replacement could be represented by a Linkage-Learning mechanism, see (Chen 2012). However, despite CMA-ES being the bottleneck of SPAM in term of algorithmic complexity, i.e. inadequate temporal overhead and memory footprint, its excellent performance and accuracy on approximating the theoretical covariance matrix make it difficult to replace.

7.2.2 The pool of operators

The SPAM algorithm makes use of a pool of two operators. The same operators used in (Caraffini, Neri, Iacca & Mol 2013) for the PMS algorithm have also been considered in this work. In accordance with the Ockham's Razor in MC, the old memes have been selected in order not to complicate the first implementation with unnecessary operators, before having tested the prototype. So, as already done for PMS, in the next session Algorithm 5 will be referred to, as usual, with the name S, while Algorithm 6 with R. As widely discussed before, S moves along the axes and attempts to search for fitness improvements by perturbing the variables one by one, R perturbs all the variables at the same time by following the local gradient. Clearly, S and R are here employed for handling separable and non-separable problems, respectively. For this reason, they are particularly suitable for SPAM. Nonetheless, the pool of operators is a part of the landscape analyser depicted in Figure 7.8, and so is supposed to be enriched with more operators in the future, chosen ad-hoc in order to tackle other features, e.g. ill-conditioning.

7.2.3 Automatic Memetic Design

When the index ς is available, the automatic design is performed. The SPAM algorithm makes use of a Parallel Memetic Structure where two operators in parallel alternatively perturb a single solution. The best solution ever found after the application of each operator is here indicated as *elite* $\mathbf{x_{elite}}$. In spite of PMS, in this case the elite does not necessarily undergo refinement through each meme. So, with reference to the general notation given in this chapter, a generic solution $\mathbf{x_s}$ plays the role of the population $\mathbf{Pop}$ (a population of only one individual at this stage, while the separability analysis requires a proper population), while the $\mathbf{Trials}$ is the trial solution internally generated within S and R in order to improve upon $\mathbf{x_s}$. The initial solution is the best solution detected during the problem analysis, i.e. the best solution obtained by CMA-ES with a limited budget. The index ς is used to assign, for each problem, an activation probability within the MN. If $\varsigma = 0$, the problem is considered separable and only S is repeatedly used to optimise the function. If ς is high (i.e.

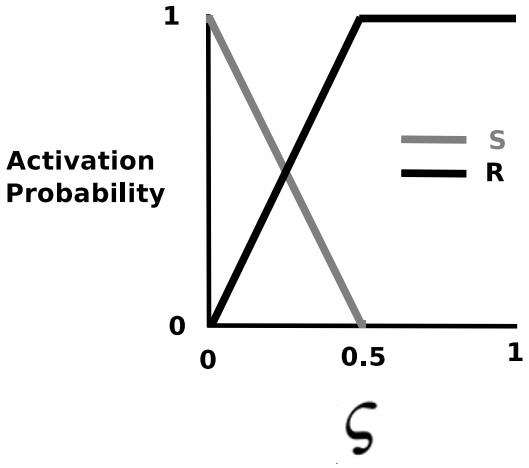


Figure. 7.10. Activation probabilities of the operators.

$\varsigma > 0.5$ according to the observations of the coefficient in preliminary experiments using CEC2005 in 10, 30 and 50 dimensionality values), the problem is considered fully non-separable and only R is repeatedly used to optimise the problem. In the remaining cases, the problem is considered to have intermediate features and the two operators coexist (or compete and collaborate). In the latter case, the assignment of the two probabilities are given by a linear trend. Figure 7.10 illustrates the activation probabilities of the two trends, which can be simply modelled by two functions of ς, namely $\Psi_S : [0,1] \longrightarrow [0,1]$ and $\Psi_R : [0,1] \longrightarrow [0,1]$, returning the corresponding probability as follow:

$$\Psi_R(\varsigma) = 1 - \Psi_S(\varsigma) = \begin{cases} 2 \cdot \varsigma & if\ \varsigma \in [0, 0.5] \\ 1 & if\ \varsigma \in (0.5, 1] \end{cases} \tag{7.5}$$

Finally, due to the deterministic nature of both S and R, in order to avoid that operator is called to perform the same steps in vain, a control has been implemented. In particular, LS deterministic routines performance heavily depends on the applied start point, and so, a restart procedure (similar to that one successfully used in RIS, Chapter 4) represents the best option to introduce a certain degree of randomness on the choice of the initial point. This is essential in those undesired cases where the same operator that has failed to improve upon a given solution, is selected again. This scenario is to avoid, in particular when $\varsigma = 1$ or 0, since the same logic would be iteratively necessarily selected over the same starting point, getting stuck in potential suboptimal solutions. For this reason, $\mathbf{x_s}$ holds the same value of the elite solution until the first failure on the application of the LS, i.e. $\mathbf{x_s}$ undergoes local refinement without any improvement on its fitness value, and in this case is regenerated according the logic in Algorithm 36. More specifically, at first a new trial solution $\mathbf{x_r}$ is randomly sampled within $\mathbf{D}$, and then mated with the elite solution via exponential cross-over, Algorithm 19. The newly generated point becomes the new solution, i.e. $\mathbf{x_s}^{t+1} = \mathcal{S}\left(\mathbf{x_s}^t, \mathbf{x_{trial}}\right)$, and, if its fitness value outperforms that one of the elite, $\mathbf{x_{elite}}$, is accordingly updated. This mechanism can be seen as a third meme using only 1 functional call, i.e. one random sampling followed by the exponential cross-over, which is activated by a second MN only when no improvement upon $\mathbf{x_s}$ has been detected. It must be noted that this

Algorithm 36 Separability Prototype for Automatic Memes

$[\varsigma, \mathbf{x_{elite}}] \leftarrow$SEPARABILITYANALYSER() ▷ 20% of the total computation budget
$\mathbf{x_s} \leftarrow \mathbf{x_{elite}}$
while *remaining budget available* **do**
  $P \leftarrow \mathcal{U}(0,1)$
  if $P > \Psi_R$ **then** ▷ Equation 7.5
    $\mathbf{x_s} \leftarrow$S($\mathbf{x_s}$) ▷ Algorithm 5
  else
    $\mathbf{x_s} \leftarrow$R($\mathbf{x_s}$) ▷ Algorithm 6
  end if
end while
if *if LS has not improved upon* $\mathbf{x_s}$ **then**
  $\mathbf{x_s} \leftarrow randomSampling\,(1, n, \mathcal{D})$
  $\mathbf{x_s} \leftarrow$XOVEREXP($\mathbf{x_{elite}}, \mathbf{x_s}$) ▷ Algorithm 19, CR calculated as in Formula 3.22
end if
if $f(\mathbf{x_s}) < f(\mathbf{x_{elite}})$ **then**
  $\mathbf{x_{elite}} \leftarrow \mathbf{x_s}$
end if
Output $\mathbf{x_{elite}}$

graphical representation is also very close to the actual software implementation. The three blocks in Figure 7.11 can be seen as software objects, derived from an abstract class "operators", that can be easily declared and initialised by a software, namely the problem analyser, which is able to run them with a certain logic coming from the information obtained during the analysis phase. SPAM algorithm has been intentionally tested over a large number of problems (132 in total) in order to be able to justify how the algorithm adapts to different fitness landscapes being able to tackle them. In order to add different problems in different dimensions, from 10 up to the large scale 1000-dimensional problems in CEC2008 and 2010, 28 more problems from CEC2010 have been considered, in both 10 and 50, on top of the usual experimental set-up. CEC2005 and BBOB2010 have been again tested in 30 and 100 dimensions respectively. All the numerical results for this algorithm are represented in terms of error, since the theoretical minimum of the f_7 problem of CEC2008 is unknown, so in order to avoid confusion, its result is not displayed, but is included in the Holm-Bonferroni procedure in Table 7.10.

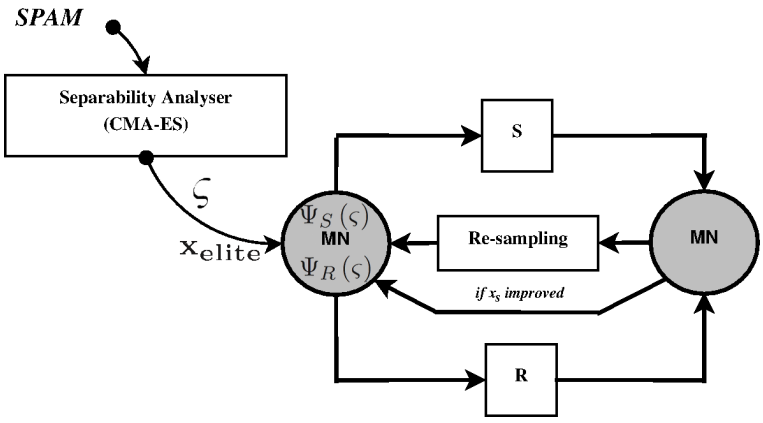


Figure. 7.11. Graphical representation of SPAM parallel structure.

7.2.4 Does the automatic design actually work? An experimental test

In order to experimentally demonstrate the effectiveness of the automatic design, SPAM has been compared with an algorithm composed of the same operators, which performs the analysis (thus detecting the initial solution $\mathbf{x_{elite}}$), but then forces the activation probabilities of S and R to 0.5. This algorithm, namely SPAM$_{0.5}$ is a version of SPAM without the automatic design based on separability analysis. In this way, SPAM$_0$.5 is a control algorithm that has all the components of SPAM except the adaptation resulting from the problem analysis.

Table 7.7. Average Error $\pm$ Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against SPAM$_{0.5}$ on CEC2013 and CEC2005 in 10 and 30 dimensions respectively.

(a) CEC2013 in 10 dimensions

	SPAM	SPAM$_{0.5}$	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$0.00e+00 \pm 0.00e+00$	=
f_2	$\mathbf{0.00e+00 \pm 0.00e+00}$	$0.00e+00 \pm 0.00e+00$	=
f_3	$\mathbf{4.45e+01 \pm 2.61e+02}$	$8.36e+02 \pm 7.47e+03$	=
f_4	$\mathbf{0.00e+00 \pm 0.00e+00}$	$0.00e+00 \pm 0.00e+00$	=
f_5	$\mathbf{2.36e-08 \pm 2.77e-08}$	$5.21e-08 \pm 8.76e-08$	+
f_6	$\mathbf{3.94e+00 \pm 4.67e+00}$	$6.70e+00 \pm 6.80e+00$	+
f_7	$\mathbf{6.67e+01 \pm 5.65e+01}$	$2.22e+09 \pm 1.94e+10$	=
f_8	$\mathbf{2.03e+01 \pm 1.88e-01}$	$2.05e+01 \pm 1.28e-01$	+
f_9	$\mathbf{6.09e+00 \pm 3.38e+00}$	$1.17e+01 \pm 4.19e+00$	+
f_{10}	$\mathbf{1.49e-02 \pm 1.24e-02}$	$1.68e-02 \pm 1.90e-02$	=
f_{11}	$\mathbf{4.17e+00 \pm 1.67e+00}$	$5.39e+01 \pm 1.21e+02$	+
f_{12}	$\mathbf{1.44e+01 \pm 6.82e+00}$	$1.19e+02 \pm 2.23e+02$	+
f_{13}	$\mathbf{8.02e+01 \pm 9.29e+01}$	$1.35e+02 \pm 2.35e+02$	=
f_{14}	$\mathbf{1.33e+02 \pm 9.00e+01}$	$1.28e+03 \pm 7.00e+02$	+
f_{15}	$\mathbf{7.14e+02 \pm 1.99e+02}$	$1.63e+03 \pm 4.24e+02$	+
f_{16}	$\mathbf{3.19e-01 \pm 2.33e-01}$	$4.24e-01 \pm 3.88e-01$	=
f_{17}	$\mathbf{1.12e+01 \pm 3.79e+00}$	$4.22e+02 \pm 5.16e+02$	+
f_{18}	$\mathbf{6.83e+01 \pm 8.47e+01}$	$4.28e+02 \pm 4.65e+02$	+
f_{19}	$\mathbf{9.05e-01 \pm 3.26e-01}$	$1.03e+00 \pm 4.25e-01$	=
f_{20}	$\mathbf{3.92e+00 \pm 4.48e-01}$	$4.34e+00 \pm 5.07e-01$	+
f_{21}	$\mathbf{2.43e+02 \pm 1.19e+02}$	$3.62e+02 \pm 8.81e+01$	+
f_{22}	$\mathbf{2.70e+02 \pm 2.26e+02}$	$1.52e+03 \pm 8.56e+02$	+
f_{23}	$\mathbf{9.97e+02 \pm 3.34e+02}$	$2.24e+03 \pm 4.17e+02$	+
f_{24}	$\mathbf{1.37e+02 \pm 3.81e+01}$	$3.00e+02 \pm 1.16e+02$	+
f_{25}	$\mathbf{1.99e+02 \pm 3.06e+01}$	$2.43e+02 \pm 5.09e+01$	+
f_{26}	$\mathbf{1.48e+02 \pm 4.29e+01}$	$2.45e+02 \pm 1.12e+02$	+
f_{27}	$\mathbf{3.56e+02 \pm 8.99e+01}$	$4.06e+02 \pm 1.20e+02$	+
f_{28}	$\mathbf{2.38e+02 \pm 9.25e+01}$	$1.03e+03 \pm 1.08e+03$	+

(b) CEC2005 in 30 dimensions

	SPAM	SPAM$_{0.5}$	
f_1	$\mathbf{0.00e+00 \pm 4.96e-14}$	$0.00e+00 \pm 5.18e-14$	=
f_2	$\mathbf{0.00e+00 \pm 5.45e-14}$	$5.68e-14 \pm 1.61e-14$	=
f_3	$1.87e+03 \pm 1.39e+03$	$\mathbf{1.75e+03 \pm 1.64e+03}$	=
f_4	$\mathbf{2.47e+04 \pm 9.22e+03}$	$1.87e+05 \pm 4.34e+05$	+
f_5	$\mathbf{8.21e+02 \pm 4.69e+02}$	$8.26e+02 \pm 4.37e+02$	=
f_6	$\mathbf{4.69e+01 \pm 9.38e+01}$	$4.73e+01 \pm 8.79e+01$	+
f_7	$\mathbf{6.41e-04 \pm 2.19e-03}$	$4.80e+03 \pm 1.21e+02$	+
f_8	$\mathbf{2.00e+01 \pm 8.76e-04}$	$2.03e+01 \pm 3.63e-01$	+
f_9	$\mathbf{2.13e-10 \pm 5.06e-11}$	$1.66e+02 \pm 1.93e+02$	+
f_{10}	$\mathbf{6.83e+01 \pm 3.22e+01}$	$2.09e+02 \pm 3.08e+02$	=
f_{11}	$\mathbf{1.35e+01 \pm 5.15e+00}$	$1.59e+01 \pm 9.74e+00$	=
f_{12}	$\mathbf{4.29e+02 \pm 7.94e+02}$	$1.24e+03 \pm 2.04e+03$	+
f_{13}	$\mathbf{2.79e+00 \pm 6.01e-01}$	$3.24e+00 \pm 7.94e-01$	+
f_{14}	$\mathbf{4.39e+01 \pm 2.75e-01}$	$4.43e+01 \pm 4.60e-01$	+
f_{15}	$\mathbf{1.28e+02 \pm 1.13e+02}$	$4.26e+02 \pm 2.54e+02$	+
f_{16}	$\mathbf{1.36e+02 \pm 5.52e+01}$	$3.14e+02 \pm 2.87e+02$	+
f_{17}	$\mathbf{2.22e+02 \pm 5.44e+01}$	$5.36e+02 \pm 2.79e+02$	+
f_{18}	$\mathbf{9.03e+02 \pm 3.04e+01}$	$9.38e+02 \pm 7.71e+01$	+
f_{19}	$\mathbf{8.95e+02 \pm 4.00e+01}$	$9.38e+02 \pm 8.13e+01$	+
f_{20}	$\mathbf{9.00e+02 \pm 5.68e+01}$	$9.45e+02 \pm 1.17e+02$	+
f_{21}	$\mathbf{4.99e+02 \pm 9.02e+00}$	$5.77e+02 \pm 1.89e+02$	=
f_{22}	$9.04e+02 \pm 2.92e+01$	$\mathbf{9.00e+02 \pm 2.83e+01}$	=
f_{23}	$\mathbf{5.32e+02 \pm 2.00e+01}$	$6.45e+02 \pm 2.11e+02$	+
f_{24}	$\mathbf{2.00e+02 \pm 3.80e-11}$	$2.21e+02 \pm 1.17e+02$	=
f_{25}	$\mathbf{1.24e+03 \pm 4.43e+02}$	$1.67e+03 \pm 1.24e+01$	+

Table 7.8. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against $SPAM_{0.5}$ on CEC2013 and BBOB2010 in 50 and 100 dimensions respectively.

(a) CEC2013 in 50 dimensions

	SPAM	$SPAM_{0.5}$	
f_1	$2.27e-13 \pm 2.27e-14$	$\mathbf{2.27e-13 \pm 0.00e+00}$	=
f_2	$3.76e+04 \pm 1.97e+04$	$\mathbf{3.35e+04 \pm 1.63e+04}$	=
f_3	$\mathbf{7.86e+06 \pm 1.32e+07}$	$9.00e+06 \pm 1.51e+07$	=
f_4	$1.20e+03 \pm 4.28e+03$	$\mathbf{5.84e+02 \pm 4.99e+02}$	=
f_5	$\mathbf{4.29e-07 \pm 5.98e-08}$	$2.11e-05 \pm 2.10e-05$	+
f_6	$\mathbf{2.61e+01 \pm 1.90e+01}$	$4.14e+01 \pm 9.74e+00$	+
f_7	$\mathbf{4.78e+01 \pm 2.15e+01}$	$3.20e+04 \pm 3.18e+05$	=
f_8	$\mathbf{2.11e+01 \pm 6.29e-02}$	$2.12e+01 \pm 4.92e-02$	+
f_9	$\mathbf{4.77e+01 \pm 3.86e+00}$	$7.23e+01 \pm 1.22e+01$	+
f_{10}	$\mathbf{1.08e-02 \pm 7.37e-03}$	$2.42e-02 \pm 1.64e-02$	+
f_{11}	$\mathbf{5.03e+01 \pm 1.03e+01}$	$1.76e+02 \pm 4.36e+02$	+
f_{12}	$\mathbf{3.17e+02 \pm 1.21e+02}$	$1.32e+03 \pm 1.39e+03$	+
f_{13}	$\mathbf{4.88e+02 \pm 8.32e+01}$	$1.98e+03 \pm 1.65e+03$	+
f_{14}	$\mathbf{1.28e+03 \pm 2.81e+02}$	$5.11e+03 \pm 3.24e+03$	+
f_{15}	$\mathbf{6.48e+03 \pm 5.37e+02}$	$8.86e+03 \pm 1.08e+03$	+
f_{16}	$8.59e-02 \pm 3.78e-02$	$\mathbf{8.53e-02 \pm 5.09e-02}$	=
f_{17}	$\mathbf{9.42e+01 \pm 9.78e+00}$	$3.97e+03 \pm 3.34e+03$	+
f_{18}	$\mathbf{5.08e+02 \pm 8.68e+01}$	$4.06e+03 \pm 3.21e+03$	+
f_{19}	$\mathbf{5.05e+00 \pm 9.04e-01}$	$5.62e+00 \pm 1.32e+00$	+
f_{20}	$\mathbf{2.43e+01 \pm 5.55e-01}$	$2.47e+01 \pm 4.44e-01$	+
f_{21}	$\mathbf{4.37e+02 \pm 3.36e+02}$	$7.58e+02 \pm 3.69e+02$	+
f_{22}	$\mathbf{1.93e+03 \pm 4.31e+02}$	$8.10e+03 \pm 4.44e+03$	+
f_{23}	$\mathbf{8.78e+03 \pm 1.26e+03}$	$1.17e+04 \pm 1.13e+03$	+
f_{24}	$\mathbf{3.25e+02 \pm 2.43e+01}$	$1.47e+03 \pm 1.06e+03$	+
f_{25}	$\mathbf{3.68e+02 \pm 1.34e+01}$	$4.68e+02 \pm 1.72e+02$	+
f_{26}	$\mathbf{2.24e+02 \pm 6.92e+01}$	$6.05e+02 \pm 7.20e+02$	+
f_{27}	$\mathbf{1.28e+03 \pm 1.97e+02}$	$1.29e+03 \pm 3.51e+02$	=
f_{28}	$\mathbf{1.42e+03 \pm 1.96e+03}$	$3.73e+03 \pm 5.80e+03$	=

(b) BBOB2010 in 100 dimensions

	SPAM	$SPAM_{0.5}$	
f_1	$\mathbf{2.42e-13 \pm 2.13e-13}$	$2.42e-13 \pm 2.13e-13$	=
f_2	$\mathbf{3.21e-07 \pm 7.33e-08}$	$2.82e+02 \pm 3.48e+02$	=
f_3	$\mathbf{9.60e+01 \pm 1.24e+01}$	$1.84e+02 \pm 8.30e+01$	+
f_4	$\mathbf{1.28e+02 \pm 1.86e+01}$	$2.79e+02 \pm 1.30e+02$	+
f_5	$\mathbf{1.31e-04 \pm 1.27e-05}$	$1.18e+02 \pm 1.05e+02$	=
f_6	$\mathbf{3.93e-08 \pm 3.87e-08}$	$5.05e-08 \pm 6.71e-08$	=
f_7	$\mathbf{5.07e+01 \pm 1.35e+01}$	$5.45e+01 \pm 1.40e+01$	+
f_8	$\mathbf{3.45e+01 \pm 1.31e+01}$	$5.41e+01 \pm 2.28e+01$	+
f_9	$\mathbf{4.70e+01 \pm 1.14e+01}$	$5.87e+01 \pm 1.80e+01$	+
f_{10}	$6.98e+02 \pm 2.21e+02$	$\mathbf{6.11e+02 \pm 2.04e+02}$	-
f_{11}	$\mathbf{8.15e+01 \pm 2.76e+01}$	$1.01e+02 \pm 3.72e+01$	+
f_{12}	$\mathbf{1.97e-02 \pm 6.48e-02}$	$2.01e-02 \pm 8.77e-02$	=
f_{13}	$\mathbf{7.74e-01 \pm 9.16e-01}$	$1.63e+00 \pm 3.10e+00$	=
f_{14}	$4.21e-05 \pm 8.74e-06$	$\mathbf{3.99e-05 \pm 8.28e-06}$	=
f_{15}	$\mathbf{2.68e+02 \pm 3.86e+01}$	$2.78e+02 \pm 4.35e+01$	=
f_{16}	$\mathbf{2.35e+00 \pm 8.17e-01}$	$2.37e+00 \pm 9.12e-01$	=
f_{17}	$\mathbf{7.93e+00 \pm 3.86e+00}$	$1.01e+01 \pm 4.78e+00$	+
f_{18}	$\mathbf{1.65e+01 \pm 9.86e+00}$	$1.91e+01 \pm 1.36e+01$	=
f_{19}	$\mathbf{1.66e+00 \pm 3.11e-01}$	$2.18e+00 \pm 1.20e+00$	+
f_{20}	$\mathbf{1.27e+00 \pm 1.47e-01}$	$1.65e+00 \pm 2.29e-01$	+
f_{21}	$\mathbf{3.42e+00 \pm 3.50e+00}$	$1.23e+01 \pm 1.27e+01$	+
f_{22}	$\mathbf{6.27e+00 \pm 7.42e+00}$	$1.56e+01 \pm 1.18e+01$	+
f_{23}	$\mathbf{6.62e-01 \pm 2.75e-01}$	$1.42e+00 \pm 8.45e-01$	+
f_{24}	$\mathbf{3.11e+02 \pm 5.78e+01}$	$3.24e+02 \pm 7.46e+01$	=

Numerical results of this test are shown in Tables 7.7(a), 7.7(b), 7.8(a), 7.8(b), 7.9(a), and 7.9(b), clearly showing that the probability setting, based on the analysis of separability leads to a performance equally high or higher than the equally distributed one. SPAM and $SPAM_{0.5}$ display a similar performance when the separability analysis leads to a 0.5 and 0.5 setting of S and R probabilities, i.e. when the two algorithms coincide. If we consider that this experiment has been carried on a diverse test-bed containing 132 problems, the success in the totality of comparison gives a convincing indication that the proposed framework is versatile and efficient.

Table 7.9. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against $SPAM_{0.5}$ on CEC2008 and CEC2010 in 1000 dimensions.

(a) CEC2008 in 1000 dimensions

	SPAM	$SPAM_{0.5}$	
f_1	$5.12e-13 \pm 6.32e-13$	$\mathbf{5.12e-13 \pm 6.31e-13}$	=
f_2	$\mathbf{1.32e-01 \pm 3.27e-02}$	$1.17e+01 \pm 1.17e+01$	+
f_3	$\mathbf{8.97e+02 \pm 2.83e+01}$	$9.28e+02 \pm 4.42e+01$	+
f_4	$\mathbf{7.17e+01 \pm 1.16e+01}$	$5.39e+03 \pm 5.28e+03$	+
f_5	$\mathbf{7.56e-05 \pm 7.36e-04}$	$6.65e-04 \pm 2.51e-03$	=
f_6	$\mathbf{1.22e+01 \pm 9.51e+00}$	$1.73e+01 \pm 6.53e+00$	+

(b) BBOB2010 in 100 dimensions

	SPAM	$SPAM_{0.5}$	
f_1	$\mathbf{1.92e-06 \pm 1.33e-07}$	$3.12e+06 \pm 3.07e+06$	=
f_2	$\mathbf{6.15e+01 \pm 1.28e+01}$	$4.58e+03 \pm 4.95e+03$	+
f_3	$\mathbf{8.06e+00 \pm 9.49e+00}$	$1.75e+01 \pm 6.33e+00$	+
f_4	$\mathbf{1.14e+11 \pm 4.32e+10}$	$2.89e+11 \pm 2.00e+11$	+
f_5	$6.78e+08 \pm 9.14e+07$	$\mathbf{6.64e+08 \pm 1.01e+08}$	=
f_6	$\mathbf{1.98e+07 \pm 7.24e+04}$	$1.98e+07 \pm 6.30e+04$	=
f_7	$\mathbf{4.17e+05 \pm 5.05e+05}$	$3.06e+06 \pm 3.04e+06$	+
f_8	$\mathbf{3.17e+06 \pm 2.82e+06}$	$9.62e+06 \pm 1.24e+07$	+
f_9	$\mathbf{3.21e+06 \pm 3.48e+05}$	$5.13e+06 \pm 2.18e+06$	+
f_{10}	$\mathbf{5.37e+03 \pm 3.84e+02}$	$7.80e+03 \pm 2.46e+03$	+
f_{11}	$\mathbf{2.00e+02 \pm 2.22e+01}$	$2.12e+02 \pm 1.83e+01$	+
f_{12}	$\mathbf{3.76e-03 \pm 1.40e-03}$	$8.17e-02 \pm 8.90e-02$	+
f_{13}	$\mathbf{1.22e+03 \pm 8.23e+02}$	$1.42e+03 \pm 8.84e+02$	=
f_{14}	$\mathbf{4.65e+06 \pm 4.18e+05}$	$7.20e+06 \pm 2.10e+06$	+
f_{15}	$\mathbf{1.00e+04 \pm 1.04e+03}$	$1.03e+04 \pm 6.44e+02$	=
f_{16}	$\mathbf{2.64e+02 \pm 1.26e+02}$	$3.70e+02 \pm 7.64e+01$	+
f_{17}	$\mathbf{4.36e-01 \pm 1.13e-01}$	$1.03e+01 \pm 1.19e+01$	+
f_{18}	$\mathbf{1.71e+03 \pm 1.09e+03}$	$2.19e+03 \pm 1.05e+03$	+
f_{19}	$\mathbf{3.32e+05 \pm 3.25e+04}$	$5.56e+05 \pm 2.27e+05$	+
f_{20}	$\mathbf{8.92e+02 \pm 3.28e+01}$	$9.31e+02 \pm 4.52e+01$	+

7.2.5 Comparison with the state-of-the-art

In order to test the potentials of SPAM with respect to modern algorithms in literature, it has also been compared against 11 more optimisers belonging to different optimisation families, as listed in Table 7.10. The pool of comparison algorithms incorporates many frameworks described in this thesis, plus the Cultural Algorithms with Iterated Local Search (CA-ILS) algorithm, belonging to the class of "Cultural Algorithms" and described in (Nguyen & Yao 2008). Powerful implementations of EAs and PSO have also been considered, i.e. G-CMA-ES (Auger & Hansen 2005) and cPSO (Neri et al. 2013). Due to the excessive number of tables, detailed numerical results of all the 9 comparison algorithms over the 132 tested problems has been avoided in order to facilitate the reading of this thesis, but can be found on a dedicated file available on the internet[1]. However, as a summary of the experiments, Table 7.10 reports the result of the Holm-Bonferroni procedure applied to the entire data set. The difference in performance between SPAM and $SPAM_{0.5}$ highlights the effect of the proposed analysis on separability. The comparison of $SPAM_{0.5}$, PMS, and other algorithms clearly shows how two simple but diverse operators can be competitive with complex algorithms such as JADE and CLPSO. It can be easily seen that SPAM significantly outperforms CA-ILS and cPSO, while G-CMA-ES seems to behave better on the chosen set of optimisation problems, but not enough to equal SPAM performances. The null-hypothesis is indeed rejected. In a nutshell, Table 7.10 shows that according to the Holm-Bonferroni procedure SPAM by far outperforms all the other algorithms contained in this study. For coherence with the

[1] http://sites.google.com/site/facaraff/home/Downloads/SPAM_Detailed_Results.pdf

experimental set-up used in this thesis, the usual set of modern and complex meta-heuristics, representing the actual cutting edge developments for the three families DE, PSO, and MA, has also been reported and discussed. In particular, detailed tables displaying results for SPAM against MDE-pBX, CCPSO2 and MA-LSCh, have been grouped in Appendix E.5. Tables E.32 (10 dimensions), E.33 (30 dimensions), E.34 (50 dimensions), E.35

Table 7.10. Holm-Bonferroni procedure on the algorithms under consideration (reference algorithm: SPAM, Rank= $9.60e+00$

j	Optimizer	$Rank$	z_j	p_j	δ/j	Hypothesis
1	SADE	7.43e+00	-5.31e+00	5.57e-08	5.00e-02	Rejected
2	MDE-pBX	7.38e+00	-5.44e+00	2.71e-08	2.50e-02	Rejected
3	CCPSO2	7.24e+00	-5.77e+00	3.94e-09	1.67e-02	Rejected
4	PMS	7.08e+00	-6.16e+00	3.62e-10	1.25e-02	Rejected
5	CLPSO	6.92e+00	-6.57e+00	2.53e-11	1.00e-02	Rejected
6	MA-LSCh	6.84e+00	-6.75e+00	7.16e-12	8.33e-03	Rejected
7	G-CMA-ES	6.52e+00	-7.53e+00	2.46e-14	7.14e-03	Rejected
8	$\text{SPAM}_{0.5}$	6.46e+00	-7.68e+00	7.80e-15	6.25e-03	Rejected
9	JADE	6.20e+00	-8.33e+00	3.98e-17	5.56e-03	Rejected
10	cPSO	3.70e+00	-1.45e+01	1.15e-47	5.00e-03	Rejected
11	CA-ILS	2.40e+00	-1.76e+01	7.39e-70	4.55e-03	Rejected

(100 dimensions), E.36 and E.37 (1000 dimensions) completely confirm the previous achievement, since in terms of Wilcoxon Rank-Sum test SPAM significantly outperforms MDE-pBX in 88 cases out of 132, CCPSO2 over 94 problems and MA-LSCh 81 times. It must be observed that while the other cutting edge developments tend to be specialised to solve some classes of functions, SPAM appears to efficiently tackle a large range of problems. For example, while MA-LSCh displays a very good performance in low dimensions, this is offset by a less than good performance in LSOP. On the contrary, SPAM, thanks to its problem analyser, coordinates its component in a versatile way, thus offering a robust performance.

7.2.6 Lennard-Jones potential minimisation

In order to prove the viability of SPAM and the efficiency of the analysis on separability for real-world problems, the previously chosen algorithms have been run against SPAM for the LJP problem. The LJP is widely used in physics for representing the interaction energy between non-bonding particles in a fluid. In this case, according to the description given in CEC2011 (Swagatam & Suganthan 2010), it refers to the minimisation problem of the potential energy of a set of atoms by locating them within three-dimensional space. In order to determine their position and evaluate the relative LJP, 3 parameters, i.e. coordinates along the axes x, y and z, for each atom needs to be stored in the fitness function. In this study, an atomic cluster of 40 atoms has been considered, resulting in a 120-dimensional problem. Each algorithm under examination has been run 10 times with a computational budget of 150000 functional calls (in accordance with the description given in (Swagatam & Suganthan 2010)). It must be said that in order to allow the sub-swarm decomposition in CCPSO2 for this problem, the original divisors have been changed and set to 30 and 60 variables. Numerical results are given in Table 7.11 and Figure 7.12, respectively. Also in this case SPAM can reliably detect solutions characterised by a high performance, outperforming the other meta-heuristics.

Table 7.11. Average Fitness $\pm$ Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on LJP problem in 120 dimensions.

SPAM	$\mathbf{-9.29e+01 \pm 8.54e+00}$	
MDE-pBX	$-8.25e+01 \pm 6.12e+00$	=
CPSO2	$-2.35e+01 \pm 1.43e+00$	+
MA-LSCh	$-1.16e+01 \pm 1.72e+00$	+

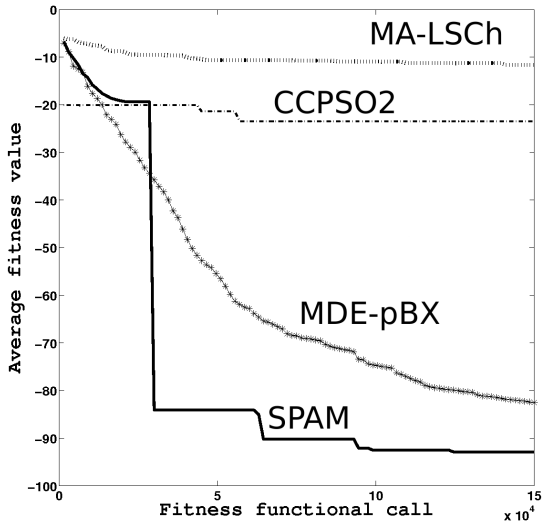


Figure. 7.12. Average fitness trend of SPAM, MDE-pBX, CCPSO2 and MA-LSCh on the LJP problem

7.3 Chapter remarks

This work must be interpreted as a first step towards a complete software platform for the automatic design of optimisation algorithms. Future work on separability (**IRQ IV**) will consider other ways to measure the correlation between pairs of variables. More importantly, it will also extend the analysis to other problem properties, such as ill-conditioning and multi-modality. In addition, future work will investigate broader databases of operators and other design criteria.

As a final summary, Table 7.12 highlights the main characteristics of the MC algorithms presented in Chapter 4 and Chapter 7, extending Table 4.1 of Chapter 4. As for memetic algorithms, each components has been listed in the table. Moreover, a particular attention has been given to the memory requirements for each meme. In this regard, it must be noted that despite some operators requiring the same amount of memory, i.e. CMA-ES, $(1+1)$-CMA-ES, Rosenbrock and Powell's memory footprint increases with the square of the dimensionality of the problem, they need a different amount of time to process a new solution. In this regard, CMA-ES represents the slowest meme and can slow down the entire optimisation process of algorithms which would be, otherwise, very light. This is the case of CMA-ES-RIS; an increase on the performance is offset by an increase on the algorithm computational overhead. To some extent, SPAM also suffers from these problems, but as previously explained, this procedure is necessary to perform the analysis on separability and then returns useful information to build up a new algorithm. Future research will also aim at replacing the analyser with a lighter one, still guaranteeing the same accuracy.

Table 7.12. Memetic Computing algorithms overview.

Algorithm	Solutions	Memes	Structure	Comments
MS-CAP	*Population based*	-CAP -MPRS	*Sequential*	*-Versatile -Employing multiple DE and PSO based perturbations*
3SOME	*Single solution*	-L -M -S	*Parallel (deterministic)*	*-Memory saving✓ -Real-time applications✓ -Bottom-up exploration strategy -Versatile (not in case of non-separability)*
ML-3SOME	*Single solution*	-L -M -S	*Parallel (probabilistic)*	*-Memory saving✓ -real-time applications✓ -Meta-Lamarckian learning -Outperforms 3SOME in low dimensions*
S-3SOME	*Single solution*	-L -SHRINKING -S	*Parallel (deterministic)*	*-Memory saving✓ -Real-time applications ✓* -Outperforms 3SOME on LSOP
RI-3SOME	*Single solution*	-L -DIAG-M -S	*Parallel (deterministic)*	*-Memory saving✓ -Real-time applications✓ -Re-adapts DE/current-to-rand/1 move -Non separable problems ∼*
RIS-3SOME	*Single solution*	-L -DIAG-SHRINKING -S	*Parallel (deterministic)*	*-Memory saving ✓-Real-time applications✓ -Re-adapts DE/current-to-rand/1 move -Non separable problems✓*
μDE-3SOME	*μ-population*	-L μDE -S	*Parallel (deterministic)*	*-Memory saving ∼ -Real-time ✓ -Single solution but needs a μ-population -Non-separable problems ✓*
(1+1)-CMA-ES-3SOME	*Single solution*	-L -M -(1+1)-CMA-ES	*Parallel (deterministic)*	*$t \sim \mathcal{O}\left(n^2\right)$ $m \sim \mathcal{O}\left(n^2\right)$ -Uni-modal ill-conditioned problems✓ -Non separable problems ✓*
3SOME-ROSENBROCK	*Single solution*	-L -M -ROSENBROCK	*Parallel (deterministic)*	*-Rotation matrix: $m \sim \mathcal{O}\left(n^2\right)$ -Outperforms 3SOME in uni-modal and non separable problems (in low dimensions)*
3SOME-POWELL	*Single solution*	-L -M -POWELL	*Parallel (deterministic)*	*-Directions matrix: $m \sim \mathcal{O}\left(n^2\right)$ -employed GSS for line minimisation -Non separable problems (low dimensions) ✓*
RS	*Single solution*	-RE-SAMPLING -S	*Sequential*	*-Memory saving ✓-Real-time ✓ -Pure random sampling (no inheritance)*
RIS	*Single solution*	-RE-SAMPLING -S	*Sequential*	*-Memory saving✓ -Real-time applications✓ -Inheritance mechanism (exponential x-over)*
CMA-ES-RIS	*population-based*	-CMA-ES -RIS	*Sequential*	*$t \sim \mathcal{O}\left(n^3\right)$ $m \sim \mathcal{O}\left(n^2\right)$ -Super-fit scheme*
VISPO	*Single solution*	VPS RESTART	*Sequential*	*-Memory saving ✓-Real-time ✓ -Learning period (non-separability detection)*
PMS	*Single solution*	-L -S -R	*Parallel (probabilistic)*	*Rotation matrix: $m \sim \mathcal{O}\left(n^2\right)$ -General purpose (2 complementary LSs) -Robust and versatile*
SPAM	*-Population based (Analysis) -Single solution (optimisation)*	*-CMAES* -RE-SAMPLING -R -S	*Parallel (probabilistic)*	*-Landscape analysis: $t \sim \mathcal{O}\left(n^3\right)$ -Automatically tailored to the problems✓ -Covariance/Rotation matrix: $m \sim \mathcal{O}\left(n^2\right)$ -Robust and versatile*

Chapter 8

The blessing of dimensionality

This chapter extends the work in (Caraffini et al. 2014) on the evaluation of an index carrying useful information on the correlation amongst the variables of a given objective function. Specifically, two independent tests based on Pearson and Spearman coefficients have been set up and run for a study on dimensionality and scalability of optimisation problems. Numerical results confirmed that, in low dimensions, separable problems are characterised by a weak (or null) correlation amongst the variables, while the variables of complex non-separable problems can be strongly correlated. The main finding of this study is that correlation amongst the variables appears to decrease while the dimensionality increases regardless of the nature of the problem. This achievement derives from an investigation summed up with the last two **IRQ**. After a clarifying section on the motivations behind this study, Chapter 8.1, **IRQ V** is answered in Chapter 8.2. **IRQ VI** is instead experimentally addressed employing two different tests, as described in the remainder of this chapter. Both the tests confirm the intuition arose during the testing of the optimiser in this thesis on LSOP, that in high dimensional problems the correlation between is weak, thus justifying the good performances obtained with optimisation algorithm perturbing a single design variable at time.

8.1 Motivations

The motivation behind this work and potential future developments in this direction are obviously related to the difficulties in tackling LSOP. Without any doubt, dimensionality plays a crucial role when facing an optimisation problem, for multiple reasons.

As a first consideration, let us consider a unidimensional decision space $\mathcal{D}$. Let $\mathcal{D}$ be a set composed of 100 points. Let us consider now a function f defined over the set $\mathcal{D}$. Without a loss of generality let us assume that there exists a solution $\mathbf{x}^* \in \mathcal{D}$ such that $f(\mathbf{x}^*)$ is minimal. Hence, in order to find the global minimum $\mathbf{x}^*$ an optimisation algorithm needs at most 100 fitness evaluations. This problem would be very easy for a modern computer. On the other hand, if the problem is scaled up to two dimensions, there will be one optimum $\mathbf{x}^*$ in a space composed of $100^2 = 10000$ candidate solutions. If the problem is scaled up to 1000 dimensions,

the optimum will be only one point in a space of 100^{1000} solutions. With an exhaustive search, the latter problem would be extremely hard to solve in a viable amount of time. Thus a specifically designed algorithm will be required to tackle it. In other words, since the decision space grows exponentially with the problem dimensionality, the detection of the optimal solution in high dimensions is like the search for a needle in a haystack and requires some specific strategies.

In addition to that, the problem dimensionality affects not only the number of candidate solutions in the search space, but also other intrinsic features of the search space itself. For example, a unitary radius sphere in a 3-dimensional Euclidean space has surface area $S_2 = 4\pi$ and volume $V_3 = \frac{4}{3}\pi$. In the generic n-dimensional space, it can be easily proved that the ratio between volume and surface is $\frac{1}{n}$. This means that if we consider a unitary radius sphere in high dimensions and randomly sample some points within it, most of them will be located on its surface as its volume is a small fraction of it.

Moreover, LSOP can be hard to solve with general-purpose algorithms, as some optimisers that easily solve a problem in e.g. 30 dimensions can display a poor performance to solving the same problem scaled up to e.g. 300 dimensions. The deterioration in the performance of optimisation algorithms as the dimensionality of the search space increases, is commonly called the "curse of dimensionality", see (Van den Bergh & Engelbrecht 2004), and generally affects every kind of search logic. For example, several studies show that DE and PSO can easily display a poor performance when applied to solve large scale optimisation problems, see e.g. (Neri & Tirronen 2010) and (Van den Bergh & Engelbrecht 2004).

Furthermore, dimensionality has a direct impact on the computational cost of the optimisation. In general, this is true because, due to the large decision space, a large budget is usually necessary to detect a solution with a high performance. Moreover, due to high dimensionality, algorithms which perform a search within the neighbourhood of a candidate solution (e.g. Hooke-Jeeves, Algorithm 4 of Chapter 2.2.3) might require a very large number of fitness evaluations at each step of the search, while population based algorithms are likely to either prematurely converge to suboptimal solutions, or stagnate due to an inability to generate new promising search directions. Other approaches that inspect the interaction between pairs or variables in order to perform an exploratory move, see e.g. (Auger & Hansen 2005), can be computationally onerous and in some cases, see e.g. (Molina, Lozano, García-Martínez & Herrera 2010*b*), unacceptably expensive for modern computers.

8.2 Identification of successful strategies for handling LSOP

Albeit difficult, in many real-world cases LSOP must be tackled in order to achieve a solution with a high performance, e.g. in scheduling see (Marchiori & Steenbeek 2000), in chemical engineering see (Kononova, Hughes, Pourkashanian & Ingham 2007) and in engineering design, see (Akay & Karaboga 2012). Thus, in recent years, several modern meta-heuristics have been proposed in order to tackle LSOP. For example, modified versions of Ant Colony Optimiser (Korošec & Šilc 2008) and a memetic version of Ant Bee Colony Algorithm (Fister, Jr., Brest & Zumer 2012) have been proposed for addressing large scale optimisation problems. Within the plethora of methods to tackle LSOP, a division into the following two macro-categories is proposed here.

- **Methods that intensively exploit promising search directions.** These algorithms with an apparently counter-intuitive action, instead of exploring the large decision space, give up the search for the global optimum and use an intensive exploitation to improve upon a set of initial solutions to detect a solution with a high quality (regardless of its optimality). Two popular ways to implement this approach have been proposed in the literature. The first way to achieve this aim is by using population based algorithms with very small populations, see (Parsopoulos 2009), and (Dasgupta, Das, Biswas & Abraham 2009), or with a population that shrinks during the run, see (Brest & Maučec 2008), (Zamuda, Brest, Bošković & Žumer 2008*a*) and (Iacca, Mallipeddi, Mininno, Neri & Suganthan 2011). The second way to achieve this aim is by using highly exploitative local search algorithms by combining them with other algorithms and integrating them within population based structures. Specifically, a coordination of multiple local search components is used to tackle LSOP in (Tseng & Chen 2008). A part or a modification of this logic has been coupled and integrated within other algorithmic frameworks in μDEA (Caraffini, Neri & Poikolainen 2013), 3SOME (Iacca, Neri, Mininno, Ong & Lim 2012), PMS (Caraffini, Neri, Iacca & Mol 2013) and RIS (Caraffini, Neri, Passow & Iacca 2013*a*). It must be said that these algorithms tend to use a simple local search component that exploits the decision space by perturbing the candidate solution along the axes.

- **Methods that decompose the search space.** Some other papers propose a technique, namely cooperative co-evolution, originally defined in (Potter & De Jong 1994) and subsequently developed in other works, see e.g. (Sofge, De Jong & Schultz 2002). The concept of the cooperative co-evolution is to decompose LSOP into a set of low dimensional problems, which can be separately solved and then recombined in order to compose the solution of the original problem. It is obvious that if the objective function (fitness function) is separable then the problem decomposition can be trivial, while for non-separable functions the problem decomposition can turn out to be a very difficult task. However, some techniques for performing the decomposition of non-separable functions have been developed, see (Potter & De Jong 2000). Recently, cooperative co-evolution procedures have been successfully integrated within DE frameworks for solving LSOP, see e.g. (Shi, Teng & Li 2005), (Yang, Tang & Yao 2007), (Zamuda, Brest, Bošković & Žumer 2008*b*), (Olorunda & Engelbrecht 2007) and (Yang, Tang & Yao 2008). Recently, a very successful implementation of cooperative co-evolution has been integrated within a PSO framework, i.e. CCPSO2, in (Li & Yao 2012).

An interesting study related to LSOP has been presented in (Ros & Hansen 2008) where a modified version of CMA-ES has been proposed for tackling separable problems. In this case, the proposed algorithm makes use of a diagonal matrix to determine the newly sampled points and then the search directions. Hence, this modified variant performs stochastic perturbation on each design variable to solve separable problems, as is shown in Figure 3.3 of Chapter 3.1.2.1. Regardless of the implementation details (for details see Chapter 3.1.2.1), it is highlighted here that the basic search strategy of this algorithm, namely sep-CMA-ES, consists of performing moves along the axes and progressively adapting the search radii along each of the axes. On the contrary, the original CMA-ES, see e.g. (Hansen et al. 2003), considers a full covariance matrix i.e. considers the interaction between each pair of variables and performs diagonal search moves. As expected, sep-CMA-ES appears to be

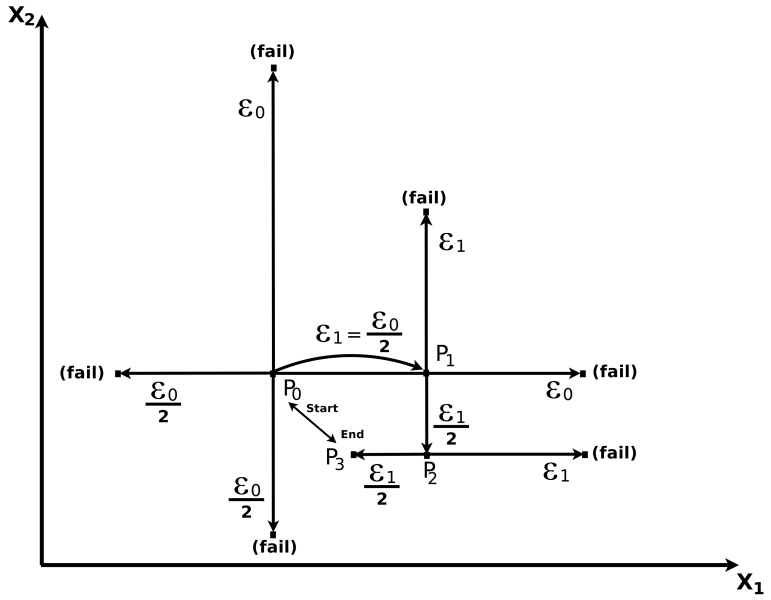


Figure. 8.1. Graphical representation of the first successful strategy for LSOPs.

efficient for separable problems, but is outperformed by CMA-ES on non-separable problems where diagonal moves are beneficial and often necessary. It was furthermore observed in (Ros & Hansen 2008) that sep-CMA-ES outperforms CMA-ES in high dimensions (larger than 100) even for non-separable problems. In other words, it was shown that while the interaction among variables is very important in low dimensions, it does not appear as important when the dimensionality grows. This stochastic perturbation along the axis represents a successful implementation for dealing with LSOP.

Another search logic for large scale optimisation is the one presented in CCPSO2 (Li & Yao 2012). Despite its simplicity, it is probably one of the most successful representatives of this class of algorithms. In (Li & Yao 2012), according to some predetermined division coefficients and a simple random selection, some variables are kept constant while others are perturbed following a simplified PSO logic. This strategy, although it does not correspond exactly to moves along the axes, is strictly related to it, as it keeps most of the variables constant and simultaneously moves along the remaining ones (e.g. groups of 5 variables in a 1000 dimensional space). For implementation details see Chapter 3.2.1 and the related Figure 3.4 shows an example of its working principle.

The last strategy for handling high dimensional problems is represented by the S operator (Algorithm 5). This optimiser has been widely used and described in this work and so does not require further explanation. For the sake of clarity, in Figure 8.1 an example is reported of its deterministic search logic along the axis.

On the basis of the short descriptions of successful strategies for LSOP, the following research question arises: *What do these strategies have in common?* or, in other words, *What algorithmic mechanism appears promising for tackling LSOP?* **(IRQ V)** . According to the previous taxonomy, the answer to this question is that the winning strategy appears to be an exploitative search that separately perturbs the variables or small groups of them while the other ones are kept constant. This consideration leads to the more intriguing **IRQ VI**: *Is*

there a way to relate the dimensionality to the success of the strategies that perturb the variables separately? or in other words *Can we give an explanation to why these approaches are working on LSOP and thus understand when to apply them in the future?*

8.3 Two tests for estimating the correlation between pairs of variables

As will become clear in the following sections, to give an answer to the second research question two numerical tests have been used to make an estimation of the correlation between pairs of variable. Both the tests make use of the following **pre-phase**. As a preliminary step, a set of λ candidate solutions is sampled within the decision space $\mathcal{D}$. Then, the CMA-ES with rank-μ-update and weighted recombination, is applied for $n \times 1000$ fitness evaluations. According the philosophy of CMA-ES, the matrix $\mathbf{C}$ evolves and reliably approximates the theoretical covariance matrix. A covariance matrix is a correlation matrix, i.e. a matrix that describes the correlation between pairs of variables and, at the same time, approximates the shape of the basins of attraction, i.e. those regions of the fitness landscape that have the best performance. In this way, CMA-ES samples new points from a distribution that adapts to the fitness landscape itself. Once the estimated covariance matrix $\mathbf{C}$ has been computed (according to the allotted computational budget), the pairwise correlation among the decision variables can be evaluated in two ways. The first test considered in this work, namely the Pearson test (Pearson 1903), takes into account each element of the matrix $C_{i,j}$ as previously described in Chapter 7.2.1. Given the covariance matrix, Equation 8.2 is applied in order to work out the Pearson correlation coefficient (ρ) for each pair of variable. Thus, the corresponding Pearson correlation matrix can be stored (only $\frac{(n^2-n)}{2}$ elements of the matrix are needed, since the diagonal is a vector of ones and the matrix is symmetric), and for the same reasons explained for the analysis of separability in SPAM, the absolute value of each element is considered. Finally, the following index is computed by averaging the elements of the matrix $|\rho|$:

$$\varsigma = \frac{2}{n^2 - n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |\rho_{i,j}|. \tag{8.1}$$

and applies the following transformation:

$$\rho_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i} C_{j,j}}}. \tag{8.2}$$

The second test used, i.e. the Spearman test (Spearman 1904), requires an introduction. By means of the covariance matrix $\mathbf{C}$, m points are sampled within the decision space. Considering that each point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a vector having n elements, these m points compose the following $m \times n$ matrix:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \ldots & x_{2,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{m,1} & x_{m,2} & x_{m,3} & \ldots & x_{m,n} \end{pmatrix}$$

that can be represented as

$$\mathbf{X} = \left(\mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^n}\right)$$

where $\mathbf{X^j}$ is the generic j^{th} column vector of the matrix $\mathbf{X}$. For each column vector, the elements are substituted with their ranking. More specifically, for the generic column vector $\mathbf{X^j}$, the lowest value is replaced with the its ranking 1, the second lowest with 2, and so on until the highest value is replaced with m. If l elements have the same value, an average ranking is assigned. For example, if three elements corresponding to the rank $3, 4$, and 5 have the same value, the ranking 4 is assigned to all of them. This procedure can be seen as a matrix transformation that associates to the matrix $\mathbf{X}$ composed of the points a new rank matrix $\mathbf{R}$ where the element $x_{i,j}$ is replaced with its rank $r_{i,j}$:

$$\mathbf{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & ... & r_{1,n} \\ r_{2,1} & r_{2,2} & r_{2,3} & ... & r_{2,n} \\ ... & ... & ... & ... & ... \\ r_{m,1} & r_{m,2} & r_{m,3} & ... & r_{m,n} \end{pmatrix} = \left(\mathbf{R^1}, \mathbf{R^2}, \ldots, \mathbf{R^n}\right).$$

From the rank matrix $\mathbf{R}$, a new matrix $\mathbf{T}$ is calculated by computing the Pearson correlation coefficients of the ranks. More specifically, the correlation between the i^{th} and j^{th} variables is given by:

$$\tau_{i,j} = \frac{\sum_{k=1}^{m}\left(r_{k,i} - \bar{R^i}\right)\cdot\left(r_{k,j} - \bar{R^j}\right)}{\sqrt{\sum_{k=1}^{m}\left(r_{k,i} - \bar{R^i}\right)^2 \cdot \sum_{k=1}^{m}\left(r_{k,j} - \bar{R^j}\right)^2}} \tag{8.3}$$

where $\bar{R^i}$ and $\bar{R^j}$ are the mean values of the i^{th} and j^{th} column vectors, respectively. Considering that $\forall i, j$, then $\tau_{i,i} = 1$ and $\tau_{i,j} = \tau_{j,i}$, the matrix $\mathbf{T}$ is symmetric and displays unitary diagonal elements. Since, analogous to the Pearson test, we are not interested in the sign of the correlation, the absolute value of the matrix $\mathbf{T}$ is calculated:

$$|\mathbf{T}| = \begin{pmatrix} 1 & |\tau_{1,2}| & |\tau_{1,3}| & ... & |\tau_{1,n}| \\ X & 1 & |\tau_{2,3}| & ... & |\tau_{2,n}| \\ X & X & 1 & ... & |\tau_{3,n}| \\ ... & ... & ... & ... & ... \\ X & X & X & X & 1 \end{pmatrix}.$$

The average Spearman correlation coefficient φ is computed as the average value of the $\frac{\left(n^2-n\right)}{2}$ elements of the matrix $\mathbf{T}$, under consideration:

$$\varphi = \frac{2}{n^2 - n} \cdot \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |\tau_{i,j}|. \tag{8.4}$$

Both the tests have been considered since each one of them has advantages and disadvantages with respect to the other. As explained in (Hauke & Kossowski 2011) the Spearman coefficient, unlike the Pearson coefficient, is not a measure of the linear correlation between two variables but is a measure of the generic correlation. The Spearman coefficient does not require any assumption on the statistical process, thus being non-parametric (distribution free). On the other hand, besides being less computationally expensive, the Pearson coefficient

is more accurate if the correlation can be approximated to be linear. This circumstance realistically occurs in several cases making the Pearson coefficient very reliable in some cases, see (Cox & Hinkley 1974).

8.4 Experimental set-up and results

The two tests illustrated above have been performed over the 19 test problems introduced in the Test Suite for the SISC2010 problems (Lozano et al. 2011). Hence, both Pearson and Spearman correlation coefficients have been calculated over these 19 test problems in 10, 30, 50, 100, 500 and 1000 dimensions. In order to find the optimal number of samples to be drawn from the distribution and apply the statistic procedure, a preliminary experiment has been performed for both the statistic tests with SISC2010 in the aforementioned dimensionality values. In detail, a number of samples proportional to the dimensionality of the problem (i.e. the following configurations have been tested: $1 \cdot n$, $5 \cdot n$, $10 \cdot n$ and $100 \cdot n$) have been considered and the corresponding index value compared against the one obtained with fixed number of samples, regardless of the growing dimensionality of the problem (3 sets of samples have been used in this case with sizes 10, 100, and 1000 points). It has been observed that a set of 100 points, for Pearson, and 1000, for Spearman, provides a coefficient as stable and reliable as the one obtained with a higher fixed number of points, or a number of points increasing with the dimensionality of the problem. In this light, and in order to keep the experimental set-up simple and light, the population size for CMA-ES has been set equal to 100, and in case of Spearman test, the final population has been sampled 10 times. Due to the stochastic nature of the tests, in order to properly visualise and present the two indices, each index has been calculated 50 times per problem. Tables 8.1 and Table 8.2 display average and standard deviation for the correlation coefficient indices.

Table 8.1. Mean value and standard deviation for the Pearson correlation index ς for SISC2010 over multiple dimensionality values.

	ς_{10D}	ς_{30D}	ς_{50D}	ς_{100D}	ς_{500D}	ς_{1000D}	Separable
f_1	0.054 ± 0.006	0.032 ± 0.001	0.025 ± 0.001	0.015 ± 0.001	0.002 ± 0.000	0.001 ± 0.000	YES
f_2	0.068 ± 0.008	0.043 ± 0.002	0.034 ± 0.001	0.021 ± 0.001	0.014 ± 0.0015	0.021 ± 0.001	–
f_3	0.178 ± 0.077	$0.072 \pm .0190$	0.035 ± 0.002	0.020 ± 0.002	0.002 ± 0.000	0.001 ± 0.000	–
f_4	0.058 ± 0.014	0.036 ± 0.005	0.026 ± 0.001	0.019 ± 0.002	0.004 ± 0.000	0.002 ± 0.000	YES
f_5	0.053 ± 0.007	0.033 ± 0.001	0.026 ± 0.001	0.017 ± 0.000	0.003 ± 0.000	0.001 ± 0.000	–
f_6	0.058 ± 0.010	0.038 ± 0.002	0.030 ± 0.008	0.017 ± 0.004	0.003 ± 0.001	0.002 ± 0.000	YES
f_7	0.059 ± 0.018	0.033 ± 0.001	0.035 ± 0.001	0.019 ± 0.001	0.003 ± 0.000	0.001 ± 0.000	YES
f_8	0.146 ± 0.008	0.069 ± 0.002	0.067 ± 0.002	0.053 ± 0.004	0.007 ± 0.002	0.003 ± 0.000	–
f_9	0.508 ± 0.428	0.073 ± 0.084	0.069 ± 0.064	0.094 ± 0.039	0.040 ± 0.015	0.025 ± 0.003	–
f_{10}	0.051 ± 0.004	0.033 ± 0.001	0.024 ± 0.001	0.016 ± 0.003	0.004 ± 0.000	0.002 ± 0.000	–
f_{11}	0.276 ± 0.272	0.092 ± 0.056	0.097 ± 0.066	0.068 ± 0.040	0.037 ± 0.0031	0.021 ± 0.006	–
f_{12}	0.109 ± 0.029	0.055 ± 0.008	0.041 ± 0.006	0.032 ± 0.009	0.008 ± 0.0006	0.005 ± 0.001	–
f_{13}	0.241 ± 0.077	0.075 ± 0.021	0.058 ± 0.003	0.046 ± 0.004	0.014 ± 0.0006	0.010 ± 0.001	–
f_{14}	0.091 ± 0.010	0.055 ± 0.005	0.040 ± 0.004	0.031 ± 0.007	0.010 ± 0.0015	0.007 ± 0.001	–
f_{15}	0.056 ± 0.008	0.040 ± 0.002	0.029 ± 0.000	0.025 ± 0.010	0.008 ± 0.0017	0.006 ± 0.001	–
f_{16}	0.094 ± 0.011	0.092 ± 0.024	0.084 ± 0.028	0.048 ± 0.013	0.017 ± 0.0021	0.013 ± 0.004	–
f_{17}	0.206 ± 0.121	0.144 ± 0.050	0.078 ± 0.011	0.061 ± 0.013	0.024 ± 0.0021	0.015 ± 0.002	–
f_{18}	0.226 ± 0.224	0.074 ± 0.024	0.049 ± 0.021	0.053 ± 0.034	0.036 ± 0.0099	0.022 ± 0.001	–
f_{19}	0.066 ± 0.006	0.063 ± 0.007	0.045 ± 0.007	0.027 ± 0.012	0.006 ± 0.0006	0.004 ± 0.001	–

As can immediately be observed, both the proposed coefficients appear to be closely related to the problem dimensionality. More specifically, regardless of the nature of the problem, the correlation amongst variables appear to decay with the dimensionality. All the problems display the maximum values of Pearson and Spearman coefficients in low dimensions, while these coefficients tend to take small values in large scale cases,

Table 8.2. Mean value and standard deviation for the Spearman correlation index φ for SISC2010 over multiple dimensionality values.

	φ_{10D}	φ_{30D}	φ_{50D}	φ_{100D}	φ_{500D}	φ_{1000D}	Separable
f_1	0.093 ± 0.011	0.085 ± 0.003	0.084 ± 0.002	0.082 ± 0.001	0.080 ± 0.001	0.014 ± 0.000	YES
f_2	0.099 ± 0.010	0.091 ± 0.003	0.089 ± 0.002	0.083 ± 0.001	0.082 ± 0.001	0.016 ± 0.001	–
f_3	0.219 ± 0.062	0.113 ± 0.018	0.092 ± 0.005	0.084 ± 0.002	0.081 ± 0.002	0.014 ± 0.000	–
f_4	0.127 ± 0.017	0.091 ± 0.004	0.085 ± 0.003	0.083 ± 0.001	0.080 ± 0.001	0.014 ± 0.000	YES
f_5	0.103 ± 0.012	0.086 ± 0.002	0.083 ± 0.001	0.081 ± 0.001	0.080 ± 0.001	0.014 ± 0.000	–
f_6	0.096 ± 0.009	0.089 ± 0.004	0.085 ± 0.004	0.082 ± 0.001	0.080 ± 0.001	0.014 ± 0.000	YES
f_7	0.103 ± 0.018	0.086 ± 0.004	0.086 ± 0.002	0.083 ± 0.001	0.080 ± 0.001	0.014 ± 0.000	YES
f_8	0.186 ± 0.019	0.116 ± 0.002	0.116 ± 0.003	0.097 ± 0.003	0.081 ± 0.003	0.014 ± 0.000	–
f_9	0.675 ± 0.417	0.128 ± 0.061	0.112 ± 0.046	0.127 ± 0.029	0.096 ± 0.029	0.022 ± 0.002	–
f_{10}	0.097 ± 0.010	0.088 ± 0.004	0.084 ± 0.002	0.082 ± 0.001	0.081 ± 0.001	0.014 ± 0.000	–
f_{11}	0.577 ± 0.404	0.187 ± 0.101	0.112 ± 0.057	0.115 ± 0.010	0.098 ± 0.030	0.022 ± 0.001	–
f_{12}	0.160 ± 0.020	0.102 ± 0.007	0.096 ± 0.005	0.093 ± 0.007	0.084 ± 0.007	0.017 ± 0.001	–
f_{13}	0.237 ± 0.069	0.130 ± 0.014	0.130 ± 0.006	0.085 ± 0.004	0.089 ± 0.004	0.019 ± 0.001	–
f_{14}	0.123 ± 0.011	0.101 ± 0.008	0.098 ± 0.005	0.093 ± 0.004	0.087 ± 0.004	0.018 ± 0.002	–
f_{15}	0.105 ± 0.015	0.089 ± 0.036	0.089 ± 0.003	0.084 ± 0.003	0.081 ± 0.003	0.022 ± 0.000	–
f_{16}	0.132 ± 0.014	0.142 ± 0.030	0.128 ± 0.021	0.105 ± 0.008	0.089 ± 0.008	0.020 ± 0.003	–
f_{17}	0.210 ± 0.138	0.189 ± 0.067	0.123 ± 0.014	0.109 ± 0.011	0.088 ± 0.011	0.018 ± 0.001	–
f_{18}	0.386 ± 0.229	0.134 ± 0.067	0.098 ± 0.015	0.106 ± 0.025	0.096 ± 0.025	0.020 ± 0.002	–
f_{19}	0.108 ± 0.011	0.102 ± 0.009	0.102 ± 0.004	0.085 ± 0.009	0.081 ± 0.009	0.014 ± 0.001	–

being nearly null in 1000 dimensions. Moreover, it can be observed that the concept of correlation between pairs of variables and the concept of separability, albeit not coincident, are logically related. For this reason, the separable functions are marked in the presented Tables. Since separable functions in n variables can be expressed as the sum of n functions in one variable, the optimisation of a separable function can be solved by optimising each variable separately, i.e. by perturbing the axes one by one until the optimum along the axis under consideration is detected. An optimisation problem of this kind is definitely characterised by uncorrelated variables. On the other hand, non-separable problems require a simultaneous perturbation of multiple variables to be tackled, as CMA-ES proposes, see e.g. (Hansen et al. 2003). Focussing on the 10 dimensional problems, we can see that both Pearson and Spearman coefficients appear to have, in the majority of the cases, their lowest values (and close to zero) in connection to the separable problems. Although some non-separable problems are still characterised by low correlation coefficients, separable problems always show a low correlation amongst the variable. Moreover, while the problems $f_1 - f_8$ are basis functions, test problems $f_9 - f_{11}$ are shifted functions, and $f_{12} - f_{19}$ are composition functions resulting from the combination of multiple functions. It can be easily observed that correlations coefficients tend to be fairly small in connection to the problems belonging to the first block, and higher in the other two blocks, especially for composition functions. For example, test problem f_1 is the sphere problem, which is obviously separable. In the case of f_1, the correlation coefficients are already nearly zero in 10 dimensions. The functions belonging to the third group are clearly non-separable (because they have been built to be so). In these cases, the correlation coefficients tend to take bigger values in low dimensions. The trend of the coefficients with respect to the dimensionality values decreases monotonously (except for little oscillations, in isolated cases, due to the stochastic nature of the tests), regardless of the nature of the problem (e.g. supposed separability) and the correlation indices in 10 dimensions. This fact occurs for both the indices considered in this study despite the fact that they use a different strategy to measures the correlation among the variables (e.g. Pearson's index measure only the linear correlation). An interpretation is that, the monotonous decay of the correlation indices with respect to the problem dimensionality means that every LSOP is characterised, to some extent, by uncorrelated variables. Hence, the problem can be tackled by operators that perturb the axes separately or that consider at each exploratory move only small groups of variables (i.e., keeping most of the variables constant). Thus, one can conclude that if on the one hand high

dimensionality increases the difficulty of the problem, on the other hand it makes the optimisation easier, since a search strategy that perturbs the axes separately is likely to return satisfactory results. This analysis explains why the three search strategies illustrated at the beginning of this chapter appear successful in approaching LSOP. In this sense the dimensionality is not only a "curse" but also a "blessing". In order to confirm that the obtained results are not biased by the chosen test-bed, the same tests have also been run over the CEC2013 test-bed, see (Liang et al. 2013). Since this test-bed is scalable for a limited amount of dimensionality values, we performed the tests over all the available dimensionality values, that is 10, 30, and 50 dimensions. Numerical results are given in Tables 8.3 and 8.4. Finally, the BBOB2010 test-bed, see (Hansen, Auger, Finck, Ros

Table 8.3. Mean value and standard deviation for the Pearson correlation index ς for CEC2013 over increasing dimensionality values.

	ς_{10D}	ς_{30D}	ς_{50D}	Separable
f_1	0.054 ± 0.007	0.032 ± 0.001	0.027 ± 0.001	YES
f_2	0.563 ± 0.036	0.256 ± 0.025	0.238 ± 0.024	–
f_3	0.271 ± 0.147	0.116 ± 0.019	0.059 ± 0.025	–
f_4	0.110 ± 0.028	0.085 ± 0.011	0.091 ± 0.009	–
f_5	0.072 ± 0.0101	0.058 ± 0.008	0.045 ± 0.004	YES
f_6	0.601 ± 0.222	0.315 ± 0.045	0.164 ± 0.010	–
f_7	0.303 ± 0.208	0.100 ± 0.019	0.060 ± 0.006	–
f_8	0.413 ± 0.061	0.142 ± 0.011	0.076 ± 0.004	–
f_9	0.325 ± 0.106	0.119 ± 0.020	0.076 ± 0.013	–
f_{10}	0.187 ± 0.010	0.093 ± 0.001	0.065 ± 0.001	–
f_{11}	0.081 ± 0.047	0.038 ± 0.007	0.028 ± 0.002	YES
f_{12}	0.277 ± 0.075	0.085 ± 0.004	0.055 ± 0.001	–
f_{13}	0.235 ± 0.068	0.113 ± 0.029	0.066 ± 0.009	–
f_{14}	0.057 ± 0.011	0.037 ± 0.010	0.036 ± 0.010	–
f_{15}	0.194 ± 0.025	0.091 ± 0.005	0.065 ± 0.006	–
f_{16}	0.435 ± 0.110	0.345 ± 0.137	0.273 ± 0.162	–
f_{17}	0.210 ± 0.062	0.099 ± 0.015	0.051 ± 0.009	–
f_{18}	0.283 ± 0.032	0.120 ± 0.016	0.082 ± 0.021	–
f_{19}	0.255 ± 0.043	0.094 ± 0.011	0.073 ± 0.019	–
f_{20}	0.360 ± 0.127	0.140 ± 0.006	0.076 ± 0.003	–
f_{21}	0.075 ± 0.011	0.032 ± 0.002	0.031 ± 0.008	–
f_{22}	0.058 ± 0.011	0.046 ± 0.017	0.045 ± 0.010	YES
f_{23}	0.252 ± 0.147	0.097 ± 0.011	0.087 ± 0.052	–
f_{24}	0.224 ± 0.074	0.067 ± 0.016	0.057 ± 0.013	–
f_{25}	0.218 ± 0.122	0.115 ± 0.010	0.065 ± 0.008	–
f_{26}	0.253 ± 0.087	0.068 ± 0.009	0.082 ± 0.069	–
f_{27}	0.142 ± 0.093	0.082 ± 0.032	0.065 ± 0.021	–
f_{28}	0.084 ± 0.063	0.033 ± 0.001	0.028 ± 0.013	–

et al. 2010), has been considered for the available dimensionality values, i.e. 10, 30, 50 and 100 variables. Numerical results confirming the same trend seen for SISC2010 and CEC2013 are reported in Tables 8.5 and 8.6 for Pearson and Spearman coefficients, respectively. For all 71 problems considered in this study, it appears clear that the correlation amongst variables (for both the ways it has been measured) tends to decay when the dimensionality increases. If one analyses the results related to the CEC2013, ς_{50D} is often smaller that the half of ς_{10D}. In addition, the decay in the correlation coefficient for those problems characterised by a strong correlation in 10 dimensions, e.g. f_8, tends to be steeper than that associated with problems that display a weak correlation in low dimensions, such as f_5. The test on the BBOB2010 functions shows that these functions are characterised by a weak correlation for all the dimensionality values. However, the study on scalability confirms that, also in this case, the correlation decreases with the increase in dimensions.

Table 8.4. Mean value and standard deviation for the Spearman correlation Index (φ) for CEC2013 over increasing dimensionality values.

	φ_{10D}	φ_{30D}	φ_{50D}	Separable
f_1	0.099 ± 0.010	0.085 ± 0.005	0.084 ± 0.001	YES
f_2	0.543 ± 0.053	0.293 ± 0.038	0.242 ± 0.030	–
f_3	0.267 ± 0.108	0.127 ± 0.023	0.112 ± 0.027	–
f_4	0.154 ± 0.061	0.121 ± 0.006	0.121 ± 0.013	–
f_5	0.113 ± 0.015	0.100 ± 0.006	0.091 ± 0.007	YES
f_6	0.543 ± 0.224	0.305 ± 0.075	0.187 ± 0.015	–
f_7	0.176 ± 0.268	0.117 ± 0.013	0.098 ± 0.005	–
f_8	0.409 ± 0.086	0.158 ± 0.010	0.108 ± 0.005	–
f_9	0.372 ± 0.178	0.150 ± 0.0201	0.107 ± 0.003	–
f_{10}	0.201 ± 0.021	0.120 ± 0.003	0.103 ± 0.004	–
f_{11}	0.132 ± 0.044	0.087 ± 0.003	0.084 ± 0.002	YES
f_{12}	0.257 ± 0.051	0.115 ± 0.007	0.097 ± 0.004	–
f_{13}	0.261 ± 0.066	0.121 ± 0.025	0.106 ± 0.005	–
f_{14}	0.093 ± 0.011	0.090 ± 0.013	0.092 ± 0.006	–
f_{15}	0.205 ± 0.031	0.119 ± 0.011	0.100 ± 0.010	–
f_{16}	0.354 ± 0.145	0.285 ± 0.022	0.245 ± 0.093	–
f_{17}	0.197 ± 0.069	0.117 ± 0.014	0.094 ± 0.005	–
f_{18}	0.264 ± 0.038	0.150 ± 0.019	0.115 ± 0.015	–
f_{19}	0.288 ± 0.059	0.124 ± 0.006	0.105 ± 0.010	–
f_{20}	0.319 ± 0.046	0.156 ± 0.008	0.109 ± 0.006	–
f_{21}	0.107 ± 0.012	0.088 ± 0.004	0.085 ± 0.002	–
f_{22}	0.100 ± 0.014	0.089 ± 0.009	0.091 ± 0.005	YES
f_{23}	0.219 ± 0.062	0.130 ± 0.029	0.122 ± 0.011	–
f_{24}	0.262 ± 0.131	0.110 ± 0.018	0.094 ± 0.010	–
f_{25}	0.197 ± 0.048	0.138 ± 0.010	0.102 ± 0.004	–
f_{26}	0.301 ± 0.010	0.152 ± 0.011	0.150 ± 0.075	–
f_{27}	0.138 ± 0.077	0.118 ± 0.032	0.105 ± 0.011	–
f_{28}	0.118 ± 0.012	0.086 ± 0.003	0.085 ± 0.002	–

As a general trend, optimisation problems with at least 100 dimensions seem characterised by a weak correlation. Optimisation problems in 500 and 1000 dimensions show a nearly null correlation amongst the variables. Regardless the dimensionality of the problem, can also be observed that the standard deviation for each function is nearly zero. This confirms the stability and reliability of the proposed coefficient, that is not affected by considerable fluctuations from the corresponding mean value. It is worth noticing that the standard deviation, rounded to 3 numerical places in tables, gets significantly small , i.e. $\simeq 10^{-3}$ or even zero, already in 100 dimension for most functions in SISC2010 (see Table 8.1 and 8.2) and all the functions in BBOB2010 (see Table 8.5 and 8.6).

As a practical consequence, if the variables of a problem are uncorrelated, the problem can be solved by perturbing each axis separately or, in an equivalent way, by decomposing the problem into many problems with low dimensions. These strategies are essentially the successful search moves used e.g. in (Tseng & Chen 2008), (Caraffini, Neri & Poikolainen 2013), (Caraffini, Neri, Iacca & Mol 2013),(Caraffini, Neri, Passow & Iacca 2013*a*), (Ros & Hansen 2008), and (Li & Yao 2012). On the basis of these results, it is possible to conclude that large scale problems on the one hand are more complex than low dimensional ones because the optimum lays in a very large space. On the other hand, they appear to be easier to solve than low dimensional problems as simple operators perturbing the variables separately can already detect solutions with a high performance. The phenomenon of the decrease of the correlation coefficients when the dimensionality increases is depicted in Fig.s 8.2, 8.3, 8.4, 8.5, 8.6 and 8.7. For each benchmark, two test problems have been selected, one being separable and the other being non-separable. It can be observed that all the trends are monotonically decreasing

Table 8.5. Mean value and standard deviation for the Pearson correlation index ς for BBOB2010 over increasing dimensionality values.

	ς_{10D}	ς_{30D}	ς_{50D}	ς_{100D}	Separable
f_1	0.054 ± 0.004	0.045 ± 0.035	0.028 ± 0.001	0.015 ± 0.000	YES
f_2	0.051 ± 0.006	0.034 ± 0.002	0.028 ± 0.001	0.015 ± 0.000	YES
f_3	0.054 ± 0.007	0.045 ± 0.040	0.030 ± 0.012	0.015 ± 0.000	YES
f_4	0.055 ± 0.007	0.045 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	YES
f_5	0.058 ± 0.004	0.035 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	YES
f_6	0.051 ± 0.006	0.042 ± 0.030	0.031 ± 0.015	0.015 ± 0.000	$-$
f_7	0.053 ± 0.005	0.033 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	$-$
f_8	0.055 ± 0.006	0.034 ± 0.002	0.028 ± 0.001	0.015 ± 0.000	$-$
f_9	0.054 ± 0.005	0.039 ± 0.015	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{10}	0.055 ± 0.007	0.034 ± 0.001	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{11}	0.051 ± 0.005	0.043 ± 0.034	0.029 ± 0.008	0.015 ± 0.000	$-$
f_{12}	0.053 ± 0.005	0.034 ± 0.002	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{13}	0.055 ± 0.007	0.035 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{14}	0.054 ± 0.007	0.034 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{15}	0.053 ± 0.007	0.035 ± 0.001	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{16}	0.051 ± 0.004	0.054 ± 0.046	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{17}	0.051 ± 0.004	0.035 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{18}	0.052 ± 0.006	0.039 ± 0.012	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{19}	0.052 ± 0.006	0.034 ± 0.001	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{20}	0.051 ± 0.006	0.034 ± 0.001	0.028 ± 0.001	0.015 ± 0.000	$-$
f_{21}	0.052 ± 0.006	0.034 ± 0.001	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{22}	0.051 ± 0.004	0.038 ± 0.013	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{23}	0.054 ± 0.007	0.035 ± 0.001	0.027 ± 0.001	0.015 ± 0.000	$-$
f_{24}	0.053 ± 0.005	0.033 ± 0.002	0.027 ± 0.001	0.015 ± 0.000	$-$

(except ς in Figure 8.3). In addition, it is shown that, while for separable functions the coefficients have low values already in 10 dimensions and further drop their values with the increase of dimensionality, see Figures 8.2, 8.5, and 8.6, non-separable functions are characterised by a major drop in the correlation coefficients when the dimensionality increases, see Figures 8.3, 8.4, and 8.7. Most importantly, it appears that, regardless of their values in 10 dimensions, both correlation coefficients tend towards zero in large scale cases. This fact can be seen, for example, by observing initial and final values of the trends in Figures 8.2 and 8.3 as well as the values in 10 and 1000 dimensions in Tables 8.1 and 8.2. More specifically while the values in 10 dimensions are diverse and range from about 0.05 to over 0.6, in 1000 dimensions the correlation coefficients are very similar to each other and all below 0.025. Similar considerations can be done for the two other benchmarks. Thus, the proposed experimental study suggests that large scale optimisation problems are all the same in terms of correlation among the variables and that all the LSOP are characterised by uncorrelated variables. Hence, LSOP do not necessarily require diagonal moves, i.e. search moves that involve the simultaneous perturbation of all the variables. On the contrary, LSOP can be solved by separately perturbing the variables independently, i.e. performing search moves along the axes.

8.4.1 Real world application case: Lennard-Jones Potential

In order to strengthen the validity of this study, the Pearson and Spearman correlation indices are here evaluated for a real world application, that is the LJP problem, filed as the problem number 2 of the CEC2011 test-bed, see (Swagatam & Suganthan 2010), and already presented in Chapter 7.2.6 for testing the SPAM algorithm.

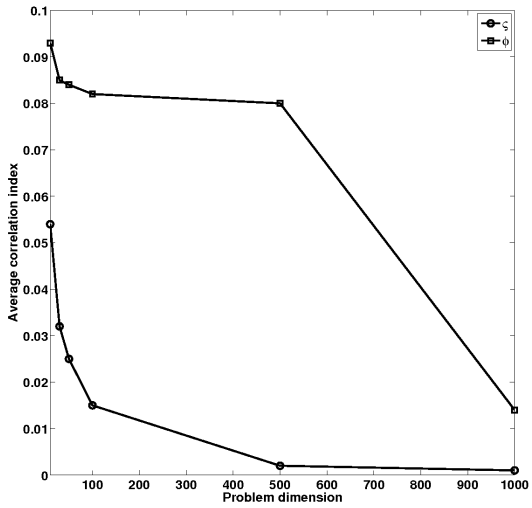


Figure. 8.2. Correlation coefficients for f_1 of SISC2010 over increasing dimensions.

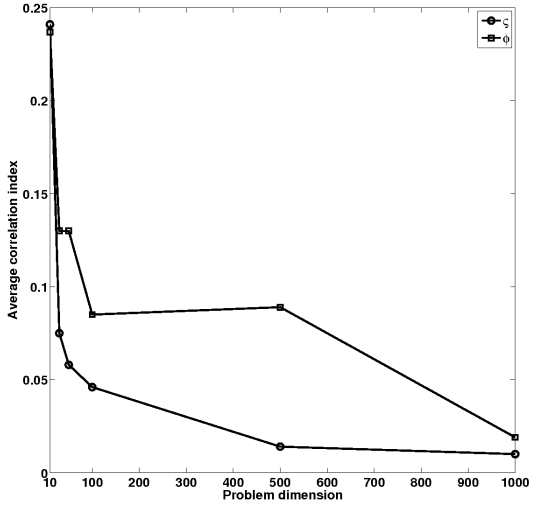


Figure. 8.3. Correlation coefficients for f_{13} of SISC2010 over increasing dimensions.

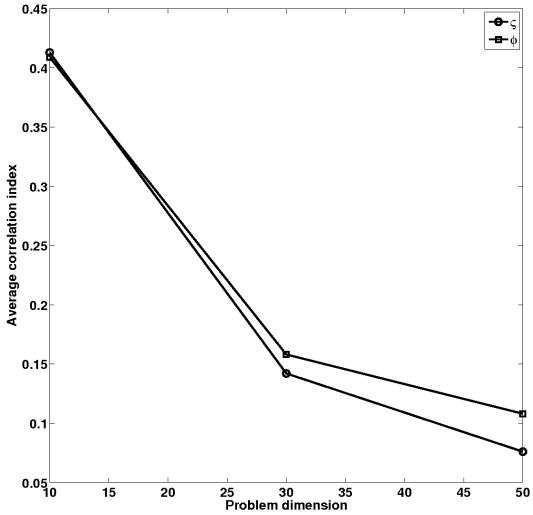


Figure. 8.4. Correlation coefficients for f_8 of CEC2013 over increasing dimensions.

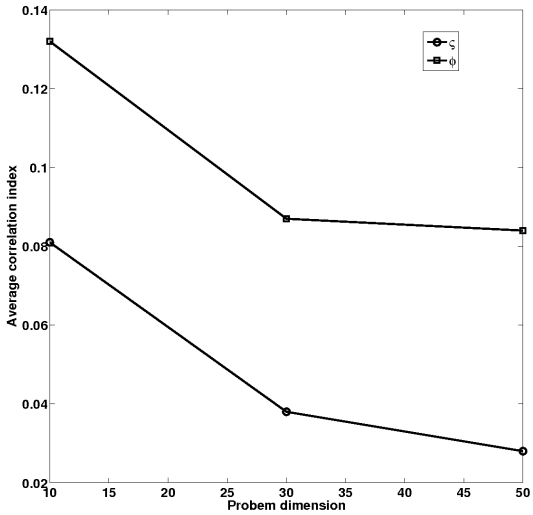


Figure. 8.5. Correlation coefficients for f_{11} of CEC2013 over increasing dimensions.

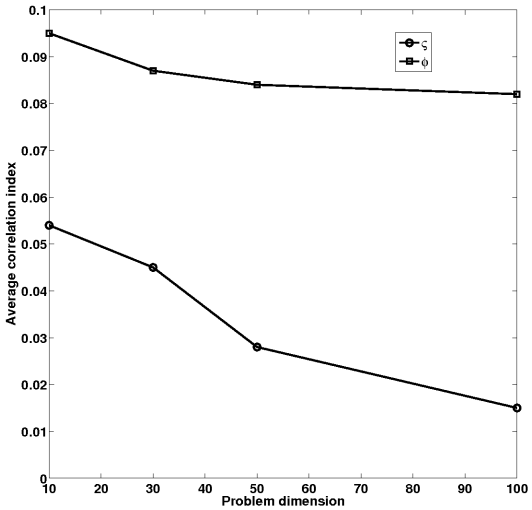


Figure. 8.6. Correlation coefficients for f_1 of BBOB2010 over increasing dimensions.

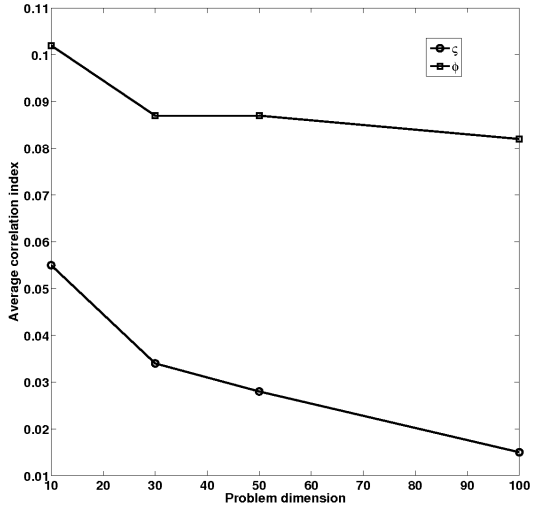


Figure. 8.7. Correlation coefficients for f_{10} of BBOB2010 over increasing dimensions.

Table 8.6. Meanvalue and standard deviation for the Spearman correlation index φ for BBOB2010 over increasing dimensionality values.

	φ_{10D}	φ_{30D}	φ_{50D}	φ_{100D}	Separable
f_1	0.095 ± 0.009	0.087 ± 0.003	0.084 ± 0.002	0.082 ± 0.001	YES
f_2	0.096 ± 0.008	0.087 ± 0.004	0.084 ± 0.002	0.081 ± 0.001	YES
f_3	0.096 ± 0.009	0.087 ± 0.003	0.084 ± 0.001	0.082 ± 0.001	YES
f_4	0.090 ± 0.008	0.087 ± 0.004	0.085 ± 0.002	0.081 ± 0.001	YES
f_5	0.092 ± 0.011	0.086 ± 0.003	0.084 ± 0.002	0.082 ± 0.001	YES
f_6	0.099 ± 0.016	0.084 ± 0.004	0.083 ± 0.002	0.082 ± 0.001	–
f_7	0.099 ± 0.013	0.087 ± 0.004	0.086 ± 0.001	0.081 ± 0.001	–
f_8	0.094 ± 0.010	0.086 ± 0.003	0.085 ± 0.002	0.081 ± 0.001	–
f_9	0.099 ± 0.010	0.089 ± 0.004	0.085 ± 0.001	0.082 ± 0.000	–
f_{10}	0.102 ± 0.018	0.087 ± 0.003	0.087 ± 0.008	0.082 ± 0.001	–
f_{11}	0.094 ± 0.011	0.087 ± 0.003	0.085 ± 0.001	0.082 ± 0.001	–
f_{12}	0.093 ± 0.014	0.088 ± 0.003	0.085 ± 0.002	0.082 ± 0.001	–
f_{13}	0.095 ± 0.011	0.086 ± 0.004	0.083 ± 0.002	0.081 ± 0.001	–
f_{14}	0.097 ± 0.009	0.088 ± 0.004	0.084 ± 0.001	0.082 ± 0.001	–
f_{15}	0.097 ± 0.008	0.086 ± 0.003	0.085 ± 0.001	0.082 ± 0.001	–
f_{16}	0.093 ± 0.013	0.085 ± 0.003	0.084 ± 0.002	0.082 ± 0.001	–
f_{17}	0.092 ± 0.013	0.087 ± 0.004	0.084 ± 0.003	0.082 ± 0.001	–
f_{18}	0.100 ± 0.012	0.089 ± 0.004	0.085 ± 0.003	0.082 ± 0.001	–
f_{19}	0.097 ± 0.008	0.087 ± 0.002	0.085 ± 0.002	0.082 ± 0.001	–
f_{20}	0.093 ± 0.009	0.086 ± 0.003	0.084 ± 0.001	0.082 ± 0.001	–
f_{21}	0.088 ± 0.008	0.087 ± 0.004	0.084 ± 0.002	0.082 ± 0.001	–
f_{22}	0.098 ± 0.010	0.086 ± 0.003	0.083 ± 0.002	0.082 ± 0.001	–
f_{23}	0.092 ± 0.011	0.091 ± 0.009	0.084 ± 0.003	0.082 ± 0.001	–
f_{24}	0.096 ± 0.008	0.086 ± 0.003	0.084 ± 0.002	0.082 ± 0.001	–

Table 8.7. Mean value and standard deviation for the Pearson and Spearman correlation indices for the Lennard-Jones Potential problem over increasing cluster size.

	2 atoms (6D)	12 atoms (36D)	22 atoms (66D)	32 atoms (96D)	42 atoms (126D)
ς	0.562 ± 0.081	0.152 ± 0.052	0.114 ± 0.050	0.053 ± 0.022	0.046 ± 0.026
φ	0.448 ± 0.096	0.171 ± 0.075	0.121 ± 0.024	0.111 ± 0.031	0.091 ± 0.006

The size of the atomic cluster determines the dimensionality of the problem (that is therefore scalable). For each added atom the dimensionality of the fitness function grows by 3 parameters. In this study, five configurations have been considered, i.e. $2, 12, 22, 32,$ and 42 atoms, corresponding to the dimensionality values of $6, 36, 66, 96,$ and 126 dimensions, respectively. The two presented tests have also been applied in this real-world case. Table 8.7 shows the correlation coefficients for the dimensionality values mentioned above and Fig. 8.8 illustrates the trends of the coefficients depending on the dimensionality. Numerical results show that, also in this case, the correlation coefficient decreases when the dimensionality increases. Thus, the large scale version of this real-world problem also appears characterised by uncorrelated variables. This result further confirms the finding obtained on the test problems, i.e. that large scale problems, regardless of their formulation, nature, and details, appear to have uncorrelated variables.

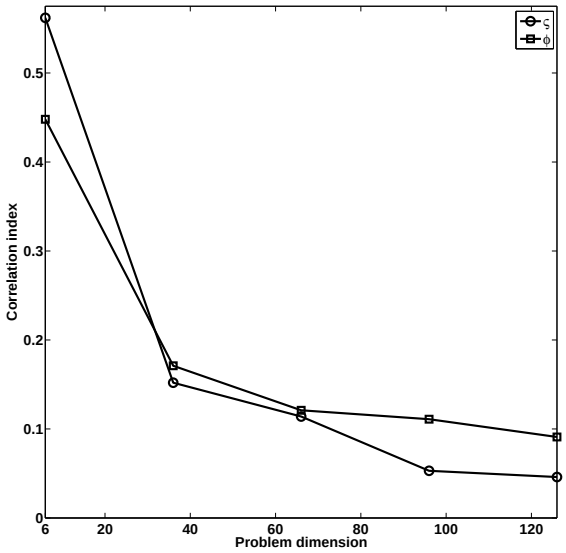


Figure. 8.8. Average correlation indices trends for the Lennard-Jones Potential problem of CEC2011 over increasing cluster size.

8.5 Chapter remarks

To summarise: 1) separable problems, unlike non-separable problems (especially in the case of composition functions) are characterised by low correlation indices; 2) regardless of the nature of the optimisation problem, the correlation decreases when the dimensionality of the problem increases; 3) for all the optimisation problems, the correlation appears to approach zero (i.e., there is no correlation among the variables) in large scale cases. In other words, numerical results of this study show that all the large scale optimisation problems are characterised by uncorrelated variables.

The scientific community can benefit from this study by considering that there is no use, in large scale optimisation, of complex operators that simultaneously perturb all the variables. On the contrary, simple operators that perturb small groups of variables or even the variables one-by-one can be efficient strategies to tackle large scale optimisation problems. Hence, high dimensionality is not only a “curse” but also a “blessing”. This piece of information is also relevant for the studies on automatic design of optimisation algorithms, as it allows us to extract some indications on the search strategy to implement, on the basis of the dimensionality of the problem.

Chapter 9

Conclusions and future developments

This work introduced novel optimisation algorithms designed by following the Ockham's Razor principle in MC, that despite their simplicity, see (Caraffini, Neri, Passow & Iacca 2013*b*), (Caraffini, Neri, Gongora & Passow 2013) and (Iacca, Caraffini, Neri & Mininno 2013), displayed remarkable performances on a vary set of benchmark problems (as well as on real-world engineering applications). A great deal of attention has been given to the concept of algorithmic structure, and a new methodology for designing optimisation algorithm by combining elementary (atomic) structures has been formulated. Simple optimisers consisting of the combinations of basic operators in elementary structures, i.e. sequential and parallel (Caraffini, Neri, Iacca & Mol 2012), have been tested and proven to be promising and efficient for tackling specific aspects characterising an optimisation problem, such as separability of the objective function, dimensionality of the problem or ill-conditioned function landscapes. Numerical data have confirmed the point that the presented approach can provide similar, or even better solutions with respect of those obtained by applying complex popular optimisation algorithms. In this light, it is possible to conclude that the algorithmic complexity in optimisation can be kept low, since extremely complex structures are not necessarily supported by extremely good results. The most significant examples supporting this point are given in Chapter 6. The RIS and RS algorithms in particular showed competitive performances and successfully addressed **IRQ I-III**. The parallel structure described in Chapter 7.1, also showed good results on a very set of problems. The proposed PMS algorithm presents a more robust structure with respect to the RIS, making it more versatile and reliable for general purpose optimisation.

Another significant outcome of this piece of work, in terms of potential future impact in the field, is represented by the presentation of a general framework for automatically combining elementary structures in order to tackle optimisation problems defined in continuous domains (Caraffini et al. 2014). The general framework considered a set of interacting software components, able to analyse the landscape of the problem at hand, selects the required operators from a pool accordingly, builds the connections between the operators (algorithm design) and solves the given problem. The actual implementation can only analyse the problem in terms of dimensionality and separability/non-separability of the landscape, thus addressing **IRQ IV**. Three operators were initially considered in the pool and combined for building elementary structures forming the

optimisation algorithm. Numerical results confirmed the efficiency of the proposed framework, which however needs further improvements in order to reach the big picture of a generic automatic solvers designer, able to handle multiple features at the same time. The combination of more features can indeed make the choice of the right component (or interacting components) quite difficult.

The last interesting achievement of this piece of research is a more theoretical finding regarding the correlation of linked design variables in large scale optimisation problems. The fact that a large scale optimisation problem presents a low interconnection degree between the design variables, at least for the test suits used in this work, is an essential information to bear in mind while facing a real-world problem in many dimensionality values. In effect, this piece of information suggested the conclusion that these problems can be efficiently addressed with the same simple operators used for separable problems in low dimensions, such as S. It also explained why successful strategies for handling large scale optimisation make use of perturbation of a single design variable at time, or anyway of a subgroup of few variables (**IRQ IV-VI**). The study on the correlation among the variables still requires improvements, but appears to be promising and essential also for the realisation of the platform for the automatic design of optimisation algorithms. Future developments in this direction will have to take this last aspect into consideration.

The innovative nature of this project within the subject is contained in the approach itself, since this would be the first software platform for optimisation that designs algorithms on the basis of an analysis on the problem. The vast majority of current adaptive approaches for building up meta-heuristics are oriented towards discrete optimisation problems, such as in the case of hyper-heuristics, SATzilla etc.

On the contrary, automatic design for real-valued (continuous) optimisation problems has not been intensively studied yet, and a platform for automatically designing algorithms to tackle real-valued optimisation problems does not yet exist.

9.1 Future work

In order to pursue this aim, the proposed framework needs to be further and extensively improved. A preliminary study on multi-modal landscapes has already been done and will be more investigated. The following four points, containing new implementations and testing, but also the replacement of some limitations of the actual presented work, will be addressed in the near future:

1. To investigate and design fast tests that can measure or reliably estimate the important features that characterize optimisation problems and must be taken into account when an algorithm is designed (Problem Analyser). Not only separability and problem dimensionality, but also multi-modality and ill-conditioning must be considered. Moreover, the current employment of the CMA-ES algorithm, even though efficient, results in a huge algorithmic temporal overhead, which could be reduced. Possible future methodologies for extracting information about inter-linkages among the design variables will consider other techniques, e.g. the principal component analysis, the fourier transform and linkage

learning methods.

2. To prepare a database of algorithmic operators and identify their role/appropriateness within the context of the selected features (Pool of Operators). The current pool of operator must be obviously extended. Potential operators that can be easily tested and inserted in the pool are the LS routine in Chapter 2, but further operators will be both taken from the literature and implemented ad-hoc.

3. To design a criterion for selection of the operators based on cross labelling between multiple problem features and operator action (Operator Selector). This point will be faced by means of the MemeNet network (Chapter 5.1), with extensive testing of the considered operators in order to be able to understand their behaviour over a variety of problems.

4. To develop an Algorithmic Builder that connects operators grouped within elementary structures.

While some of these points can be easily obtained by extending the proposed framework, the last one seems to be more challenging and will have to be carefully and intensively addressed, both theoretically and empirically. The Automatic Builder will have to be able to consider all the extracted features at the same time, and this part has not been covered in the presented piece of researcher.

The creation of a fully general purpose software platform is obviously an extraordinarily broad and ambitious project, that would certainly require years of work and investigation in several fields of computer science and software engineering. The potential impact of this research is high and the preliminary carried out work, presented in this thesis, makes think that if further studied and extended in the future, this project can guarantee ground-breaking results.

Appendix A

Statistical methods for comparing algorithms

In order to strengthen the interpretation of the numerical results, in this work, two statistic tests namely Wilcoxon Rank-Sum Test (Wilcoxon 1945) and Holm Bonferroni Procedure (Holm 1979) have been used to compare between the algorithms. An optimisation algorithm acts as a stochastic process whose realisation for a certain problem is a number expected to be as close as possible to the optimal solution. Thus, an adequate and rigorous analysis of its performance must be given statistically. The aforementioned methods refer to two non-parametric (distribution free) statistical procedures, since no assumption of Normality can be made, operating in different ways: the former performs a single hypothesis test (a single optimisation problem is considered) between each pair of algorithms, while the latter handles multiple hypotheses simultaneously.

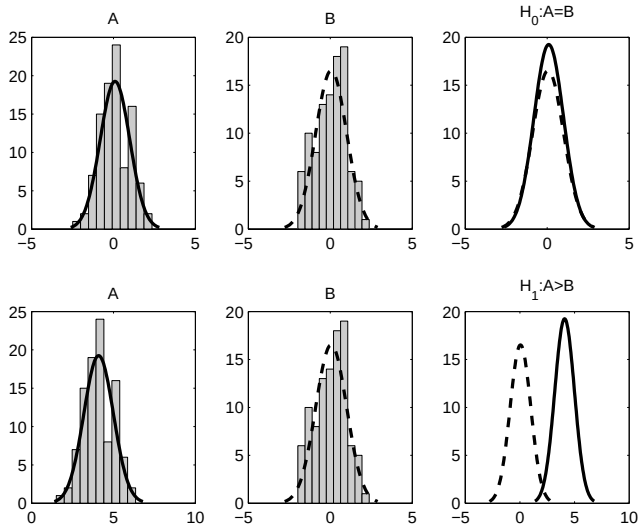
In particular, the so called *null-hypothesis* is examined and then accepted if a relationship between data from two different algorithms exists. More formally, suppose to have the final values distribution[1] of two algorithms A and B, containing n_A and n_B run respectively. The *null-hypothesis* $H_0 : A = B$ is verified when the elements of A and B are independent samples of similar continuous distributions having the same means (see Figure A). Otherwise, the two distributions differ and H_0 is rejected against the *alternative-hypothesis* which can either be $H_1 : A \geq B$ or $H_1 : A \leq B$ (or simply $H_1 : A \neq B$ when it is not clear which sample has the lowest mean).

It must be said that these tests have been chosen since they are non-parametric, meaning that they do not require any specific form or assumption for the distribution of the numeric values. When all the conditions for parametric tests are met, non parametric tests are less effective, but the opposite situation occurs in presence of outliers[2] or non Gaussian distributions, thus making the proposed tests more reliable for what concerns this piece of work. They are indeed suitable for evaluating the performance of algorithms, as stated in (García, Fernández, Luengo & Herrera 2008). Both of them belong to the class of statistical ranking methods, where scores (ranks) are assigned and then used instead of the actual numerical data in order to estimate and analyse their distribution. A brief description of these methods is provided in the following sections.

[1] Set containing the best value provided by the algorithm on each run.

[2] Observation which is numerically distant from the rest of the data.

Figure. A.1. Graphical representation of the Null-hypothesis $H_0 : \ A = B$ VS $H_1 : A \geq B$ accepted (top) or rejected (bottom)



A.1 Wilcoxon Rank-Sum Test

The Wilcoxon Rank-Sum test and the Mann-Whitney U test (Mann, Whitney et al. 1947), are the non-parametric counterpart of the popular two-sample unpaired t-test used to check for a difference between two samples. Even though the Wilcoxon Rank-Sum and the Mann-Whitney U tests are often confused, it must be noted that the original version of the first one makes use of means, as a measure of the centre of location between the two distributions, while the second uses medians. Apart from this clarification, both the tests share the same idea and it is indeed possible to use any index of central location arbitrarily. The median for instance is reliable in the presence of outliers but, for the sake of coherence, in this work we adopted the mean. All the numerical results of this thesis are in effect displayed in terms of average value and standard deviation, and outliers have been averaged on purpose in order to consider the reliability of stable and robust algorithms with respect to those whose performance highly depends on the initial solution or other initial conditions.

The Rank-Sum procedure tests the *null-hypothesis* against the *alternative-hypothesis* by performing the following steps. Starting from two samples A and B, assumed to be independent, a second set of $N = n_A + n_B$ elements is sorted in order to assign to each observation a rank: the smallest value has rank 1, the second smallest has rank 2 and so on until rank N. In case of ties[3], the rank of the tied values can be handled by replacing their ranks with their average rank (e.g. if four observations have ordinal ranks $4, 5, 6, 7$ but they have equal values, they are assigned the same rank $(4 + 5 + 6 + 7)/4 = 5.5$). Without a loss of generality let us consider the first set A as reference, the corresponding sum w_A of the ranks for this sample is called "Wilcoxon rank sum statistic" while the relative random variable will be denoted with W_A. It now possible to calculate the

[3]Observations with equal numerical value in the data set.

P-value and decide whether or not to accept the *null-hypothesis*, but it is necessary to distinguish between two cases. In the case of the "One-Sided" Wilcoxon Runk-Sum test there are two possible *alternative-hypotheses*: $H_1 : A \geq B$ or $H_1 : A \leq B$ so the associated P-value can be defined as P-value $=$ prob$\{W_A \geq w_A\}$ or P-value $=$ prob$\{W_A \leq w_A\}$ respectively. Conversely, for a "Two-Sided" test, i.e. testing $H_0 : A = B$ versus the alternative $H1 : A \neq B$, the P-value is double the probability of falling into the tail of the distribution closest to w_A. Thus, if w_A is in the lower tail then P-value $=$ prob$\{W_A \leq w_A\}$, otherwise P-value $=$ prob$\{W_A \geq w_A\}$. A high P-value gives no evidence against the *null-hypothesis*, while a small one suggests that two distributions are somehow shifted and if it is small enough, H_0 has no validity. More precisely, *null-hypothesis* is rejected when the P-value is smaller than a certain significance level α usually set to 0.05.

In order to calculate the P-value, the distribution of W_A must be considered. For small sample size (up to 10-20), this probability distribution is tabulated, otherwise it can be approximated using the normal distribution $\mathcal{N}(\mu_A, \sigma_A)$ where:

$$\mu_A = \frac{n_A \cdot (N+1)}{2} \quad \text{and} \quad \sigma_A = \sqrt{\frac{n_A \cdot n_B \cdot (N+1()}{12}} \tag{A.1}$$

The probability can then be easily evaluated via numerical integration.

A.2 Holm-Bonferroni Test

The second statistical method employed in this work, called Holm-Bonferroni, is a non-parametric test proposed by Sture Holm (Holm 1979) in order to face the problem of multiple hypotheses. The name of this procedure is due to the fact that Holm decided to employ the "Bonferroni correction" in order to adjust the P-values, see (Abdi 2010), so controlling the Familywise Error Rate (FWER)[4]. It is in fact important to keep the FWER as low as possible while considering a set of multiple statistical inferences simultaneously. The higher the number of hypotheses considered, the higher the probability of misinterpreting the *null-hypothesis*. The Holm-Bonferroni procedure improves upon the old Bonferroni procedure on this issue, and has been shown to be reliable even with many hypotheses, i.e. at least 76 problems were tested simultaneously in each paper in appendix C. The Holm-Bonferroni procedure consists of the following. Given the final values of each run of a set of N_A algorithms over a set of N_{TP} test functions, the algorithms are ranked on the basis of their average performance calculated over each problem. More specifically, a rank $R_{k,i}$ is assigned for each algorithm ($i = 1, \ldots, N_A$) in the following way: for each problem $k = 1, \ldots, N_{TP}$, a score of N_A is given to the algorithm displaying the best mean value, $N_A - 1$ is attributed to the second best, $N_A - 2$ to the third and so on. The algorithm displaying the worst performance scores 1. For each algorithm, the scores obtained on the problems are summed up and averaged over the amount of test problems (N_{TP}) obtaining R_i:

$$R_i = \frac{\sum_{k=1}^{N_{TP}} R_{k,i}}{N_{TP}} \quad \forall i \in \{1, \ldots, N_A\} \tag{A.2}$$

[4]In statistics, familywise error rate refers to the probability of making one or more false discoveries among all the hypotheses under consideration.

On the basis of these scores the values z_j can be calculated. In order to compute this operation it is first necessary to select one of the N_A algorithms as a reference and rename its score to R_0. Obviously, the remaining algorithms must be re-sorted from best to worst and renamed accordingly: R_j for $j = 1, \ldots, N_A - 1$. z_j can be calculated as:

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A \cdot (N_A+1)}{6 \cdot N_{TP}}}} \tag{A.3}$$

By means of the z_j values, the corresponding cumulative normal distribution values p_j are calculated. These p_j values are then compared with the corresponding δ/j where δ is the significance level, usually set to 0.05. As long as the condition $p_i < \delta/(N_A - j)$ holds, the null-hypothesis (that the two algorithms have indistinguishable performances) is rejected, i.e. the reference algorithm statistically outperforms the selected j-th algorithm. This repeated check stops as soon as the above condition fails, meaning that the j-th null-hypothesis is accepted, i.e. the two algorithms statistically have the same performance. At the end of this procedure, it is then possible to compare between the algorithms looking at the *null-hypothesis*, accepted/rejected, and at the rank. In case of rejection, the position of an algorithm (in terms of rank) is useful to understand whether the reference has slightly, significantly or evidently outperformed the comparison algorithm.

Appendix B

No Free Lunch Theorem

The NFL theorem for optimisation problems, first introduced in 1997 by David Wolpert and William G. Macready (Wolpert & Macready 1997), was a turning point in CIO. During the '80s and the '90s researchers used to believe that meta-heuristics, and in particular EAs, were better than all the other algorithms and that the research in this field would have led to an universal algorithm able to outperform the others for all possible problems. Conversely, being a general-purpose framework, their average performance for all possible optimisation problems cannot be otherwise than the same for every arbitrary pair of optimisers. In more detail, the bigger the size of applicability the lesser the efficiency in solving a specific problem. This interesting trade-off implies that the best optimiser for a given problem is the one specifically designed for that problem. Obviously, an algorithm can be tailored to face a single problem only when *a priori* knowledge is available, otherwise only a general-purpose scheme can be employed. More formally, assuming that algorithm never re-evaluates a candidate solution and that all objective functions f are equally likely to be used as input for each arbitrary pair of algorithms A and B, the theorem can be formulated as follow:

$$\sum_{f} P(\mathbf{x_m} \big| m, f, A) = \sum_{f} P(\mathbf{x_m} \big| m, f, B) \tag{B.1}$$

where the performance of the algorithm A/B is expressed in terms of probability of detecting the optimal solution $\mathbf{x_m}$ when iterated m times. For the sake of completeness it must be said that this formula, also called 1st NFL theorem, describes the case of static optimisation, i.e. the fitness function does not vary during the optimisation process, but there even exists a second formulation for dynamic optimisation. As with every theorem, NFL's validity largely depends on the validity of its hypotheses. It has been pointed out that for continuous optimisation the aforementioned hypotheses are too stringent, and in particular the assumption of the non-revisiting nature of the algorithms does not always hold, breaking the reliability of this theorem (Auger & Teytaud 2010). In actual fact, to make an algorithm non-revisiting is quite straightforward, i.e. an archive containing all the generated solutions must be kept updated in order to be able to check whether a solution must be discarded and regenerated. However, as a side effect this would make the algorithm unusable due to the high time and memory overhead. Auger and Teytaud have then mathematically proven that for continuous domains "continuous free lunches exist". On the other hand, it can be argued that in Computer Science continuous problems do not exist since is impossible to store an infinitely dense set within a digital device, and the NFL theorem for discrete problems has not yet been disproved.

Appendix C

Complete list of publications produced during my PhD

During my doctoral studies 7 scientific articles have been published in peer reviewed journals and 11 in international conference proceedings. The complete list of articles is reported below:

[1] Caraffini, F., Iacca, G., Neri, F. & Mininno, E. (2012a), The importance of being structured: A comparative study on multi stage memetic approaches, in 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1-8.

[2] Caraffini, F., Iacca, G., Neri, F. & Mininno, E. (2012b), Three variants of three stage optimal memetic exploration for handling non-separable fitness lscapes, in 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1-8.

[3] Caraffini, F., Iacca, G., Neri, F., Picinali, L. & Mininno, E. (2013), A cma-es super-fit scheme for the re-sampled inheritance search, in 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1123-1130.

[4] Caraffini, F., Neri, F., Cheng, J., Zhang, G., Picinali, L. & G. Iacca, E. M. (2013), Super-fit multicriteria adaptive differential evolution, in 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1678-1685.

[5] Caraffini, F., Neri, F., Gongora, M. & Passow, B. (2013), Re-sampling search: A seriously simple memetic approach with a high performance, in 'IEEE Symposium Series on Computational Intelligence, Workshop on Memetic Computing', pp. 52-59.

[6] Caraffini, F., Neri, F., Iacca, G. & Mol, A. (2013), 'Parallel memetic structures', *Information Sciences* **227**(0), 60-82.

[7] Caraffini, F., Neri, F., Passow, B. N. & Iacca, G. (2013), 'Re-sampled inheritance search: high performance despite the simplicity', *Soft Computing* pp. 1-22.

[8] Caraffini, F., Neri, F. & Picinali, L. (2014), 'An analysis on separability for memetic computing automatic design', Information Sciences **265**(0), 1-22.

[9] Caraffini, F., Neri, F. & Poikolainen, I. (2013), Micro-differential evolution with extra moves along the axes, in 'IEEE Symposium Series on Computational Intelligence, Symposium on Differential Evolution', pp. 46-53.

[10] Iacca, G., Caraffini, F. & Neri, F. (2012), 'Compact differential evolution light: high performance despite limited memory requirement modest computational overhead', Journal of Computer Science Technology **27**(5), 1056-1076.

[11] Iacca, G., Caraffini, F. & Neri, F. (2013), 'Memory-saving memetic computing for path-following mobile robots', Applied Soft Computing **13**(4), 2003-2016.

[12] Iacca, G., Caraffini, F. & Neri, F. (2014), 'Multy-strategy coevolving aging particle optimisation', International Journal of Neural Systems **24**(01), 1450008.

[13] Iacca, G., Caraffini, F., Neri, F. & Mininno, E. (2012), Robot base disturbance optimisation with compact differential evolution light, in 'EvoApplications', pp. 285-294.

[14] Iacca, G., Caraffini, F., Neri, F. & Mininno, E. (2013), Single particle algorithms for continuous optimisation, in 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1610-1617.

[15] Iacca, G., Neri, F., Caraffini, F., & Suganthan, P. N. (2014), A differential evolution framework with ensemble of parameters strategies pool of local search algorithms, in 'EvoApplications', p. To appear.

[16] Neri, F., Weber, M., Caraffini, F. & Poikolainen, I. (2012), Meta-lamarckian learning in three stage optimal memetic exploration, in 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1-8.

[17] Poikolainen, I., Caraffini, F., Neri, F. & Weber, M. (2012), Handling non-separability in three stage memetic exploration, in 'Proceedings of the Fifth International Conference on Bioinspired Optimisation Methods their Applications', pp. 195-205.

[18] Poikolainen, I., Iacca, G., Caraffini, F. & Neri, F. (2013), Focusing the search: a progressively shrinking memetic computing framework', *International Journal of Innovative Computing Applications* **5**(3), 127-142.

Appendix D

Parameters setting

In this appendix is reported the complete list of parameters values used for generating the numerical results presented in this thesis. By strictly following the implementation proposed in the pseudo-codes together with this parameters setting, it is possible to reproduce similar results and so drawing the same conclusions achieved in this piece of work. For the sake of clarity, it must be said that the same exact numerical values cannot be replicated due to the stochastic nature of meta-heuristic approaches.

- Brent's line search (Brent 1973) implemented and configured as in (Press 2007). : $CGOLD = 0.381966$, $myZeps = 10^{-3}$, $tol = 3 \cdot 10^{-8}$ and $xmin = fmin = 0$.
- Nelder-Mead (Nelder & Mead 1965),with reflection coefficient $\alpha = 1$, contraction coefficient $\beta = 0.5$, expansion coefficient $\gamma = 2$ and shrinkage coefficient $\delta = 0.5$.
- Rosenbrock (Rosenbrock 1960), with positive perturbation factor $\alpha = 2$, negative perturbation factor $\beta = -0.5$ and threshold for coordinate system rotation $\epsilon = 10^{-5}$.
- G-CMA-ES (Auger & Hansen 2005), with initial population $\lambda_{start} = 10$ and factor for increasing the population size equal to 2.
- cGA (Mininno, Cupertino & Naso 2008*b*), with virtual population size equal to 300.
- ISPO (Zhen et al. 2010), with acceleration $A = 1$, acceleration power factor $P = 10$, learning coefficient $B = 2$, learning factor reduction ratio $S = 4$, minimum threshold on learning factor $E = 1e - 5$, and particle learning steps $PartLoop = 30$.
- CLPSO (Liang et al. 2006), with population size equal to 60 individuals.
- CCPSO2 (Li & Yao 2012), with population size equal to 30 individuals, Cauchy/Gaussian-sampling selection probability $p = 0.5$ and set of potential group sizes $S = \{2, 5, 10\}$, $S = \{2, 5, 10, 50, 100\}$, $S = \{2, 5, 10, 50, 100, 250\}$ for experiments in 30, 100 and 1000 dimensions, respectively.

- cPSO (Neri et al. 2013), with virtual population size of 300 individuals, $\phi_1 = -0.2$, $\phi_2 = -0.07$, $\phi_3 = 3.74$, $\gamma_1 = 1$, and $\gamma_2 = 1$.
- SADE (Qin et al. 2009), with Learning Period $LP = 20$ and population size $N_p = 50$.
- JADE (Zhang & Sanderson 2009), with population size equal to 60 individuals, group size factor $p = 0.05$ and parameters adaptation rate factor $c = 0.1$.
- jDE (Brest et al. 2006), with $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = \tau_2 = 0.1$ and population size $N_p = 50$.
- MDE-pBX (Islam et al. 2012), with population size equal to 100 individuals and group size q equal to 15% of the population size.
- cDE (Mininno et al. 2011) and cDE-light (Iacca, Caraffini & Neri 2012), with virtual population size $N_p = 300$, scale factor $F = 0.5$, and $\alpha_m = 0.25$.
- MA-LSCh-CMA (Molina, Lozano, García-Martínez & Herrera 2010*a*) with population size equal to 60 individuals, probability of updating a chromosome by mutation equal to 0.125, local/global search ratio $r_{\frac{L}{G}} = 0.5$, BLX-α crossover with $\alpha = 0.5$, n_{ass} parameter for Negative Assortative Mating set to 3, LS intensity stretch $I_{str} = 500$ and threshold $\delta_{LS}^{min} = 10^{-8}$.
- MA-LSCh-SSW (Molina, Lozano & Herrera 2010), with population size equal to 100 individuals, probability of updating a chromosome by mutation equal to 0.125, local/global search ratio $r_{\frac{L}{G}} = 0.5$, BLX-α crossover with $\alpha = 0.5$, n_{ass} parameter for Negative Assortative Mating set to 3, LS intensity stretch $I_{str} = 500$ and threshold $\delta_{LS}^{min} = 0$.
- 3SOME (Iacca, Neri, Mininno, Ong & Lim 2012), with inheritance factor $\alpha = 0.05$, middle distance exploration hyper-cube size Δ equal to 20% of the total decision space width, coefficient of generated points at each activation of the middle distance exploration $k = 4$, short distance exploration radius $\delta = 0.4$ and local budget fixed to 150 iterations.
- RIS-3SOME (Caraffini, Iacca, Neri & Mininno 2012*b*), was executed with the same parameter setting of 3SOME, while the DE/current-to-rand/1 mutation was applied with scale factor $F = 0.4$, and the threshold ε was set equal to $1e-4$.
- μDE-3SOME (Caraffini, Iacca, Neri & Mininno 2012*b*), was executed with the same values of 3SOME, plus $m = 5$ individuals, scale factor $F = 0.75$, and number of DE iterations equal to the problem dimension n.
- (1+1)-CMA-ES-3SOME (Caraffini, Iacca, Neri & Mininno 2012*b*), The same values of α_e and ρ where used also in . As for (1+1)-CMA-ES, the standard values used in its original Java implementation available on (Hansen 2012) and suggested in the original paper were used, namely $c_p = 1/12$, $p_{succ}^{target} = 2/11$, $p_{thresh} = 0.44$ and $\sigma_0 = 1$. The budget for each activation of (1+1)-CMA-ES was set to $10 \times n$.

- RS (Caraffini, Neri, Gongora & Passow 2013), with initial search radius $\delta[i]$ set equal to the 40% of the domain width along each variable i, and the stop-threshold ε set equal to 10^{-6}.

- RIS (Caraffini, Neri, Passow & Iacca 2013*a*), with inheritance factor α, eq. (3.22), has been equal to 0.5. Regarding the local searcher, the initial search radius $\delta[i]$, has been set equal to the 40% of the domain width along each variable i, while the stop-threshold ε has been set equal to 10^{-6}.

- VISPO (Iacca, Caraffini, Neri & Mininno 2013), with learning period $LP = 10$, number of perturbations for each design variable $H = 30$ and learning threshold 0.65.

- 3SOME-Rosenbrock (Caraffini, Iacca, Neri & Mininno 2012*a*), with inheritance factor $\alpha = 0.05$, middle distance exploration hyper-cube size Δ equal to 20% of the total decision space width, coefficient of generated points at each activation of the middle distance exploration $k = 4$. As for Rosenbrock the following parameters are required: initial step size $h = 0.1$, threshold $\varepsilon = 10^{-8}$, forward factor $\alpha = 2$ and backward factor $\beta = 0.5$.

- 3SOME-Powell (Caraffini, Iacca, Neri & Mininno 2012*a*), with inheritance factor $\alpha = 0.05$, middle distance exploration hyper-cube size Δ equal to 20% of the total decision space width, coefficient of generated points at each activation of the middle distance exploration $k = 4$. As for Powell the fitness tolerance f_{tol} is set equal to 10^{-5}, while GSS is applied with bounds $[-100, 100]$ and a local budget of 20 fitness evaluations.

- PMS (Caraffini, Neri, Iacca & Mol 2013), with inheritance factor α set equal to 0.95, initial search radius δ equal to 0.4 and computational budget for the first local searcher set to 150 iterations. Regarding Rosenbrock, see above, the original setting has been kept: $\alpha = 2$, $\beta = 0.5$ and $\varepsilon = 10^{-5}$.

- SPAM (Caraffini et al. 2014), with inheritance factor $\alpha = 0.5$ for performing the re-sampling procedure, while for the local search, R has been used with $\alpha = 2$, $\beta = 0.5$, $\varepsilon = 10^{-5}$ and h initialised as a vector of 0.1. Regarding S, the initial search radius δ, it has been set equal to 40% and the local budget to 150 functional calls.

- CA-ILS (Nguyen & Yao 2008), with population size $\mu = 3$, maximum moves per individual $\eta = 5$, acceptance rate $\rho = 0.3$, initial move length proportion $\tau = 0.02$, and move length constraint $s = 0$. The reduction rate β for the global temperature T have been set equal to 0.838 and 100 respectively. As local search, in this study we made use of SPSA (Spall 1987), with $a = 0.5$, $A = 1$, $\alpha = 0.602$, $c = 0.032$, $\gamma = 0.1$ and $\epsilon = 0.01$.

Appendix E

Numerical Results

For the sake of readability, extended tables are grouped in this appendix and organised in subsections. Data are displayed as explained in the foreword of this thesis, and obtained according to the experimental set-up in Chapter 5 and parameters setting in Appendix D.

E.1 Extended numerical results for 3SOME variants

Table E.1. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = ML-3SOME) for ML-3SOME against 3SOME on BBOB2010 in 10 (a) and 40 (b) dimensions.

(a) BBOB2010 in 10 dimensions

	ML3SOME	3SOME	
f_1	$\mathbf{7.95e+01 \pm 1.22e-14}$	$\mathbf{7.95e+01 \pm 1.21e-14}$	=
f_2	$\mathbf{-2.10e+02 \pm 1.58e-14}$	$\mathbf{-2.10e+02 \pm 1.63e-14}$	=
f_3	$\mathbf{-4.61e+02 \pm 2.77e+00}$	$-4.61e+02 \pm 1.18e+00$	+
f_4	$\mathbf{-4.60e+02 \pm 4.22e+00}$	$-4.60e+02 \pm 1.39e+00$	+
f_5	$\mathbf{-9.21e+00 \pm 5.42e-14}$	$5.33e+00 \pm 2.91e+01$	+
f_6	$\mathbf{3.59e+01 \pm 3.81e-03}$	$8.25e+01 \pm 2.83e+02$	=
f_7	$\mathbf{1.03e+02 \pm 7.31e+00}$	$1.05e+02 \pm 1.23e+01$	=
f_8	$1.49e+02 \pm 1.89e-01$	$\mathbf{1.49e+02 \pm 1.86e-01}$	-
f_9	$\mathbf{1.24e+02 \pm 9.47e-01}$	$1.25e+02 \pm 1.69e+00$	+
f_{10}	$\mathbf{3.13e+02 \pm 1.64e+02}$	$3.95e+03 \pm 2.63e+04$	+
f_{11}	$1.60e+02 \pm 3.21e+01$	$\mathbf{1.57e+02 \pm 3.36e+01}$	=
f_{12}	$-6.02e+02 \pm 2.32e+01$	$\mathbf{-6.12e+02 \pm 1.33e+01}$	-
f_{13}	$\mathbf{4.08e+01 \pm 9.36e+00}$	$4.26e+01 \pm 1.28e+01$	=
f_{14}	$-5.23e+01 \pm 2.41e-05$	$\mathbf{-5.23e+01 \pm 3.05e-05}$	-
f_{15}	$\mathbf{1.07e+03 \pm 4.10e+01}$	$1.10e+03 \pm 6.38e+01$	+
f_{16}	$\mathbf{7.83e+01 \pm 4.25e+00}$	$7.97e+01 \pm 4.63e+00$	+
f_{17}	$\mathbf{-1.31e+01 \pm 2.74e+00}$	$-1.03e+01 \pm 6.57e+00$	+
f_{18}	$\mathbf{-3.60e+00 \pm 1.06e+01}$	$5.80e+00 \pm 2.56e+01$	+
f_{19}	$\mathbf{-9.93e+01 \pm 1.72e+00}$	$-9.80e+01 \pm 2.98e+00$	+
f_{20}	$-5.46e+02 \pm 2.99e-01$	$\mathbf{-5.46e+02 \pm 2.59e-01}$	=
f_{21}	$\mathbf{5.05e+01 \pm 1.14e+01}$	$5.36e+01 \pm 1.34e+01$	=
f_{22}	$\mathbf{-9.90e+02 \pm 1.33e+01}$	$-9.88e+02 \pm 1.55e+01$	=
f_{23}	$\mathbf{7.80e+00 \pm 4.53e-01}$	$7.86e+00 \pm 4.95e-01$	=
f_{24}	$\mathbf{1.71e+02 \pm 2.80e+01}$	$1.92e+02 \pm 4.46e+01$	+

(b) BBOB2010 in 40 dimensions

	ML3SOME	3SOME	
f_1	$\mathbf{7.95e+01 \pm 1.96e-14}$	$\mathbf{7.95e+01 \pm 2.56e-14}$	=
f_2	$\mathbf{-2.10e+02 \pm 3.18e-14}$	$\mathbf{-2.10e+02 \pm 3.28e-14}$	=
f_3	$\mathbf{-4.56e+02 \pm 9.98e+00}$	$-4.54e+02 \pm 3.44e+00$	+
f_4	$\mathbf{-4.53e+02 \pm 8.17e+00}$	$-4.51e+02 \pm 4.06e+00$	+
f_5	$\mathbf{-9.21e+00 \pm 8.63e-13}$	$5.63e+01 \pm 1.78e+02$	+
f_6	$3.59e+01 \pm 3.02e-06$	$\mathbf{3.59e+01 \pm 9.31e-07}$	=
f_7	$\mathbf{1.60e+02 \pm 2.50e+01}$	$2.10e+02 \pm 6.39e+01$	+
f_8	$\mathbf{1.50e+02 \pm 8.20e+00}$	$1.53e+02 \pm 1.69e+01$	=
f_9	$1.26e+02 \pm 7.77e+00$	$\mathbf{1.25e+02 \pm 1.53e+00}$	-
f_{10}	$\mathbf{1.00e+03 \pm 3.53e+02}$	$1.95e+05 \pm 1.40e+06$	=
f_{11}	$4.20e+02 \pm 7.64e+01$	$\mathbf{3.80e+02 \pm 6.30e+01}$	-
f_{12}	$\mathbf{-6.16e+02 \pm 6.25e+00}$	$-6.11e+02 \pm 8.98e+00$	+
f_{13}	$4.37e+01 \pm 1.25e+01$	$\mathbf{4.19e+01 \pm 1.28e+01}$	=
f_{14}	$-5.23e+01 \pm 5.44e-05$	$\mathbf{-5.23e+01 \pm 7.18e-05}$	-
f_{15}	$\mathbf{1.40e+03 \pm 1.71e+02}$	$2.06e+03 \pm 4.04e+02$	+
f_{16}	$\mathbf{8.63e+01 \pm 5.03e+00}$	$8.87e+01 \pm 5.44e+00$	+
f_{17}	$\mathbf{-9.70e+00 \pm 2.00e+00}$	$-5.52e+00 \pm 3.25e+00$	+
f_{18}	$\mathbf{1.13e+01 \pm 8.67e+00}$	$2.56e+01 \pm 1.47e+01$	+
f_{19}	$\mathbf{-9.62e+01 \pm 2.43e+00}$	$-9.33e+01 \pm 3.68e+00$	+
f_{20}	$-5.45e+02 \pm 1.98e-01$	$\mathbf{-5.46e+02 \pm 1.28e-01}$	-
f_{21}	$\mathbf{5.06e+01 \pm 1.47e+01}$	$5.28e+01 \pm 1.62e+01$	=
f_{22}	$\mathbf{-9.87e+02 \pm 1.12e+01}$	$-9.85e+02 \pm 1.31e+01$	=
f_{23}	$\mathbf{8.06e+00 \pm 5.71e-01}$	$8.10e+00 \pm 5.26e-01$	=
f_{24}	$\mathbf{6.06e+02 \pm 1.98e+02}$	$9.44e+02 \pm 2.79e+02$	+

Table E.2. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = S-3SOME) for S-3SOME against cGA, cDE and ISPO on CEC2010 in 1000 dimensions.

	S-3SOME	cDE		cGA		ISPO	
f1	$6.03e-04 \pm 4.71e-04$	$1.70e+11 \pm 1.67e+10$	+	$1.08e+11 \pm 1.60e+10$	+	$\mathbf{0.00e+00 \pm 0.00e+00}$	-
f2	$\mathbf{3.99e-02 \pm 9.13e-03}$	$1.49e+04 \pm 3.51e+02$	+	$2.02e+04 \pm 4.59e+02$	+	$1.38e+04 \pm 4.22e+02$	+
f3	$\mathbf{7.56e-03 \pm 6.83e-04}$	$2.08e+01 \pm 4.19e-02$	+	$2.12e+01 \pm 4.14e-02$	+	$1.99e+01 \pm 1.38e-02$	+
f4	$2.08e+13 \pm 6.90e+12$	$9.51e+13 \pm 4.02e+13$	+	$2.02e+14 \pm 6.92e+13$	+	$\mathbf{8.49e+12 \pm 2.60e+12}$	-
f5	$4.27e+08 \pm 1.16e+08$	$3.79e+08 \pm 7.46e+07$	=	$\mathbf{2.75e+08 \pm 6.82e+07}$	-	$8.84e+08 \pm 1.48e+08$	+
f6	$1.54e+07 \pm 5.39e+06$	$1.76e+07 \pm 2.17e+06$	=	$\mathbf{1.30e+07 \pm 3.27e+06}$	=	$1.98e+07 \pm 4.88e+04$	+
f7	$\mathbf{1.07e+10 \pm 2.83e+09}$	$3.99e+10 \pm 1.23e+10$	+	$6.69e+10 \pm 1.70e+10$	+	$3.85e+10 \pm 1.81e+10$	+
f8	$1.78e+09 \pm 2.68e+09$	$4.56e+11 \pm 6.70e+11$	+	$5.82e+13 \pm 5.83e+13$	+	$\mathbf{1.33e+09 \pm 2.12e+09}$	-
f9	$3.54e+08 \pm 7.89e+07$	$7.44e+10 \pm 7.07e+09$	+	$1.10e+11 \pm 1.54e+10$	+	$\mathbf{8.55e+07 \pm 9.80e+06}$	-
f10	$\mathbf{5.12e+03 \pm 2.63e+02}$	$1.80e+04 \pm 5.19e+02$	+	$2.02e+04 \pm 4.75e+02$	+	$1.49e+04 \pm 4.90e+02$	+
f11	$\mathbf{1.97e+02 \pm 4.27e+00}$	$2.31e+02 \pm 4.61e-01$	+	$2.31e+02 \pm 3.94e-01$	+	$2.18e+02 \pm 2.25e-01$	+
f12	$\mathbf{8.74e+04 \pm 2.12e+04}$	$6.18e+06 \pm 4.44e+05$	+	$8.88e+06 \pm 6.73e+05$	+	$2.21e+05 \pm 2.84e+04$	+
f13	$5.61e+05 \pm 7.34e+05$	$6.88e+11 \pm 7.25e+10$	+	$1.53e+12 \pm 1.93e+11$	+	$\mathbf{5.98e+03 \pm 4.10e+03}$	-
f14	$\mathbf{8.79e+07 \pm 2.66e+06}$	$6.76e+10 \pm 6.23e+09$	+	$1.05e+11 \pm 1.53e+10$	+	$1.97e+08 \pm 1.41e+07$	+
f15	$\mathbf{1.33e+04 \pm 2.40e+03}$	$1.91e+04 \pm 4.61e+02$	+	$1.99e+04 \pm 3.09e+02$	+	$1.58e+04 \pm 5.33e+02$	+
f16	$\mathbf{1.60e+02 \pm 1.14e+02}$	$4.22e+02 \pm 6.53e-01$	+	$4.23e+02 \pm 5.48e-01$	+	$3.97e+02 \pm 2.63e-01$	+
f17	$\mathbf{6.31e+04 \pm 8.74e+03}$	$1.05e+07 \pm 9.29e+05$	+	$1.46e+07 \pm 1.84e+06$	+	$4.84e+05 \pm 6.56e+04$	+
f18	$\mathbf{3.88e+03 \pm 4.66e+03}$	$2.47e+12 \pm 1.65e+11$	+	$4.65e+12 \pm 2.42e+11$	+	$1.79e+04 \pm 7.91e+03$	+
f19	$\mathbf{1.16e+06 \pm 8.91e+04}$	$1.10e+07 \pm 1.21e+06$	+	$3.54e+07 \pm 5.03e+06$	+	$5.57e+07 \pm 1.41e+07$	+
f20	$1.20e+03 \pm 1.93e+02$	$2.81e+12 \pm 1.65e+11$	+	$5.25e+12 \pm 2.10e+11$	+	$\mathbf{1.19e+03 \pm 3.07e+02}$	=

Table E.3. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = S-3SOME) for S-3SOME against G-CMA-ES, SADE and CCPSO2 on CEC2010 in 1000 dimensions.

	S-3SOME	G-CMA-ES		SADE		CCPSO2	
f1	$6.03e-04 \pm 4.71e-04$	$4.75e+05 \pm 4.02e+04$	+	$2.52e+07 \pm 4.74e+07$	+	$\mathbf{5.12e-05 \pm 7.94e-05}$	-
f2	$\mathbf{3.99e-02 \pm 9.13e-03}$	$1.02e+04 \pm 4.45e+02$	+	$5.70e+03 \pm 3.34e+02$	+	$1.33e+02 \pm 1.26e+02$	+
f3	$7.56e-03 \pm 6.83e-04$	$1.99e+01 \pm 1.08e-02$	+	$1.90e+01 \pm 2.17e-01$	+	$\mathbf{2.13e-06 \pm 3.52e-06}$	-
f4	$2.08e+13 \pm 6.90e+12$	$\mathbf{1.54e+11 \pm 2.13e+10}$	-	$3.44e+12 \pm 2.41e+12$	-	$3.80e+12 \pm 2.29e+12$	-
f5	$4.27e+08 \pm 1.16e+08$	$6.73e+08 \pm 9.20e+07$	+	$\mathbf{1.05e+08 \pm 1.78e+07}$	-	$4.08e+08 \pm 1.10e+08$	=
f6	$1.54e+07 \pm 5.39e+06$	$1.98e+07 \pm 6.46e+04$	+	$\mathbf{5.52e+05 \pm 8.13e+05}$	-	$1.67e+07 \pm 4.94e+06$	=
f7	$1.07e+10 \pm 2.83e+09$	$\mathbf{5.46e+06 \pm 3.41e+05}$	-	$2.35e+08 \pm 4.41e+08$	-	$1.33e+10 \pm 1.33e+10$	=
f8	$1.78e+09 \pm 2.68e+09$	$\mathbf{5.62e+06 \pm 1.88e+05}$	-	$8.28e+07 \pm 3.25e+07$	-	$7.40e+07 \pm 4.88e+07$	-
f9	$3.54e+08 \pm 7.89e+07$	$\mathbf{5.04e+05 \pm 4.39e+04}$	-	$3.90e+08 \pm 3.22e+08$	-	$8.51e+07 \pm 1.25e+07$	-
f10	$5.12e+03 \pm 2.63e+02$	$1.04e+04 \pm 3.99e+02$	+	$6.37e+03 \pm 2.69e+02$	+	$\mathbf{4.55e+03 \pm 2.93e+02}$	-
f11	$\mathbf{1.97e+02 \pm 4.27e+00}$	$2.18e+02 \pm 2.14e-01$	+	$2.05e+02 \pm 3.33e+00$	+	$2.01e+02 \pm 6.51e+00$	=
f12	$8.74e+04 \pm 2.12e+04$	$\mathbf{1.04e-12 \pm 8.83e-14}$	-	$4.85e+05 \pm 1.82e+05$	+	$1.38e+05 \pm 1.27e+05$	+
f13	$5.61e+05 \pm 7.34e+05$	$\mathbf{2.20e+02 \pm 3.37e+02}$	-	$1.32e+07 \pm 3.61e+07$	-	$1.36e+03 \pm 3.96e+02$	-
f14	$8.79e+07 \pm 2.66e+06$	$\mathbf{5.52e+05 \pm 5.43e+04}$	-	$6.30e+08 \pm 2.98e+08$	+	$2.93e+08 \pm 5.53e+07$	+
f15	$1.33e+04 \pm 2.40e+03$	$1.03e+04 \pm 5.24e+02$	-	$\mathbf{6.58e+03 \pm 4.22e+02}$	-	$9.27e+03 \pm 6.90e+02$	-
f16	$\mathbf{1.60e+02 \pm 1.14e+02}$	$3.97e+02 \pm 3.14e-01$	+	$3.83e+02 \pm 1.82e+00$	+	$3.94e+02 \pm 1.40e+00$	+
f17	$6.31e+04 \pm 8.74e+03$	$\mathbf{1.97e-11 \pm 8.02e-12}$	-	$9.23e+05 \pm 1.67e+05$	+	$2.68e+05 \pm 1.38e+05$	+
f18	$3.88e+03 \pm 4.66e+03$	$\mathbf{4.47e+02 \pm 3.88e+02}$	-	$1.98e+09 \pm 4.20e+09$	+	$8.02e+03 \pm 6.53e+03$	+
f19	$1.16e+06 \pm 8.91e+04$	$\mathbf{1.48e+04 \pm 3.07e+03}$	-	$2.69e+06 \pm 1.96e+05$	+	$4.38e+06 \pm 7.34e+06$	=
f20	$1.20e+03 \pm 1.93e+02$	$\mathbf{8.33e+02 \pm 6.48e+01}$	-	$2.29e+09 \pm 2.93e+09$	+	$1.52e+03 \pm 1.36e+02$	+

Table E.4. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum test (reference = 3SOME) for 3SOME against 3SOME-Rosenbrock and 3SOME-Powell on BBOB2010 in 10 dimensions

	3SOME	3SOME-Powell		3SOME-Rosenbrock	
f_1	$\mathbf{7.95e+01 \pm 1.21e-14}$	$7.95e+01 \pm 8.94e-05$	+	$\mathbf{7.95e+01 \pm 0.00e+00}$	-
f_2	$\mathbf{-2.10e+02 \pm 1.63e-14}$	$-1.86e+02 \pm 4.48e+01$	+	$-2.10e+02 \pm 4.17e-13$	+
f_3	$-4.61e+02 \pm 1.18e+00$	$-4.61e+02 \pm 5.03e-01$	-	$\mathbf{-4.62e+02 \pm 2.54e-01}$	-
f_4	$-4.60e+02 \pm 1.39e+00$	$-4.61e+02 \pm 8.10e-01$	-	$\mathbf{-4.62e+02 \pm 7.35e-01}$	-
f_5	$5.33e+00 \pm 2.91e+01$	$-8.56e+00 \pm 2.49e-01$	-	$\mathbf{-9.21e+00 \pm 6.35e-09}$	-
f_6	$8.25e+01 \pm 2.83e+02$	$4.66e+01 \pm 1.05e+01$	-	$\mathbf{3.63e+01 \pm 6.31e-01}$	-
f_7	$1.05e+02 \pm 1.23e+01$	$\mathbf{1.02e+02 \pm 5.88e+00}$	=	$1.03e+02 \pm 7.60e+00$	=
f_8	$\mathbf{1.49e+02 \pm 1.86e-01}$	$1.52e+02 \pm 2.47e+00$	+	$1.51e+02 \pm 2.09e+00$	+
f_9	$\mathbf{1.25e+02 \pm 1.69e+00}$	$1.47e+02 \pm 4.28e+01$	+	$1.26e+02 \pm 1.10e+01$	+
f_{10}	$3.95e+03 \pm 2.63e+04$	$6.42e+03 \pm 5.41e+03$	+	$\mathbf{2.01e+02 \pm 2.13e+02}$	-
f_{11}	$1.57e+02 \pm 3.36e+01$	$\mathbf{1.01e+02 \pm 9.08e+00}$	-	$1.66e+02 \pm 3.20e+01$	=
f_{12}	$\mathbf{-6.12e+02 \pm 1.33e+01}$	$-2.40e+02 \pm 3.93e+02$	+	$-6.12e+02 \pm 1.50e+01$	=
f_{13}	$\mathbf{4.26e+01 \pm 1.28e+01}$	$4.62e+01 \pm 1.29e+01$	+	$4.28e+01 \pm 1.17e+01$	=
f_{14}	$-5.23e+01 \pm 3.05e-05$	$-5.23e+01 \pm 2.06e-03$	+	$\mathbf{-5.23e+01 \pm 6.52e-05}$	-
f_{15}	$1.10e+03 \pm 6.38e+01$	$\mathbf{1.06e+03 \pm 2.23e+01}$	-	$1.06e+03 \pm 2.51e+01$	-
f_{16}	$7.97e+01 \pm 4.63e+00$	$\mathbf{7.58e+01 \pm 2.03e+00}$	-	$7.67e+01 \pm 3.80e+00$	-
f_{17}	$-1.03e+01 \pm 6.57e+00$	$\mathbf{-1.51e+01 \pm 9.11e-01}$	-	$-2.32e+00 \pm 1.09e+01$	+
f_{18}	$5.80e+00 \pm 2.56e+01$	$\mathbf{-1.17e+01 \pm 3.09e+00}$	-	$3.96e+01 \pm 5.28e+01$	+
f_{19}	$-9.80e+01 \pm 2.98e+00$	$\mathbf{-9.93e+01 \pm 7.62e-01}$	-	$-9.44e+01 \pm 4.24e+00$	+
f_{20}	$-5.46e+02 \pm 2.59e-01$	$\mathbf{-5.46e+02 \pm 2.17e-01}$	-	$-5.46e+02 \pm 2.44e-01$	-
f_{21}	$5.36e+01 \pm 1.34e+01$	$\mathbf{4.46e+01 \pm 3.62e+00}$	-	$4.51e+01 \pm 4.16e+00$	-
f_{22}	$-9.88e+02 \pm 1.55e+01$	$\mathbf{-9.96e+02 \pm 4.73e+00}$	-	$-9.96e+02 \pm 7.80e+00$	-
f_{23}	$7.86e+00 \pm 4.95e-01$	$7.92e+00 \pm 2.46e-01$	=	$\mathbf{7.54e+00 \pm 2.98e-01}$	-
f_{24}	$1.92e+02 \pm 4.46e+01$	$\mathbf{1.56e+02 \pm 1.43e+01}$	-	$1.67e+02 \pm 2.06e+01$	-

Table E.5. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum test (reference = 3SOME) for 3SOME against 3SOME-Rosenbrock and 3SOME-Powell on BBOB2010 in 20 dimensions

	3SOME	3SOME-Powell		3SOME-Rosenbrock	
f_1	$\mathbf{7.95e+01 \pm 1.70e-14}$	$7.95e+01 \pm 8.12e-04$	+	$\mathbf{7.95e+01 \pm 1.41e-14}$	=
f_2	$\mathbf{-2.10e+02 \pm 1.99e-14}$	$-1.54e+02 \pm 9.96e+01$	-	$-2.10e+02 \pm 2.96e-13$	=
f_3	$-4.59e+02 \pm 1.86e+00$	$-4.60e+02 \pm 8.59e-01$	=	$\mathbf{-4.62e+02 \pm 7.91e-01}$	=
f_4	$-4.57e+02 \pm 2.53e+00$	$-4.58e+02 \pm 1.40e+00$	=	$\mathbf{-4.61e+02 \pm 1.02e+00}$	=
f_5	$2.05e+01 \pm 7.73e+01$	$-7.66e+00 \pm 4.55e-01$	-	$\mathbf{-9.21e+00 \pm 3.07e-10}$	-
f_6	$\mathbf{3.59e+01 \pm 5.12e-06}$	$6.33e+01 \pm 3.51e+01$	+	$3.67e+01 \pm 4.15e+00$	+
f_7	$1.16e+02 \pm 1.60e+01$	$1.15e+02 \pm 1.17e+01$	=	$\mathbf{1.15e+02 \pm 1.29e+01}$	=
f_8	$\mathbf{1.49e+02 \pm 5.28e-01}$	$1.91e+02 \pm 3.48e+01$	+	$1.51e+02 \pm 2.05e+00$	+
f_9	$1.25e+02 \pm 1.68e+00$	$1.53e+02 \pm 3.13e+01$	+	$\mathbf{1.25e+02 \pm 1.65e+00}$	+
f_{10}	$\mathbf{2.90e+02 \pm 2.34e+02}$	$1.39e+04 \pm 7.53e+03$	+	$3.99e+02 \pm 2.98e+02$	+
f_{11}	$2.55e+02 \pm 8.50e+01$	$\mathbf{1.47e+02 \pm 1.88e+01}$	-	$2.32e+02 \pm 4.90e+01$	-
f_{12}	$6.89e+06 \pm 4.84e+07$	$2.75e+02 \pm 6.83e+02$	+	$\mathbf{-6.15e+02 \pm 1.02e+01}$	=
f_{13}	$3.78e+01 \pm 1.01e+01$	$4.73e+01 \pm 1.16e+01$	+	$\mathbf{3.75e+01 \pm 1.07e+01}$	=
f_{14}	$\mathbf{-5.23e+01 \pm 8.13e-05}$	$-5.23e+01 \pm 3.95e-03$	=	$-5.23e+01 \pm 3.56e-04$	=
f_{15}	$1.27e+03 \pm 1.58e+02$	$\mathbf{1.21e+03 \pm 6.02e+01}$	=	$1.24e+03 \pm 7.42e+01$	=
f_{16}	$8.37e+01 \pm 5.89e+00$	$\mathbf{8.26e+01 \pm 3.33e+00}$	=	$8.31e+01 \pm 3.88e+00$	=
f_{17}	$-7.05e+00 \pm 5.64e+00$	$\mathbf{-1.19e+01 \pm 1.34e+00}$	-	$-2.99e+00 \pm 6.88e+00$	-
f_{18}	$2.08e+01 \pm 2.63e+01$	$\mathbf{9.27e-01 \pm 5.23e+00}$	-	$4.11e+01 \pm 3.07e+01$	+
f_{19}	$-9.60e+01 \pm 3.34e+00$	$\mathbf{-9.65e+01 \pm 9.30e-01}$	=	$-9.19e+01 \pm 5.08e+00$	=
f_{20}	$-5.46e+02 \pm 1.91e-01$	$\mathbf{-5.46e+02 \pm 1.58e-01}$	=	$-5.46e+02 \pm 1.71e-01$	=
f_{21}	$5.97e+01 \pm 1.80e+01$	$5.53e+01 \pm 1.35e+01$	=	$\mathbf{5.21e+01 \pm 1.10e+01}$	-
f_{22}	$-9.84e+02 \pm 1.50e+01$	$\mathbf{-9.86e+02 \pm 1.39e+01}$	=	$-9.85e+02 \pm 1.30e+01$	=
f_{23}	$\mathbf{7.94e+00 \pm 6.06e-01}$	$8.41e+00 \pm 3.63e-01$	+	$8.04e+00 \pm 4.72e-01$	+
f_{24}	$3.69e+02 \pm 1.16e+02$	$\mathbf{2.88e+02 \pm 3.96e+01}$	-	$3.22e+02 \pm 5.90e+01$	-

Table E.6. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum test (reference = 3SOME) for 3SOME against 3SOME-Rosenbrock and 3SOME-Powell on BBOB2010 in 40 dimensions

	3SOME	3SOME-Powell		3SOME-Rosenbrock	
f_1	$\mathbf{7.95e+01 \pm 2.56e-14}$	$7.95e+01 \pm 2.89e-03$	+	$\mathbf{7.95e+01 \pm 1.99e-14}$	=
f_2	$\mathbf{-2.10e+02 \pm 3.28e-14}$	$-8.97e+01 \pm 1.09e+02$	+	$-2.10e+02 \pm 2.07e-13$	+
f_3	$-4.54e+02 \pm 3.44e+00$	$-4.57e+02 \pm 1.51e+00$	-	$\mathbf{-4.59e+02 \pm 1.86e+00}$	-
f_4	$-4.51e+02 \pm 4.06e+00$	$-4.52e+02 \pm 2.64e+00$	-	$\mathbf{-4.57e+02 \pm 2.56e+00}$	-
f_5	$5.63e+01 \pm 1.78e+02$	$-5.43e+00 \pm 7.20e-01$	-	$\mathbf{-9.21e+00 \pm 6.81e-12}$	-
f_6	$\mathbf{3.59e+01 \pm 9.31e-07}$	$1.13e+02 \pm 1.04e+02$	+	$3.66e+01 \pm 1.07e+00$	+
f_7	$\mathbf{2.10e+02 \pm 6.39e+01}$	$2.30e+02 \pm 5.20e+01$	+	$2.26e+02 \pm 5.19e+01$	+
f_8	$\mathbf{1.53e+02 \pm 1.69e+01}$	$2.31e+02 \pm 3.64e+01$	+	$1.83e+02 \pm 3.56e+01$	+
f_9	$\mathbf{1.25e+02 \pm 1.53e+00}$	$1.77e+02 \pm 3.46e+01$	+	$1.27e+02 \pm 1.54e+00$	+
f_{10}	$1.95e+05 \pm 1.40e+06$	$2.62e+04 \pm 1.25e+04$	-	$\mathbf{7.73e+02 \pm 3.58e+02}$	-
f_{11}	$3.80e+02 \pm 6.30e+01$	$\mathbf{2.26e+02 \pm 3.43e+01}$	-	$3.78e+02 \pm 6.60e+01$	=
f_{12}	$\mathbf{-6.11e+02 \pm 8.98e+00}$	$5.07e+03 \pm 7.43e+03$	+	$-6.10e+02 \pm 8.73e+00$	=
f_{13}	$\mathbf{4.19e+01 \pm 1.28e+01}$	$5.63e+01 \pm 1.56e+01$	+	$4.32e+01 \pm 1.38e+01$	=
f_{14}	$-5.23e+01 \pm 7.18e-05$	$-5.23e+01 \pm 4.10e-03$	+	$\mathbf{-5.23e+01 \pm 2.29e-05}$	-
f_{15}	$2.06e+03 \pm 4.04e+02$	$\mathbf{1.69e+03 \pm 1.82e+02}$	-	$1.80e+03 \pm 1.59e+02$	-
f_{16}	$\mathbf{8.87e+01 \pm 5.44e+00}$	$9.17e+01 \pm 4.68e+00$	+	$9.12e+01 \pm 4.82e+00$	+
f_{17}	$-5.52e+00 \pm 3.25e+00$	$\mathbf{-8.89e+00 \pm 1.52e+00}$	-	$-5.94e+00 \pm 2.91e+00$	=
f_{18}	$2.56e+01 \pm 1.47e+01$	$\mathbf{1.34e+01 \pm 4.92e+00}$	-	$2.55e+01 \pm 1.33e+01$	=
f_{19}	$\mathbf{-9.33e+01 \pm 3.68e+00}$	$-9.32e+01 \pm 1.28e+00$	=	$-8.96e+01 \pm 4.60e+00$	+
f_{20}	$-5.46e+02 \pm 1.28e-01$	$\mathbf{-5.46e+02 \pm 1.08e-01}$	-	$-5.46e+02 \pm 1.19e-01$	-
f_{21}	$5.28e+01 \pm 1.62e+01$	$\mathbf{4.79e+01 \pm 9.75e+00}$	=	$4.94e+01 \pm 1.19e+01$	=
f_{22}	$-9.85e+02 \pm 1.31e+01$	$-9.83e+02 \pm 1.48e+01$	+	$\mathbf{-9.87e+02 \pm 1.07e+01}$	=
f_{23}	$\mathbf{8.10e+00 \pm 5.26e-01}$	$9.38e+00 \pm 4.72e-01$	+	$8.70e+00 \pm 6.40e-01$	+
f_{24}	$9.44e+02 \pm 2.79e+02$	$\mathbf{6.97e+02 \pm 1.18e+02}$	-	$8.06e+02 \pm 1.18e+02$	-

Table E.7. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum test (reference = 3SOME) for 3SOME against 3SOME-Rosenbrock and 3SOME-Powell on BBOB2010 in 100 dimensions

	3SOME	3SOME-Powell		3SOME-Rosenbrock	
f_1	$\mathbf{7.95e+01 \pm 3.29e-14}$	$7.95e+01 \pm 7.05e-03$	+	$7.95e+01 \pm 2.86e-14$	+
f_2	$\mathbf{-2.10e+02 \pm 5.69e-14}$	$4.80e+02 \pm 5.03e+02$	+	$-2.10e+02 \pm 1.75e-13$	+
f_3	$-4.39e+02 \pm 7.28e+00$	$-4.43e+02 \pm 4.04e+00$	-	$\mathbf{-4.45e+02 \pm 5.80e+00}$	-
f_4	$-4.27e+02 \pm 8.70e+00$	$-4.28e+02 \pm 5.07e+00$	=	$\mathbf{-4.36e+02 \pm 7.52e+00}$	-
f_5	$7.40e+00 \pm 1.65e+02$	$3.31e+00 \pm 1.71e+00$	-	$\mathbf{-9.21e+00 \pm 6.33e-12}$	-
f_6	$\mathbf{3.59e+01 \pm 8.86e-08}$	$4.43e+02 \pm 2.11e+02$	+	$4.31e+01 \pm 5.29e+00$	+
f_7	$\mathbf{5.97e+02 \pm 2.83e+02}$	$9.36e+02 \pm 2.65e+02$	+	$8.45e+02 \pm 2.19e+02$	+
f_8	$\mathbf{1.83e+02 \pm 3.31e+01}$	$3.19e+02 \pm 6.10e+01$	+	$2.31e+02 \pm 4.62e+01$	+
f_9	$\mathbf{1.76e+02 \pm 1.36e+01}$	$2.53e+02 \pm 4.53e+01$	+	$1.77e+02 \pm 1.76e+01$	=
f_{10}	$2.68e+03 \pm 6.96e+02$	$6.31e+04 \pm 1.78e+04$	+	$\mathbf{2.10e+03 \pm 5.99e+02}$	-
f_{11}	$3.83e+02 \pm 8.22e+01$	$\mathbf{3.45e+02 \pm 7.84e+01}$	-	$5.51e+02 \pm 1.14e+02$	+
f_{12}	$-6.09e+02 \pm 1.83e+01$	$6.00e+03 \pm 4.95e+03$	+	$\mathbf{-6.12e+02 \pm 1.64e+01}$	=
f_{13}	$3.35e+01 \pm 4.87e+00$	$5.79e+01 \pm 9.02e+00$	+	$\mathbf{3.32e+01 \pm 4.16e+00}$	=
f_{14}	$-5.23e+01 \pm 5.47e-05$	$-5.23e+01 \pm 4.90e-03$	+	$\mathbf{-5.23e+01 \pm 1.67e-05}$	-
f_{15}	$4.53e+03 \pm 5.89e+02$	$\mathbf{4.08e+03 \pm 5.10e+02}$	-	$4.20e+03 \pm 5.26e+02$	-
f_{16}	$\mathbf{9.51e+01 \pm 6.11e+00}$	$1.03e+02 \pm 4.79e+00$	+	$1.02e+02 \pm 5.54e+00$	+
f_{17}	$-2.63e-02 \pm 3.97e+00$	$\mathbf{-2.60e+00 \pm 2.97e+00}$	-	$-1.32e+00 \pm 3.44e+00$	-
f_{18}	$4.55e+01 \pm 1.54e+01$	$\mathbf{3.57e+01 \pm 1.04e+01}$	-	$4.02e+01 \pm 1.27e+01$	-
f_{19}	$\mathbf{-9.08e+01 \pm 3.39e+00}$	$-8.80e+01 \pm 2.19e+00$	+	$-8.90e+01 \pm 3.92e+00$	+
f_{20}	$-5.46e+02 \pm 9.61e-02$	$\mathbf{-5.46e+02 \pm 7.49e-02}$	-	$-5.46e+02 \pm 1.17e-01$	=
f_{21}	$5.19e+01 \pm 1.21e+01$	$5.17e+01 \pm 1.29e+01$	=	$\mathbf{5.02e+01 \pm 1.11e+01}$	=
f_{22}	$-9.82e+02 \pm 1.47e+01$	$-9.80e+02 \pm 1.54e+01$	+	$\mathbf{-9.84e+02 \pm 1.30e+01}$	=
f_{23}	$\mathbf{8.21e+00 \pm 4.93e-01}$	$1.02e+01 \pm 4.95e-01$	+	$9.13e+00 \pm 5.39e-01$	+
f_{24}	$2.79e+03 \pm 4.75e+02$	$\mathbf{2.65e+03 \pm 2.50e+02}$	-	$2.68e+03 \pm 3.15e+02$	-

Table E.8. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum test (reference = 3SOME) for 3SOME against 3SOME-Rosenbrock and 3SOME-Powell on CEC2010 in 1000 dimensions

	3SOME	3SOME-Powell		3SOME-Rosenbrock	
f_1	$8.81e-03 \pm 1.72e-02$	$7.12e+07 \pm 1.59e+07$	+	$\mathbf{0.00e+00 \pm 0.00e+00}$	-
f_2	$\mathbf{1.48e+01 \pm 2.05e+01}$	$3.97e+02 \pm 2.73e+01$	+	$1.16e+03 \pm 2.11e+02$	+
f_3	$\mathbf{3.36e-01 \pm 3.36e-01}$	$3.01e+00 \pm 1.32e-01$	+	$2.23e+00 \pm 2.29e-01$	+
f_4	$8.65e+12 \pm 2.50e+12$	$9.40e+12 \pm 2.99e+12$	=	$\mathbf{8.47e+12 \pm 2.56e+12}$	=
f_5	$6.89e+08 \pm 1.11e+08$	$\mathbf{6.18e+08 \pm 1.25e+08}$	-	$6.54e+08 \pm 1.19e+08$	=
f_6	$1.98e+07 \pm 8.83e+04$	$1.98e+07 \pm 1.06e+05$	=	$\mathbf{1.98e+07 \pm 1.31e+05}$	-
f_7	$\mathbf{1.50e+09 \pm 3.67e+08}$	$1.51e+09 \pm 3.91e+08$	=	$1.54e+09 \pm 4.38e+08$	=
f_8	$3.65e+08 \pm 1.47e+09$	$\mathbf{2.69e+08 \pm 1.04e+09}$	=	$2.94e+08 \pm 1.27e+09$	-
f_9	$3.76e+08 \pm 7.12e+07$	$4.67e+08 \pm 4.37e+07$	+	$\mathbf{5.42e+07 \pm 1.09e+08}$	-
f_{10}	$\mathbf{6.79e+03 \pm 3.75e+02}$	$7.15e+03 \pm 3.69e+02$	+	$8.19e+03 \pm 4.36e+02$	+
f_{11}	$\mathbf{1.99e+02 \pm 5.33e-01}$	$2.08e+02 \pm 6.47e-01$	+	$2.18e+02 \pm 3.59e-01$	+
f_{12}	$1.53e+05 \pm 6.80e+04$	$3.92e+05 \pm 2.43e+04$	+	$\mathbf{1.33e+04 \pm 5.65e+04}$	-
f_{13}	$1.54e+04 \pm 5.67e+03$	$4.37e+04 \pm 9.84e+03$	+	$\mathbf{1.23e+03 \pm 6.24e+02}$	-
f_{14}	$1.19e+08 \pm 2.71e+07$	$1.27e+09 \pm 8.65e+07$	+	$\mathbf{2.95e+07 \pm 9.98e+06}$	-
f_{15}	$\mathbf{1.38e+04 \pm 5.64e+02}$	$1.42e+04 \pm 5.31e+02$	+	$1.39e+04 \pm 5.20e+02$	=
f_{16}	$\mathbf{3.71e+02 \pm 7.75e+01}$	$4.18e+02 \pm 3.63e+00$	+	$3.97e+02 \pm 3.16e-01$	+
f_{17}	$2.85e+05 \pm 2.41e+05$	$9.27e+05 \pm 3.92e+04$	+	$\mathbf{1.51e+05 \pm 3.15e+05}$	-
f_{18}	$2.71e+04 \pm 1.43e+04$	$5.64e+05 \pm 1.45e+05$	+	$\mathbf{2.22e+03 \pm 8.48e+02}$	-
f_{19}	$1.47e+05 \pm 2.19e+04$	$1.87e+05 \pm 2.86e+04$	+	$\mathbf{9.20e+04 \pm 1.69e+04}$	-
f_{20}	$1.14e+03 \pm 1.40e+02$	$5.55e+05 \pm 1.64e+05$	+	$\mathbf{9.22e+02 \pm 4.21e+02}$	-

Table E.9. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = 3SOME) for 3SOME against its rotation invariant versions RIS-3SOME, μDE-3SOME and (1+1)-CMA-ES on BBOB2010 in 10 dimensions.

	3SOME	(1+1)-CMA-ES-3SOME		μDE-3SOME		RIS-3SOME	
f_1	$7.95e+01 \pm 1.21e-14$	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$7.95e+01 \pm 1.06e-14$	=	$7.95e+01 \pm 1.03e-14$	=
f_2	$-2.10e+02 \pm 1.63e-14$	$\mathbf{-2.10e+02 \pm 0.00e+00}$	-	$-2.10e+02 \pm 1.52e-14$	=	$-2.10e+02 \pm 1.35e-14$	=
f_3	$\mathbf{-4.61e+02 \pm 1.18e+00}$	$-4.56e+02 \pm 2.69e+00$	+	$-4.60e+02 \pm 9.81e-01$	+	$-4.54e+02 \pm 4.41e+00$	+
f_4	$\mathbf{-4.60e+02 \pm 1.39e+00}$	$-4.55e+02 \pm 3.12e+00$	+	$-4.60e+02 \pm 1.54e+00$	=	$-4.51e+02 \pm 6.80e+00$	+
f_5	$5.33e+00 \pm 2.91e+01$	$6.41e+00 \pm 3.04e+01$	=	$-4.99e+00 \pm 8.59e+00$	-	$\mathbf{-7.56e+00 \pm 9.86e+00}$	-
f_6	$8.25e+01 \pm 2.83e+02$	$1.50e+02 \pm 7.66e+02$	+	$3.70e+01 \pm 1.13e+01$	-	$\mathbf{3.59e+01 \pm 1.71e-03}$	=
f_7	$1.05e+02 \pm 1.23e+01$	$\mathbf{9.32e+01 \pm 4.00e-01}$	-	$1.01e+02 \pm 4.58e+00$	-	$1.03e+02 \pm 9.01e+00$	=
f_8	$1.49e+02 \pm 1.86e-01$	$\mathbf{1.49e+02 \pm 0.00e+00}$	-	$1.49e+02 \pm 1.51e-01$	=	$1.49e+02 \pm 1.49e-01$	=
f_9	$1.25e+02 \pm 1.69e+00$	$1.25e+02 \pm 1.50e+00$	-	$\mathbf{1.24e+02 \pm 9.47e-01}$	-	$1.26e+02 \pm 1.01e+01$	=
f_{10}	$3.95e+03 \pm 2.63e+04$	$2.26e+03 \pm 2.31e+04$	-	$4.03e+03 \pm 1.10e+04$	+	$\mathbf{2.60e+02 \pm 1.60e+02}$	=
f_{11}	$1.57e+02 \pm 3.36e+01$	$\mathbf{7.63e+01 \pm 0.00e+00}$	-	$1.29e+02 \pm 2.62e+01$	-	$1.36e+02 \pm 2.72e+01$	-
f_{12}	$-6.12e+02 \pm 1.33e+01$	$\mathbf{-6.21e+02 \pm 1.02e+00}$	-	$-6.00e+02 \pm 2.19e+01$	+	$-6.08e+02 \pm 1.63e+01$	=
f_{13}	$4.26e+01 \pm 1.28e+01$	$4.06e+01 \pm 1.08e+01$	=	$\mathbf{3.83e+01 \pm 8.89e+00}$	=	$4.09e+01 \pm 1.11e+01$	=
f_{14}	$-5.23e+01 \pm 3.05e-05$	$\mathbf{-5.23e+01 \pm 1.94e-11}$	-	$-5.23e+01 \pm 2.01e-05$	-	$-5.23e+01 \pm 2.40e-05$	=
f_{15}	$1.10e+03 \pm 6.38e+01$	$1.08e+03 \pm 4.71e+01$	=	$1.06e+03 \pm 2.48e+01$	-	$\mathbf{1.06e+03 \pm 2.92e+01}$	-
f_{16}	$7.97e+01 \pm 4.63e+00$	$7.83e+01 \pm 3.91e+00$	-	$\mathbf{7.59e+01 \pm 2.19e+00}$	-	$7.71e+01 \pm 3.53e+00$	-
f_{17}	$-1.03e+01 \pm 6.57e+00$	$-1.28e+01 \pm 2.39e+00$	-	$-1.32e+01 \pm 3.40e+00$	-	$\mathbf{-1.45e+01 \pm 1.50e+00}$	-
f_{18}	$5.80e+00 \pm 2.56e+01$	$-2.47e+00 \pm 9.57e+00$	=	$-4.37e-01 \pm 1.65e+01$	=	$\mathbf{-9.02e+00 \pm 4.48e+00}$	-
f_{19}	$-9.80e+01 \pm 2.98e+00$	$-9.94e+01 \pm 1.81e+00$	-	$\mathbf{-1.00e+02 \pm 1.49e+00}$	-	$-1.00e+02 \pm 1.38e+00$	-
f_{20}	$-5.46e+02 \pm 2.59e-01$	$-5.45e+02 \pm 3.69e-01$	+	$\mathbf{-5.46e+02 \pm 2.64e-01}$	-	$-5.45e+02 \pm 3.02e-01$	+
f_{21}	$5.36e+01 \pm 1.34e+01$	$4.82e+01 \pm 7.15e+00$	-	$\mathbf{4.46e+01 \pm 4.07e+00}$	-	$4.73e+01 \pm 6.24e+00$	-
f_{22}	$-9.88e+02 \pm 1.55e+01$	$-9.91e+02 \pm 1.29e+01$	-	$\mathbf{-9.98e+02 \pm 3.03e+00}$	-	$-9.94e+02 \pm 8.23e+00$	-
f_{23}	$7.86e+00 \pm 4.95e-01$	$7.86e+00 \pm 5.54e-01$	=	$\mathbf{7.60e+00 \pm 3.09e-01}$	-	$7.88e+00 \pm 5.66e-01$	=
f_{24}	$1.92e+02 \pm 4.46e+01$	$1.72e+02 \pm 2.72e+01$	-	$\mathbf{1.57e+02 \pm 1.57e+01}$	-	$1.61e+02 \pm 2.03e+01$	-

Table E.10. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = 3SOME) for 3SOME against its rotation invariant versions RIS-3SOME, μDE-3SOME and (1+1)-CMA-ES on BBOB2010 in 20 dimensions.

	3SOME	(1+1)-CMA-ES-3SOME		μDE-3SOME		RIS-3SOME	
f_1	$7.95e+01 \pm 1.70e-14$	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$7.95e+01 \pm 1.77e-14$	=	$7.95e+01 \pm 1.68e-14$	=
f_2	$-2.10e+02 \pm 1.99e-14$	$\mathbf{-2.10e+02 \pm 1.07e-14}$	-	$-2.10e+02 \pm 2.33e-14$	=	$-2.10e+02 \pm 2.07e-14$	=
f_3	$\mathbf{-4.59e+02 \pm 1.86e+00}$	$-4.40e+02 \pm 7.82e+00$	+	$-4.56e+02 \pm 2.50e+00$	+	$-4.39e+02 \pm 7.33e+00$	+
f_4	$\mathbf{-4.57e+02 \pm 2.53e+00}$	$-4.37e+02 \pm 7.92e+00$	+	$-4.53e+02 \pm 3.25e+00$	+	$-4.37e+02 \pm 8.66e+00$	+
f_5	$2.05e+01 \pm 7.73e+01$	$4.88e+00 \pm 5.59e+01$	-	$9.27e+00 \pm 3.66e+01$	=	$\mathbf{-9.21e+00 \pm 2.07e-13}$	-
f_6	$3.59e+01 \pm 5.12e-06$	$\mathbf{3.59e+01 \pm 0.00e+00}$	-	$1.94e+02 \pm 1.12e+03$	+	$3.59e+01 \pm 6.08e-07$	=
f_7	$1.16e+02 \pm 1.60e+01$	$\mathbf{9.85e+01 \pm 4.74e+00}$	-	$1.11e+02 \pm 1.06e+01$	=	$1.16e+02 \pm 1.40e+01$	=
f_8	$1.49e+02 \pm 5.28e-01$	$\mathbf{1.49e+02 \pm 1.02e-04}$	-	$1.50e+02 \pm 7.77e+00$	=	$1.50e+02 \pm 6.82e-01$	+
f_9	$1.25e+02 \pm 1.68e+00$	$\mathbf{1.25e+02 \pm 1.59e+00}$	-	$1.26e+02 \pm 1.95e+00$	=	$1.27e+02 \pm 7.63e+00$	+
f_{10}	$2.90e+02 \pm 2.34e+02$	$\mathbf{-4.12e+01 \pm 8.58e+01}$	-	$2.04e+04 \pm 8.55e+04$	+	$3.12e+02 \pm 2.70e+02$	=
f_{11}	$2.55e+02 \pm 8.50e+01$	$\mathbf{9.35e+01 \pm 1.22e+02}$	-	$2.01e+02 \pm 4.94e+01$	-	$1.97e+02 \pm 3.28e+01$	-
f_{12}	$6.89e+06 \pm 4.84e+07$	$\mathbf{-6.20e+02 \pm 1.96e+00}$	-	$-5.95e+02 \pm 2.71e+01$	-	$-6.05e+02 \pm 2.32e+01$	-
f_{13}	$\mathbf{3.78e+01 \pm 1.01e+01}$	$3.88e+01 \pm 1.24e+01$	=	$5.07e+01 \pm 1.66e+01$	+	$4.35e+01 \pm 1.36e+01$	+
f_{14}	$-5.23e+01 \pm 8.13e-05$	$\mathbf{-5.23e+01 \pm 4.98e-08}$	-	$-5.23e+01 \pm 7.97e-05$	-	$-5.23e+01 \pm 9.11e-05$	=
f_{15}	$1.27e+03 \pm 1.58e+02$	$1.24e+03 \pm 9.68e+01$	=	$\mathbf{1.17e+03 \pm 6.42e+01}$	-	$1.17e+03 \pm 7.33e+01$	-
f_{16}	$8.37e+01 \pm 5.89e+00$	$8.28e+01 \pm 4.97e+00$	=	$8.12e+01 \pm 3.85e+00$	-	$\mathbf{8.02e+01 \pm 4.29e+00}$	-
f_{17}	$-7.05e+00 \pm 5.64e+00$	$-9.84e+00 \pm 2.66e+00$	-	$-1.15e+01 \pm 2.19e+00$	-	$\mathbf{-1.22e+01 \pm 1.90e+00}$	-
f_{18}	$2.08e+01 \pm 2.63e+01$	$8.62e+00 \pm 1.18e+01$	-	$6.08e+00 \pm 1.20e+01$	-	$\mathbf{2.35e+00 \pm 7.42e+00}$	-
f_{19}	$-9.60e+01 \pm 3.34e+00$	$-9.82e+01 \pm 2.14e+00$	-	$\mathbf{-9.93e+01 \pm 1.48e+00}$	-	$-9.89e+01 \pm 1.43e+00$	-
f_{20}	$-5.46e+02 \pm 1.91e-01$	$-5.45e+02 \pm 2.23e-01$	+	$\mathbf{-5.46e+02 \pm 2.07e-01}$	-	$-5.45e+02 \pm 2.27e-01$	+
f_{21}	$5.97e+01 \pm 1.80e+01$	$5.32e+01 \pm 1.36e+01$	-	$5.19e+01 \pm 1.19e+01$	-	$\mathbf{5.00e+01 \pm 1.04e+01}$	-
f_{22}	$-9.84e+02 \pm 1.50e+01$	$-9.87e+02 \pm 1.44e+01$	-	$-9.89e+02 \pm 1.15e+01$	=	$\mathbf{-9.90e+02 \pm 1.11e+01}$	-
f_{23}	$7.94e+00 \pm 6.06e-01$	$8.08e+00 \pm 5.71e-01$	+	$\mathbf{7.89e+00 \pm 4.42e-01}$	=	$7.95e+00 \pm 5.39e-01$	=
f_{24}	$3.69e+02 \pm 1.16e+02$	$3.16e+02 \pm 6.29e+01$	-	$2.72e+02 \pm 4.16e+01$	-	$\mathbf{2.69e+02 \pm 5.46e+01}$	-

Table E.11. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = 3SOME) for 3SOME against its rotation invariant versions RIS-3SOME, μDE-3SOME and (1+1)-CMA-ES on BBOB2010 in 40 dimensions.

	3SOME	(1+1)-CMA-ES-3SOME		μDE-3SOME		RIS-3SOME	
f_1	$7.95e+01 \pm 2.56e-14$	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$7.95e+01 \pm 2.18e-14$	=	$7.95e+01 \pm 2.62e-14$	=
f_2	$-2.10e+02 \pm 3.28e-14$	$\mathbf{-2.10e+02 \pm 1.45e-14}$	-	$-2.10e+02 \pm 3.13e-14$	=	$-2.10e+02 \pm 3.44e-14$	=
f_3	$\mathbf{-4.54e+02 \pm 3.44e+00}$	$-4.16e+02 \pm 1.21e+01$	+	$-4.39e+02 \pm 4.92e+00$	+	$-4.15e+02 \pm 1.05e+01$	+
f_4	$\mathbf{-4.51e+02 \pm 4.06e+00}$	$-4.08e+02 \pm 1.43e+01$	+	$-4.31e+02 \pm 6.85e+00$	+	$-4.05e+02 \pm 1.43e+01$	+
f_5	$5.63e+01 \pm 1.78e+02$	$1.23e+01 \pm 1.05e+02$	-	$7.67e+01 \pm 1.42e+02$	+	$\mathbf{-9.21e+00 \pm 8.58e-13}$	-
f_6	$3.59e+01 \pm 9.31e-07$	$\mathbf{3.59e+01 \pm 0.00e+00}$	-	$3.59e+01 \pm 4.52e-06$	=	$3.59e+01 \pm 8.18e-08$	=
f_7	$2.10e+02 \pm 6.39e+01$	$\mathbf{1.25e+02 \pm 1.06e+01}$	-	$1.63e+02 \pm 3.21e+01$	-	$1.76e+02 \pm 3.20e+01$	-
f_8	$1.53e+02 \pm 1.69e+01$	$1.53e+02 \pm 3.53e+00$	+	$1.52e+02 \pm 1.36e+01$	=	$\mathbf{1.49e+02 \pm 5.23e-01}$	=
f_9	$1.25e+02 \pm 1.53e+00$	$1.31e+02 \pm 3.32e+00$	+	$\mathbf{1.25e+02 \pm 1.07e+00}$	=	$1.25e+02 \pm 1.48e+00$	+
f_{10}	$1.95e+05 \pm 1.40e+06$	$\mathbf{9.23e+01 \pm 6.56e+01}$	-	$1.62e+05 \pm 4.48e+05$	-	$8.93e+02 \pm 2.94e+02$	=
f_{11}	$3.80e+02 \pm 6.30e+01$	$\mathbf{8.16e+01 \pm 5.16e+01}$	-	$3.13e+02 \pm 5.82e+01$	-	$3.24e+02 \pm 4.72e+01$	-
f_{12}	$-6.11e+02 \pm 8.98e+00$	$-6.11e+02 \pm 8.50e+00$	=	$\mathbf{-6.16e+02 \pm 6.47e+00}$	-	$-6.15e+02 \pm 6.46e+00$	-
f_{13}	$4.19e+01 \pm 1.28e+01$	$\mathbf{4.07e+01 \pm 1.35e+01}$	=	$4.39e+01 \pm 1.06e+01$	+	$4.20e+01 \pm 1.05e+01$	=
f_{14}	$-5.23e+01 \pm 7.18e-05$	$\mathbf{-5.23e+01 \pm 6.31e-07}$	-	$-5.22e+01 \pm 1.88e+00$	+	$-5.23e+01 \pm 5.67e-05$	=
f_{15}	$2.06e+03 \pm 4.04e+02$	$1.75e+03 \pm 2.12e+02$	-	$\mathbf{1.37e+03 \pm 1.21e+02}$	-	$1.45e+03 \pm 1.57e+02$	-
f_{16}	$8.87e+01 \pm 5.44e+00$	$8.95e+01 \pm 5.53e+00$	=	$8.72e+01 \pm 5.46e+00$	-	$\mathbf{8.45e+01 \pm 4.71e+00}$	-
f_{17}	$-5.52e+00 \pm 3.25e+00$	$-9.42e+00 \pm 1.41e+00$	-	$\mathbf{-1.07e+01 \pm 1.53e+00}$	-	$-1.05e+01 \pm 1.27e+00$	-
f_{18}	$2.56e+01 \pm 1.47e+01$	$1.14e+01 \pm 5.42e+00$	-	$7.91e+00 \pm 5.93e+00$	-	$\mathbf{7.21e+00 \pm 4.66e+00}$	-
f_{19}	$-9.33e+01 \pm 3.68e+00$	$-9.54e+01 \pm 2.38e+00$	-	$\mathbf{-9.80e+01 \pm 2.12e+00}$	-	$-9.68e+01 \pm 1.94e+00$	-
f_{20}	$\mathbf{-5.46e+02 \pm 1.28e-01}$	$-5.45e+02 \pm 1.61e-01$	+	$-5.46e+02 \pm 1.97e-01$	+	$-5.45e+02 \pm 1.59e-01$	+
f_{21}	$5.28e+01 \pm 1.62e+01$	$4.96e+01 \pm 1.22e+01$	-	$\mathbf{4.44e+01 \pm 6.26e+00}$	-	$4.54e+01 \pm 8.39e+00$	-
f_{22}	$-9.85e+02 \pm 1.31e+01$	$-9.88e+02 \pm 8.83e+00$	-	$-9.87e+02 \pm 7.35e+00$	=	$\mathbf{-9.90e+02 \pm 9.58e+00}$	-
f_{23}	$\mathbf{8.10e+00 \pm 5.26e-01}$	$8.42e+00 \pm 6.70e-01$	+	$8.10e+00 \pm 5.52e-01$	=	$8.16e+00 \pm 5.76e-01$	=
f_{24}	$9.44e+02 \pm 2.79e+02$	$6.49e+02 \pm 1.43e+02$	-	$\mathbf{5.17e+02 \pm 7.50e+01}$	-	$5.80e+02 \pm 1.18e+02$	-

Table E.12. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = 3SOME) for 3SOME against its rotation invariant versions RIS-3SOME, μDE-3SOME and (1+1)-CMA-ES on BBOB2010 in 100 dimensions.

	3SOME	(1+1)-CMA-ES-3SOME		μDE-3SOME		RIS-S3SOME	
f_1	$7.95e+01 \pm 3.29e-14$	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$7.95e+01 \pm 3.96e-14$	=	$7.95e+01 \pm 4.10e-14$	=
f_2	$-2.10e+02 \pm 5.69e-14$	$\mathbf{-2.10e+02 \pm 2.45e-14}$	-	$-2.10e+02 \pm 5.49e-14$	=	$-2.10e+02 \pm 5.72e-14$	=
f_3	$\mathbf{-4.39e+02 \pm 7.28e+00}$	$-3.34e+02 \pm 2.37e+01$	+	$-3.64e+02 \pm 1.64e+01$	+	$-3.34e+02 \pm 2.09e+01$	+
f_4	$\mathbf{-4.27e+02 \pm 8.70e+00}$	$-3.07e+02 \pm 2.66e+01$	+	$-3.36e+02 \pm 2.14e+01$	+	$-2.96e+02 \pm 2.48e+01$	+
f_5	$7.40e+00 \pm 1.65e+02$	$-8.18e+00 \pm 3.02e-01$	-	$2.43e+02 \pm 4.90e+02$	+	$\mathbf{-9.21e+00 \pm 4.28e-12}$	-
f_6	$3.59e+01 \pm 8.86e-08$	$3.59e+01 \pm 1.48e-03$	+	$1.33e+04 \pm 1.32e+05$	+	$\mathbf{3.59e+01 \pm 3.89e-08}$	=
f_7	$5.97e+02 \pm 2.83e+02$	$\mathbf{2.90e+02 \pm 7.06e+01}$	-	$3.90e+02 \pm 1.06e+02$	-	$3.96e+02 \pm 1.03e+02$	-
f_8	$\mathbf{1.83e+02 \pm 3.31e+01}$	$2.13e+02 \pm 1.99e+01$	+	$1.84e+02 \pm 4.08e+01$	=	$1.89e+02 \pm 4.24e+01$	=
f_9	$\mathbf{1.76e+02 \pm 1.36e+01}$	$1.89e+02 \pm 1.36e+01$	+	$1.78e+02 \pm 2.38e+01$	=	$1.78e+02 \pm 1.34e+01$	+
f_{10}	$2.68e+03 \pm 6.96e+02$	$\mathbf{1.59e+03 \pm 4.52e+02}$	-	$6.99e+04 \pm 5.83e+05$	+	$2.97e+03 \pm 6.44e+02$	+
f_{11}	$3.83e+02 \pm 8.22e+01$	$\mathbf{7.63e+01 \pm 5.83e-03}$	-	$6.82e+02 \pm 1.21e+02$	+	$7.26e+02 \pm 8.46e+01$	+
f_{12}	$-6.09e+02 \pm 1.83e+01$	$-6.12e+02 \pm 1.65e+01$	=	$-6.17e+02 \pm 6.31e+00$	-	$\mathbf{-6.17e+02 \pm 9.12e+00}$	-
f_{13}	$3.35e+01 \pm 4.87e+00$	$\mathbf{3.30e+01 \pm 4.63e+00}$	=	$3.64e+01 \pm 5.08e+00$	+	$3.61e+01 \pm 4.80e+00$	+
f_{14}	$-5.23e+01 \pm 5.47e-05$	$\mathbf{-5.23e+01 \pm 2.08e-06}$	-	$-5.23e+01 \pm 5.56e-05$	-	$-5.23e+01 \pm 5.89e-05$	=
f_{15}	$4.53e+03 \pm 5.89e+02$	$3.65e+03 \pm 4.54e+02$	-	$\mathbf{2.26e+03 \pm 2.81e+02}$	-	$2.49e+03 \pm 4.94e+02$	-
f_{16}	$9.51e+01 \pm 6.11e+00$	$9.90e+01 \pm 4.38e+00$	+	$9.28e+01 \pm 8.46e+00$	-	$\mathbf{8.94e+01 \pm 3.94e+00}$	-
f_{17}	$-2.63e-02 \pm 3.97e+00$	$-6.72e+00 \pm 1.91e+00$	-	$\mathbf{-8.67e+00 \pm 1.82e+00}$	-	$-7.28e+00 \pm 1.79e+00$	-
f_{18}	$4.55e+01 \pm 1.54e+01$	$2.34e+01 \pm 7.42e+00$	-	$\mathbf{1.44e+01 \pm 7.27e+00}$	-	$1.91e+01 \pm 6.52e+00$	-
f_{19}	$-9.08e+01 \pm 3.39e+00$	$-8.89e+01 \pm 3.68e+00$	+	$\mathbf{-9.39e+01 \pm 2.07e+00}$	-	$-9.27e+01 \pm 3.76e+00$	-
f_{20}	$\mathbf{-5.46e+02 \pm 9.61e-02}$	$-5.45e+02 \pm 1.04e-01$	+	$-5.45e+02 \pm 1.01e-01$	+	$-5.45e+02 \pm 9.12e-02$	+
f_{21}	$5.19e+01 \pm 1.21e+01$	$4.95e+01 \pm 9.21e+00$	-	$\mathbf{4.81e+01 \pm 6.53e+00}$	-	$4.94e+01 \pm 8.84e+00$	=
f_{22}	$-9.82e+02 \pm 1.47e+01$	$-9.84e+02 \pm 1.38e+01$	-	$\mathbf{-9.87e+02 \pm 1.04e+01}$	-	$-9.87e+02 \pm 1.06e+01$	=
f_{23}	$\mathbf{8.21e+00 \pm 4.93e-01}$	$8.75e+00 \pm 5.79e-01$	+	$8.30e+00 \pm 5.67e-01$	=	$8.24e+00 \pm 4.56e-01$	=
f_{24}	$2.79e+03 \pm 4.75e+02$	$1.86e+03 \pm 2.86e+02$	-	$\mathbf{1.30e+03 \pm 1.44e+02}$	-	$1.82e+03 \pm 3.01e+02$	-

E.2 Extended numerical results for RIS

Table E.13. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against its predecessor 3SOME and popular meta-heuristics on CEC2005 in 30 dimensions.

	RIS	3SOME		CLPSO		JADE	
f_1	$-4.50e+02 \pm 1.31e-13$	$\mathbf{-4.50e+02 \pm 7.59e-14}$	-	$-4.50e+02 \pm 6.12e-05$	+	$-3.51e+02 \pm 1.99e+02$	+
f_2	$-4.50e+02 \pm 1.23e-10$	$\mathbf{-4.50e+02 \pm 2.23e-12}$	-	$6.44e+03 \pm 9.22e+02$	+	$3.08e+02 \pm 7.32e+02$	+
f_3	$\mathbf{2.03e+05 \pm 1.25e+05}$	$2.80e+05 \pm 1.70e+05$	+	$2.55e+07 \pm 6.25e+06$	+	$3.71e+06 \pm 1.77e+06$	+
f_4	$1.88e+04 \pm 5.57e+03$	$2.27e+04 \pm 1.13e+04$	+	$1.43e+04 \pm 1.70e+03$	-	$\mathbf{2.27e+03 \pm 1.68e+03}$	-
f_5	$\mathbf{3.48e+03 \pm 7.70e+02}$	$1.08e+04 \pm 3.27e+03$	+	$7.14e+03 \pm 5.68e+02$	+	$3.91e+03 \pm 9.18e+02$	+
f_6	$6.91e+02 \pm 4.39e+02$	$\mathbf{5.43e+02 \pm 2.64e+02}$	-	$6.08e+02 \pm 4.97e+01$	-	$5.07e+06 \pm 1.53e+07$	+
f_7	$\mathbf{-1.80e+02 \pm 3.38e-03}$	$-1.80e+02 \pm 1.13e-02$	+	$1.41e+11 \pm 7.45e+10$	+	$1.69e+13 \pm 1.78e+13$	+
f_8	$\mathbf{-1.20e+02 \pm 4.05e-04}$	$-1.20e+02 \pm 7.40e-04$	+	$-1.19e+02 \pm 5.05e-02$	+	$-1.19e+02 \pm 5.75e-02$	+
f_9	$-1.18e+02 \pm 1.63e-05$	$\mathbf{-1.18e+02 \pm 7.73e-14}$	-	$-1.15e+02 \pm 8.97e-01$	+	$-1.17e+02 \pm 1.22e+00$	+
f_{10}	$2.14e+02 \pm 1.52e+01$	$2.87e+02 \pm 2.99e+01$	+	$2.73e+02 \pm 1.22e+01$	+	$\mathbf{2.02e+02 \pm 2.20e+01}$	-
f_{11}	$\mathbf{1.09e+02 \pm 1.90e+00}$	$1.22e+02 \pm 4.59e+00$	+	$1.19e+02 \pm 1.69e+00$	+	$1.16e+02 \pm 4.48e+00$	+
f_{12}	$\mathbf{1.90e+02 \pm 1.41e+03}$	$1.27e+03 \pm 3.33e+03$	+	$4.46e+04 \pm 9.39e+03$	+	$1.72e+04 \pm 1.54e+04$	+
f_{13}	$-1.21e+02 \pm 1.92e+00$	$-1.25e+02 \pm 1.19e+00$	-	$-1.20e+02 \pm 8.21e-01$	+	$\mathbf{-1.26e+02 \pm 9.64e-01}$	-
f_{14}	$-2.86e+02 \pm 2.79e-01$	$-2.86e+02 \pm 3.31e-01$	-	$\mathbf{-2.87e+02 \pm 1.65e-01}$	-	$-2.87e+02 \pm 2.02e-01$	-
f_{15}	$\mathbf{1.44e+03 \pm 3.89e-01}$	$1.44e+03 \pm 8.77e-01$	=	$1.46e+03 \pm 2.18e+00$	+	$1.45e+03 \pm 3.45e+00$	+
f_{16}	$\mathbf{1.55e+03 \pm 7.51e+00}$	$1.62e+03 \pm 2.78e+01$	+	$1.61e+03 \pm 4.94e+00$	+	$1.56e+03 \pm 6.50e+00$	+
f_{17}	$1.66e+03 \pm 1.54e+01$	$1.67e+03 \pm 1.88e+01$	+	$1.67e+03 \pm 6.15e+00$	+	$\mathbf{1.59e+03 \pm 7.53e+00}$	-
f_{18}	$9.10e+02 \pm 4.58e-10$	$\mathbf{9.10e+02 \pm 5.90e-12}$	-	$9.10e+02 \pm 3.54e-05$	+	$9.10e+02 \pm 2.62e-01$	+
f_{19}	$9.10e+02 \pm 4.52e-10$	$\mathbf{9.10e+02 \pm 5.17e-12}$	-	$9.10e+02 \pm 3.06e-05$	+	$9.10e+02 \pm 1.31e-01$	+
f_{20}	$9.10e+02 \pm 4.29e-10$	$\mathbf{9.10e+02 \pm 4.76e-12}$	-	$9.10e+02 \pm 3.09e-05$	+	$9.10e+02 \pm 1.40e-01$	+
f_{21}	$1.69e+03 \pm 4.33e+00$	$1.73e+03 \pm 1.27e+01$	+	$1.72e+03 \pm 3.69e+00$	+	$\mathbf{1.69e+03 \pm 4.32e+00}$	=
f_{22}	$2.42e+03 \pm 2.92e+01$	$2.66e+03 \pm 8.39e+01$	+	$2.55e+03 \pm 1.91e+01$	+	$\mathbf{2.29e+03 \pm 3.44e+01}$	-
f_{23}	$1.72e+03 \pm 6.16e+00$	$1.72e+03 \pm 9.82e+00$	=	$1.77e+03 \pm 5.94e+00$	+	$\mathbf{1.70e+03 \pm 4.48e+00}$	-
f_{24}	$1.68e+03 \pm 6.83e+00$	$1.72e+03 \pm 1.13e+01$	+	$1.70e+03 \pm 6.21e+00$	+	$\mathbf{1.66e+03 \pm 1.40e+01}$	-
f_{25}	$\mathbf{1.43e+03 \pm 3.74e+02}$	$1.66e+03 \pm 3.97e+02$	+	$1.89e+03 \pm 8.46e+01$	+	$1.86e+03 \pm 4.65e+01$	+

Table E.14. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against its predecessor 3SOME and popular meta-heuristics on BBOB2010 in 100 dimensions.

	RIS	3SOME		CLPSO		JADE	
f_1	$7.95e+01 \pm 1.59e-12$	$\mathbf{7.95e+01 \pm 3.29e-14}$	-	$7.95e+01 \pm 4.12e-10$	+	$8.77e+01 \pm 7.64e+00$	+
f_2	$-2.10e+02 \pm 7.48e-08$	$\mathbf{-2.10e+02 \pm 5.69e-14}$	-	$-2.10e+02 \pm 9.39e-07$	+	$5.73e+04 \pm 8.69e+04$	+
f_3	$-3.35e+02 \pm 2.15e+01$	$-4.39e+02 \pm 7.28e+00$	-	$\mathbf{-4.62e+02 \pm 4.45e-01}$	-	$-3.11e+02 \pm 4.95e+01$	+
f_4	$-2.85e+02 \pm 3.40e+01$	$-4.27e+02 \pm 8.70e+00$	-	$\mathbf{-4.55e+02 \pm 1.50e+00}$	-	$-1.43e+02 \pm 9.83e+01$	+
f_5	$\mathbf{-9.21e+00 \pm 1.83e-05}$	$7.40e+00 \pm 1.65e+02$	+	$2.58e+02 \pm 1.17e+01$	+	$1.24e+02 \pm 5.08e+01$	+
f_6	$3.59e+01 \pm 3.75e-07$	$\mathbf{3.59e+01 \pm 8.86e-08}$	-	$4.25e+02 \pm 2.65e+01$	+	$3.92e+02 \pm 1.18e+02$	+
f_7	$\mathbf{2.09e+02 \pm 1.93e+01}$	$5.97e+02 \pm 2.83e+02$	+	$2.25e+02 \pm 1.07e+01$	+	$3.08e+02 \pm 6.50e+01$	+
f_8	$2.30e+02 \pm 4.51e+01$	$\mathbf{1.83e+02 \pm 3.31e+01}$	-	$2.82e+02 \pm 3.53e+01$	+	$7.24e+03 \pm 6.15e+03$	+
f_9	$1.77e+02 \pm 2.64e+01$	$\mathbf{1.76e+02 \pm 1.36e+01}$	-	$2.21e+02 \pm 5.45e-01$	+	$1.78e+03 \pm 1.33e+03$	+
f_{10}	$\mathbf{2.44e+03 \pm 5.46e+02}$	$2.68e+03 \pm 6.96e+02$	+	$5.32e+05 \pm 6.46e+04$	+	$1.94e+05 \pm 8.87e+04$	+
f_{11}	$9.47e+02 \pm 1.10e+02$	$3.83e+02 \pm 8.22e+01$	-	$3.18e+02 \pm 1.67e+01$	-	$\mathbf{2.11e+02 \pm 2.76e+01}$	-
f_{12}	$\mathbf{-6.18e+02 \pm 4.16e+00}$	$-6.09e+02 \pm 1.83e+01$	+	$-6.03e+02 \pm 5.15e+00$	+	$2.22e+07 \pm 2.13e+07$	+
f_{13}	$\mathbf{3.21e+01 \pm 1.68e+00}$	$3.35e+01 \pm 4.87e+00$	=	$5.08e+01 \pm 2.08e+00$	+	$7.69e+02 \pm 2.46e+02$	+
f_{14}	$\mathbf{-5.23e+01 \pm 4.89e-05}$	$-5.23e+01 \pm 5.47e-05$	+	$-5.23e+01 \pm 2.77e-03$	+	$-4.65e+01 \pm 3.89e+00$	+
f_{15}	$1.95e+03 \pm 1.18e+02$	$4.53e+03 \pm 5.89e+02$	+	$2.10e+03 \pm 3.63e+01$	+	$\mathbf{1.59e+03 \pm 8.67e+01}$	-
f_{16}	$\mathbf{8.25e+01 \pm 1.56e+00}$	$9.51e+01 \pm 6.11e+00$	+	$9.53e+01 \pm 2.04e+00$	+	$1.01e+02 \pm 3.46e+00$	+
f_{17}	$-8.97e+00 \pm 1.62e+00$	$-2.63e-02 \pm 3.97e+00$	+	$-1.21e+01 \pm 2.96e-01$	-	$\mathbf{-1.45e+01 \pm 5.96e-01}$	-
f_{18}	$1.56e+01 \pm 6.47e+00$	$4.55e+01 \pm 1.54e+01$	+	$1.12e+00 \pm 1.09e+00$	-	$\mathbf{-8.79e+00 \pm 2.11e+00}$	-
f_{19}	$-9.32e+01 \pm 2.16e+00$	$-9.08e+01 \pm 3.39e+00$	+	$-9.48e+01 \pm 2.29e-01$	-	$\mathbf{-9.50e+01 \pm 2.26e-01}$	-
f_{20}	$-5.45e+02 \pm 1.17e-01$	$\mathbf{-5.46e+02 \pm 9.61e-02}$	-	$-5.45e+02 \pm 6.24e-02$	+	$-5.12e+02 \pm 9.92e+01$	+
f_{21}	$4.32e+01 \pm 2.63e+00$	$5.19e+01 \pm 1.21e+01$	+	$\mathbf{4.17e+01 \pm 1.21e+00}$	-	$4.93e+01 \pm 6.25e+00$	+
f_{22}	$-9.96e+02 \pm 5.45e+00$	$-9.82e+02 \pm 1.47e+01$	+	$\mathbf{-9.98e+02 \pm 5.95e-01}$	-	$-9.94e+02 \pm 6.66e+00$	+
f_{23}	$\mathbf{7.52e+00 \pm 1.07e-01}$	$8.21e+00 \pm 4.93e-01$	+	$1.03e+01 \pm 2.74e-01$	+	$1.07e+01 \pm 3.78e-01$	+
f_{24}	$1.26e+03 \pm 1.41e+02$	$2.79e+03 \pm 4.75e+02$	+	$1.21e+03 \pm 4.50e+01$	-	$\mathbf{1.03e+03 \pm 4.44e+01}$	-

Table E.15. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference= RIS) for RIS against its predecessor 3SOME and popular meta-heuristics on CEC2008 in 1000 dimensions.

	RIS	3SOME		CLPSO		JADE	
f_1	$\mathbf{-4.50e+02 \pm 1.33e-09}$	$-4.50e+02 \pm 4.03e-08$	+	$-2.98e+02 \pm 3.50e+01$	+	$1.00e+06 \pm 3.23e+05$	+
f_2	$\mathbf{-4.50e+02 \pm 2.43e-02}$	$-4.50e+02 \pm 3.17e-02$	+	$-3.72e+02 \pm 6.49e-01$	+	$-3.21e+02 \pm 8.63e+00$	+
f_3	$1.51e+03 \pm 8.53e+01$	$\mathbf{1.38e+03 \pm 8.37e+01}$	-	$4.02e+03 \pm 2.64e+02$	+	$4.26e+11 \pm 2.13e+11$	+
f_4	$5.83e+03 \pm 3.67e+02$	$\mathbf{-3.30e+02 \pm 4.11e-04}$	-	$-7.28e+01 \pm 1.44e+01$	-	$4.45e+03 \pm 1.00e+03$	-
f_5	$\mathbf{-1.80e+02 \pm 1.89e-03}$	$-1.80e+02 \pm 8.46e-03$	+	$-1.78e+02 \pm 2.17e-01$	+	$8.45e+03 \pm 3.11e+03$	+
f_6	$\mathbf{-1.40e+02 \pm 5.02e-07}$	$-1.40e+02 \pm 4.49e-04$	+	$-1.40e+02 \pm 9.84e-05$	+	$-1.22e+02 \pm 6.01e-01$	+
f_7	$-1.35e+04 \pm 1.08e+02$	$\mathbf{-1.39e+04 \pm 6.56e+01}$	-	$-1.33e+04 \pm 4.19e+01$	+	$-1.19e+04 \pm 4.24e+02$	+

Table E.16. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against its predecessor 3SOME and popular meta-heuristics on CEC2010 in 1000 dimensions.

	RIS	3SOME		CLPSO		JADE	
f_1	$\mathbf{4.16e-06 \pm 5.82e-07}$	$1.88e-02 \pm 5.82e-02$	+	$6.15e+05 \pm 1.42e+05$	+	$1.40e+10 \pm 6.91e+09$	+
f_2	$5.80e+03 \pm 4.17e+02$	$\mathbf{1.97e+01 \pm 2.87e+01}$	-	$1.99e+02 \pm 1.27e+01$	-	$4.56e+03 \pm 1.04e+03$	-
f_3	$\mathbf{4.84e-06 \pm 5.21e-07}$	$3.93e-01 \pm 3.61e-01$	+	$9.04e-01 \pm 1.06e-01$	+	$1.76e+01 \pm 6.75e-01$	+
f_4	$2.13e+13 \pm 4.05e+12$	$8.57e+12 \pm 2.82e+12$	-	$1.22e+13 \pm 2.82e+12$	-	$\mathbf{2.62e+12 \pm 1.03e+12}$	-
f_5	$4.62e+08 \pm 1.11e+08$	$7.17e+08 \pm 1.22e+08$	+	$1.93e+08 \pm 2.20e+07$	-	$\mathbf{8.58e+07 \pm 1.77e+07}$	-
f_6	$1.95e+07 \pm 2.27e+06$	$1.98e+07 \pm 1.16e+05$	=	$\mathbf{9.51e+03 \pm 2.09e+04}$	-	$3.48e+06 \pm 1.40e+06$	-
f_7	$1.89e+10 \pm 4.23e+09$	$1.57e+09 \pm 4.02e+08$	-	$\mathbf{5.53e+08 \pm 1.30e+08}$	-	$3.37e+09 \pm 3.66e+09$	-
f_8	$2.39e+10 \pm 1.33e+10$	$4.80e+08 \pm 1.78e+09$	-	$\mathbf{7.53e+07 \pm 2.77e+07}$	-	$6.31e+13 \pm 1.80e+14$	+
f_9	$\mathbf{1.69e+08 \pm 6.69e+06}$	$4.01e+08 \pm 6.95e+07$	+	$9.54e+08 \pm 5.86e+07$	+	$1.67e+10 \pm 5.87e+09$	+
f_10	$7.22e+03 \pm 2.90e+02$	$6.75e+03 \pm 3.73e+02$	-	$\mathbf{6.36e+03 \pm 1.92e+02}$	-	$7.50e+03 \pm 1.07e+03$	+
f_{11}	$1.34e+02 \pm 3.80e+01$	$1.99e+02 \pm 7.14e-01$	+	$\mathbf{9.71e+01 \pm 7.27e+00}$	-	$1.94e+02 \pm 7.49e+00$	+
f_{12}	$\mathbf{1.26e+04 \pm 3.41e+03}$	$1.59e+05 \pm 7.73e+04$	+	$7.77e+05 \pm 3.78e+04$	+	$2.32e+06 \pm 4.55e+05$	+
f_{13}	$2.54e+05 \pm 4.03e+04$	$1.49e+04 \pm 5.82e+03$	-	$\mathbf{9.09e+03 \pm 2.13e+03}$	-	$8.02e+10 \pm 4.76e+10$	+
f_{14}	$\mathbf{4.47e+07 \pm 1.44e+06}$	$1.22e+08 \pm 3.18e+07$	+	$1.45e+09 \pm 9.35e+07$	+	$1.31e+10 \pm 4.64e+09$	+
f_{15}	$\mathbf{7.26e+03 \pm 3.80e+02}$	$1.38e+04 \pm 5.34e+02$	+	$1.25e+04 \pm 3.21e+02$	+	$8.51e+03 \pm 1.03e+03$	+
f_{16}	$\mathbf{1.58e+02 \pm 3.61e+01}$	$3.71e+02 \pm 7.82e+01$	+	$2.42e+02 \pm 1.61e+01$	+	$3.83e+02 \pm 1.19e+01$	+
f_{17}	$\mathbf{2.13e+04 \pm 4.35e+03}$	$2.77e+05 \pm 2.15e+05$	+	$1.80e+06 \pm 6.51e+04$	+	$2.63e+06 \pm 7.56e+05$	+
f_{18}	$\mathbf{1.55e+03 \pm 1.27e+03}$	$2.68e+04 \pm 1.42e+04$	+	$7.08e+04 \pm 1.22e+04$	+	$4.42e+11 \pm 1.91e+11$	+
f_{19}	$2.47e+06 \pm 2.79e+05$	$\mathbf{1.44e+05 \pm 1.73e+04}$	-	$5.82e+06 \pm 2.81e+05$	+	$3.59e+06 \pm 7.17e+05$	+
f_{20}	$1.18e+03 \pm 1.64e+02$	$\mathbf{1.13e+03 \pm 1.26e+02}$	-	$3.65e+04 \pm 7.28e+03$	+	$5.48e+11 \pm 2.10e+11$	+

Table E.17. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against state-of-the-art algorithms on CEC2005 in 30 dimensions.

	RIS	CCPSO2		MA-CMA-Chains		MDE-pBX	
f_1	$-4.50e+02 \pm 1.31e-13$	$-4.50e+02 \pm 6.40e-13$	+	$-4.50e+02 \pm 8.36e-10$	+	$\mathbf{-4.50e+02 \pm 1.52e-13}$	-
f_2	$\mathbf{-4.50e+02 \pm 1.23e-10}$	$-4.38e+02 \pm 3.92e+01$	+	$-4.50e+02 \pm 1.47e-02$	+	$-4.50e+02 \pm 2.54e-03$	+
f_3	$2.03e+05 \pm 1.25e+05$	$1.56e+06 \pm 9.03e+05$	+	$\mathbf{1.45e+05 \pm 4.32e+05}$	-	$2.81e+05 \pm 1.99e+05$	+
f_4	$1.88e+04 \pm 5.57e+03$	$1.81e+04 \pm 3.74e+03$	=	$3.21e+02 \pm 5.06e+02$	-	$\mathbf{-1.29e+02 \pm 9.67e+02}$	-
f_5	$3.48e+03 \pm 7.70e+02$	$9.18e+03 \pm 1.59e+03$	+	$\mathbf{5.60e+02 \pm 5.63e+02}$	-	$2.74e+03 \pm 6.34e+02$	-
f_6	$6.91e+02 \pm 4.39e+02$	$4.63e+02 \pm 5.05e+01$	=	$\mathbf{4.00e+02 \pm 3.07e+01}$	-	$4.33e+02 \pm 4.81e+01$	-
f_7	$\mathbf{-1.80e+02 \pm 3.38e-03}$	$-1.80e+02 \pm 1.86e-02$	+	$-1.41e+02 \pm 1.16e+02$	+	$2.03e+06 \pm 1.88e+07$	+
f_8	$\mathbf{-1.20e+02 \pm 4.05e-04}$	$-1.19e+02 \pm 5.34e-02$	+	$-1.20e+02 \pm 9.08e-03$	+	$-1.19e+02 \pm 4.23e-01$	+
f_9	$-1.18e+02 \pm 1.63e-05$	$-1.18e+02 \pm 1.98e-01$	+	$\mathbf{-3.30e+02 \pm 7.35e-01}$	-	$-1.17e+02 \pm 1.14e+00$	+
f_{10}	$2.14e+02 \pm 1.52e+01$	$2.55e+02 \pm 1.98e+01$	+	$\mathbf{-2.96e+02 \pm 2.17e+01}$	-	$2.23e+02 \pm 2.44e+01$	+
f_{11}	$\mathbf{1.09e+02 \pm 1.90e+00}$	$1.18e+02 \pm 2.48e+00$	+	$1.14e+02 \pm 3.18e+00$	+	$1.11e+02 \pm 4.59e+00$	+
f_{12}	$\mathbf{1.90e+02 \pm 1.41e+03}$	$3.36e+03 \pm 4.99e+03$	+	$2.46e+02 \pm 1.07e+03$	=	$3.77e+03 \pm 3.87e+03$	+
f_{13}	$-1.21e+02 \pm 1.92e+00$	$\mathbf{-1.27e+02 \pm 1.90e-01}$	-	$-1.27e+02 \pm 1.96e+00$	-	$-1.19e+02 \pm 2.28e+00$	+
f_{14}	$-2.86e+02 \pm 2.79e-01$	$-2.87e+02 \pm 2.89e-01$	-	$-2.87e+02 \pm 3.18e-01$	-	$\mathbf{-2.87e+02 \pm 4.50e-01}$	-
f_{15}	$1.44e+03 \pm 3.89e-01$	$1.44e+03 \pm 1.25e-01$	-	$\mathbf{4.27e+02 \pm 2.92e+01}$	-	$1.46e+03 \pm 6.43e+00$	+
f_{16}	$1.55e+03 \pm 7.51e+00$	$1.58e+03 \pm 9.17e+00$	+	$\mathbf{2.61e+02 \pm 1.62e+02}$	-	$1.58e+03 \pm 1.11e+01$	+
f_{17}	$1.66e+03 \pm 1.54e+01$	$1.70e+03 \pm 1.29e+01$	+	$\mathbf{3.05e+02 \pm 1.68e+02}$	-	$1.62e+03 \pm 9.04e+00$	-
f_{18}	$9.10e+02 \pm 4.58e-10$	$\mathbf{9.10e+02 \pm 1.58e-12}$	-	$9.11e+02 \pm 3.88e+01$	+	$9.10e+02 \pm 8.31e-11$	-
f_{19}	$9.10e+02 \pm 4.52e-10$	$9.10e+02 \pm 1.41e-12$	-	$\mathbf{9.07e+02 \pm 4.09e+01}$	-	$9.10e+02 \pm 2.42e-10$	-
f_{20}	$9.10e+02 \pm 4.29e-10$	$9.10e+02 \pm 1.65e-12$	-	$\mathbf{9.09e+02 \pm 3.94e+01}$	-	$9.10e+02 \pm 3.41e-11$	-
f_{21}	$1.69e+03 \pm 4.33e+00$	$1.71e+03 \pm 5.62e+00$	+	$\mathbf{8.82e+02 \pm 1.13e+02}$	-	$1.70e+03 \pm 5.51e+00$	+
f_{22}	$2.42e+03 \pm 2.92e+01$	$2.49e+03 \pm 2.85e+01$	+	$\mathbf{1.25e+03 \pm 1.31e+01}$	-	$2.41e+03 \pm 4.97e+01$	=
f_{23}	$1.72e+03 \pm 6.16e+00$	$1.71e+03 \pm 4.86e+00$	-	$\mathbf{9.00e+02 \pm 6.27e+01}$	-	$1.70e+03 \pm 5.28e+00$	-
f_{24}	$1.68e+03 \pm 6.83e+00$	$1.70e+03 \pm 7.97e+00$	+	$\mathbf{4.60e+02 \pm 0.00e+00}$	-	$1.67e+03 \pm 1.55e+01$	-
f_{25}	$\mathbf{1.43e+03 \pm 3.74e+02}$	$1.85e+03 \pm 9.19e+01$	+	$1.89e+03 \pm 8.19e+00$	+	$1.83e+03 \pm 1.55e+02$	+

Table E.18. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against state-of-the-art algorithms onBBOB2010 in 100 dimensions.

	RIS	CCPSO2		MA-CMA-Chains		MDE-pBX	
f_1	$7.95e+01 \pm 1.59e-12$	$\mathbf{7.95e+01 \pm 1.81e-13}$	-	$7.95e+01 \pm 6.55e-10$	+	$7.95e+01 \pm 7.60e-05$	+
f_2	$-2.10e+02 \pm 7.48e-08$	$\mathbf{-2.10e+02 \pm 1.76e-12}$	-	$-2.10e+02 \pm 5.31e-06$	+	$-2.10e+02 \pm 6.06e-03$	+
f_3	$-3.35e+02 \pm 2.15e+01$	$\mathbf{-4.54e+02 \pm 8.40e+00}$	-	$-4.14e+02 \pm 9.03e+00$	-	$3.29e+01 \pm 7.94e+01$	+
f_4	$-2.85e+02 \pm 3.40e+01$	$\mathbf{-4.40e+02 \pm 1.31e+01}$	-	$-3.82e+02 \pm 1.14e+01$	-	$4.03e+02 \pm 1.31e+02$	+
f_5	$-9.21e+00 \pm 1.83e-05$	$-9.21e+00 \pm 1.20e-03$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	-	$-3.09e-02 \pm 1.27e+01$	+
f_6	$\mathbf{3.59e+01 \pm 3.75e-07}$	$1.25e+02 \pm 4.02e+01$	+	$3.59e+01 \pm 1.60e-03$	+	$8.03e+01 \pm 3.32e+01$	+
f_7	$2.09e+02 \pm 1.93e+01$	$4.38e+02 \pm 4.90e+01$	+	$\mathbf{1.33e+02 \pm 7.77e+00}$	-	$3.70e+02 \pm 7.43e+01$	+
f_8	$2.30e+02 \pm 4.51e+01$	$2.70e+02 \pm 3.42e+01$	+	$\mathbf{1.59e+02 \pm 1.45e+01}$	-	$3.40e+02 \pm 6.77e+01$	+
f_9	$\mathbf{1.77e+02 \pm 2.64e+01}$	$2.30e+02 \pm 2.78e+01$	+	$1.80e+02 \pm 8.49e+00$	+	$2.52e+02 \pm 3.75e+01$	+
f_{10}	$2.44e+03 \pm 5.46e+02$	$2.61e+04 \pm 6.64e+03$	+	$\mathbf{2.40e+03 \pm 1.44e+03}$	=	$1.64e+04 \pm 7.99e+03$	+
f_{11}	$9.47e+02 \pm 1.10e+02$	$6.21e+02 \pm 1.95e+02$	-	$1.85e+02 \pm 1.85e+01$	-	$\mathbf{9.16e+01 \pm 7.45e+00}$	-
f_{12}	$-6.18e+02 \pm 4.16e+00$	$-6.13e+02 \pm 1.20e+01$	+	$\mathbf{-6.21e+02 \pm 6.95e-01}$	-	$-5.99e+02 \pm 7.07e+01$	+
f_{13}	$3.21e+01 \pm 1.68e+00$	$3.31e+01 \pm 4.20e+00$	=	$\mathbf{3.11e+01 \pm 1.76e+00}$	-	$3.47e+01 \pm 6.70e+00$	=
f_{14}	$-5.23e+01 \pm 4.89e-05$	$-5.23e+01 \pm 2.34e-04$	+	$\mathbf{-5.23e+01 \pm 6.00e-05}$	-	$-5.23e+01 \pm 2.55e-03$	+
f_{15}	$1.95e+03 \pm 1.18e+02$	$2.33e+03 \pm 2.32e+02$	+	$\mathbf{1.41e+03 \pm 1.88e+02}$	-	$1.66e+03 \pm 1.10e+02$	-
f_{16}	$\mathbf{8.25e+01 \pm 1.56e+00}$	$9.88e+01 \pm 4.27e+00$	+	$8.38e+01 \pm 1.79e+00$	+	$8.85e+01 \pm 4.46e+00$	+
f_{17}	$-8.97e+00 \pm 1.62e+00$	$-8.29e+00 \pm 1.62e+00$	+	$\mathbf{-1.69e+01 \pm 4.95e-02}$	-	$-1.35e+01 \pm 4.83e-01$	-
f_{18}	$1.56e+01 \pm 6.47e+00$	$1.61e+01 \pm 6.61e+00$	=	$\mathbf{-1.64e+01 \pm 1.89e-01}$	-	$-4.84e+00 \pm 1.68e+00$	-
f_{19}	$-9.32e+01 \pm 2.16e+00$	$-9.46e+01 \pm 1.29e+00$	-	$\mathbf{-1.00e+02 \pm 1.17e+00}$	-	$-1.00e+02 \pm 7.13e-01$	-
f_{20}	$-5.45e+02 \pm 1.17e-01$	$\mathbf{-5.46e+02 \pm 6.73e-02}$	-	$-5.45e+02 \pm 9.48e-02$	+	$-5.44e+02 \pm 1.14e-01$	+
f_{21}	$\mathbf{4.32e+01 \pm 2.63e+00}$	$4.41e+01 \pm 3.38e+00$	+	$4.55e+01 \pm 7.35e+00$	+	$4.49e+01 \pm 5.88e+00$	+
f_{22}	$\mathbf{-9.96e+02 \pm 5.45e+00}$	$-9.95e+02 \pm 5.83e+00$	+	$-9.92e+02 \pm 8.63e+00$	+	$-9.92e+02 \pm 9.11e+00$	+
f_{23}	$\mathbf{7.52e+00 \pm 1.07e-01}$	$9.39e+00 \pm 4.22e-01$	+	$7.66e+00 \pm 2.66e-01$	+	$9.34e+00 \pm 7.99e-01$	+
f_{24}	$1.26e+03 \pm 1.41e+02$	$1.19e+03 \pm 1.53e+02$	-	$\mathbf{4.15e+02 \pm 1.12e+02}$	-	$4.75e+02 \pm 4.72e+01$	-

Table E.19. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against state-of-the-art algorithms on CEC2008 in 1000 dimensions.

	RIS	CCPSO2		MA-SSW-Chains		MDE-pBX	
f_1	$-4.50e+02 \pm 1.33e-09$	$\mathbf{-4.50e+02 \pm 3.54e-12}$	-	$6.18e+04 \pm 6.12e+05$	+	$1.20e+05 \pm 4.41e+04$	+
f_2	$\mathbf{-4.50e+02 \pm 2.43e-02}$	$-4.09e+02 \pm 2.46e+01$	+	$-2.66e+02 \pm 1.81e+00$	+	$-3.33e+02 \pm 4.09e+00$	+
f_3	$\mathbf{1.51e+03 \pm 8.53e+01}$	$1.80e+03 \pm 1.12e+02$	+	$1.45e+11 \pm 1.01e+12$	+	$3.13e+10 \pm 1.65e+10$	+
f_4	$5.83e+03 \pm 3.67e+02$	$\mathbf{-2.00e+02 \pm 1.07e+02}$	-	$1.53e+04 \pm 1.02e+03$	+	$7.60e+03 \pm 2.55e+02$	+
f_5	$\mathbf{-1.80e+02 \pm 1.89e-03}$	$-1.80e+02 \pm 2.98e-03$	+	$-1.72e+02 \pm 1.09e+01$	+	$1.08e+03 \pm 4.60e+02$	+
f_6	$-1.40e+02 \pm 5.02e-07$	$\mathbf{-1.40e+02 \pm 9.34e-11}$	-	$-1.20e+02 \pm 3.19e-01$	+	$-1.21e+02 \pm 5.10e-02$	+
f_7	$-1.35e+04 \pm 1.08e+02$	$\mathbf{-1.44e+04 \pm 7.46e+01}$	-	$-9.97e+03 \pm 8.06e+02$	+	$-1.11e+04 \pm 1.63e+02$	+

Table E.20. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = RIS) for RIS against state-of-the-art algorithms on CEC2010 in 1000 dimensions.

	RIS	CCPSO2		MA-SSW-Chains		MDE-pBX	
f_1	$4.16e-06 \pm 5.82e-07$	$\mathbf{6.47e-14 \pm 1.41e-13}$	-	$2.45e+11 \pm 4.07e+10$	+	$1.05e+09 \pm 6.58e+08$	+
f_2	$5.80e+03 \pm 4.17e+02$	$\mathbf{1.36e+02 \pm 1.11e+02}$	-	$1.97e+04 \pm 1.74e+03$	+	$7.02e+03 \pm 2.38e+02$	+
f_3	$4.84e-06 \pm 5.21e-07$	$\mathbf{7.34e-11 \pm 1.05e-10}$	-	$2.02e+01 \pm 4.21e-01$	+	$1.93e+01 \pm 4.76e-02$	+
f_4	$2.13e+13 \pm 4.05e+12$	$\mathbf{2.14e+12 \pm 1.27e+12}$	-	$1.24e+15 \pm 5.69e+14$	+	$3.21e+12 \pm 9.76e+11$	-
f_5	$4.62e+08 \pm 1.11e+08$	$3.92e+08 \pm 7.98e+07$	-	$8.09e+08 \pm 6.58e+07$	+	$\mathbf{1.54e+08 \pm 2.77e+07}$	-
f_6	$1.95e+07 \pm 2.27e+06$	$1.71e+07 \pm 4.45e+06$	-	$2.03e+07 \pm 2.24e+05$	+	$\mathbf{3.65e+06 \pm 1.75e+06}$	-
f_7	$1.89e+10 \pm 4.23e+09$	$7.60e+09 \pm 9.72e+09$	-	$7.64e+11 \pm 5.09e+11$	+	$\mathbf{6.79e+06 \pm 1.01e+07}$	-
f_8	$2.39e+10 \pm 1.33e+10$	$\mathbf{5.46e+07 \pm 4.16e+07}$	-	$5.34e+16 \pm 3.07e+16$	+	$2.03e+08 \pm 1.63e+08$	-
f_9	$1.69e+08 \pm 6.69e+06$	$\mathbf{5.01e+07 \pm 7.68e+06}$	-	$2.81e+11 \pm 4.08e+10$	+	$1.68e+09 \pm 1.00e+09$	+
f_{10}	$7.22e+03 \pm 2.90e+02$	$\mathbf{4.57e+03 \pm 2.75e+02}$	-	$1.85e+04 \pm 1.76e+03$	+	$7.33e+03 \pm 2.55e+02$	+
f_{11}	$\mathbf{1.34e+02 \pm 3.80e+01}$	$2.00e+02 \pm 5.98e+00$	+	$2.20e+02 \pm 3.44e+00$	+	$2.06e+02 \pm 2.40e+00$	+
f_{12}	$\mathbf{1.26e+04 \pm 3.41e+03}$	$6.12e+04 \pm 8.14e+04$	+	$1.69e+07 \pm 4.07e+06$	+	$2.92e+05 \pm 6.60e+04$	+
f_{13}	$2.54e+05 \pm 4.03e+04$	$\mathbf{1.14e+03 \pm 5.42e+02}$	-	$1.27e+12 \pm 5.77e+11$	+	$2.88e+09 \pm 3.17e+09$	+
f_{14}	$\mathbf{4.47e+07 \pm 1.44e+06}$	$1.60e+08 \pm 3.35e+07$	+	$3.23e+11 \pm 3.62e+10$	+	$1.04e+09 \pm 1.97e+08$	+
f_{15}	$\mathbf{7.26e+03 \pm 3.80e+02}$	$9.31e+03 \pm 5.52e+02$	+	$1.86e+04 \pm 1.58e+03$	+	$7.44e+03 \pm 2.80e+02$	+
f_{16}	$\mathbf{1.58e+02 \pm 3.61e+01}$	$3.95e+02 \pm 1.45e+00$	+	$4.00e+02 \pm 7.55e+00$	+	$3.84e+02 \pm 1.22e+00$	+
f_{17}	$\mathbf{2.13e+04 \pm 4.35e+03}$	$1.41e+05 \pm 1.44e+05$	+	$4.64e+07 \pm 1.30e+07$	+	$4.35e+05 \pm 8.33e+04$	+
f_{18}	$\mathbf{1.55e+03 \pm 1.27e+03}$	$5.62e+03 \pm 4.13e+03$	+	$5.38e+12 \pm 7.78e+11$	+	$3.73e+10 \pm 1.95e+10$	+
f_{19}	$2.47e+06 \pm 2.79e+05$	$1.14e+06 \pm 1.22e+06$	-	$1.24e+08 \pm 3.39e+07$	+	$\mathbf{9.22e+05 \pm 1.06e+05}$	-
f_{20}	$\mathbf{1.18e+03 \pm 1.64e+02}$	$1.42e+03 \pm 1.19e+02$	+	$6.07e+12 \pm 8.88e+11$	+	$4.18e+10 \pm 2.02e+10$	+

E.3 Extended numerical results for VISPO

Table E.21. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = VISPO) for VISPO against population-based algorithms on CEC2005 in 30 dimensions

	VISPO	ISPO		CLPSO		JADE	
f_1	$-4.50e+02 \pm 6.31e-02$	$\mathbf{-4.50e+02 \pm 6.12e-14}$	-	$-4.50e+02 \pm 6.54e-05$	-	$-3.60e+02 \pm 1.92e+02$	+
f_2	$1.18e+03 \pm 2.05e+03$	$\mathbf{-4.25e+02 \pm 1.28e+01}$	-	$6.63e+03 \pm 8.47e+02$	+	$2.52e+02 \pm 6.76e+02$	=
f_3	$7.85e+06 \pm 7.90e+06$	$\mathbf{2.47e+06 \pm 1.13e+06}$	-	$2.62e+07 \pm 6.38e+06$	+	$3.64e+06 \pm 1.96e+06$	-
f_4	$9.63e+03 \pm 3.30e+03$	$2.79e+06 \pm 2.88e+06$	+	$1.46e+04 \pm 1.86e+03$	+	$\mathbf{2.03e+03 \pm 1.45e+03}$	-
f_5	$1.44e+04 \pm 4.79e+03$	$2.44e+04 \pm 6.53e+03$	+	$7.16e+03 \pm 5.92e+02$	-	$\mathbf{3.86e+03 \pm 8.52e+02}$	-
f_6	$1.29e+03 \pm 2.59e+03$	$1.35e+03 \pm 1.93e+03$	=	$\mathbf{6.01e+02 \pm 5.17e+01}$	-	$3.99e+06 \pm 1.47e+07$	+
f_7	$4.50e+10 \pm 1.85e+11$	$\mathbf{-1.80e+02 \pm 1.25e-02}$	-	$1.65e+11 \pm 9.18e+10$	+	$1.81e+13 \pm 2.16e+13$	+
f_8	$-1.19e+02 \pm 1.04e-01$	$\mathbf{-1.20e+02 \pm 6.52e-02}$	-	$-1.19e+02 \pm 5.59e-02$	+	$-1.19e+02 \pm 5.38e-02$	+
f_9	$-1.14e+02 \pm 6.34e+00$	$-9.12e+01 \pm 1.24e+01$	+	$\mathbf{-3.30e+02 \pm 3.03e-01}$	-	$-3.20e+02 \pm 3.76e+00$	-
f_{10}	$2.92e+02 \pm 2.76e+01$	$3.07e+02 \pm 3.09e+01$	+	$-1.46e+02 \pm 1.66e+01$	-	$\mathbf{-2.66e+02 \pm 1.80e+01}$	-
f_{11}	$1.24e+02 \pm 2.92e+00$	$1.29e+02 \pm 4.12e+00$	+	$1.19e+02 \pm 1.56e+00$	-	$\mathbf{1.16e+02 \pm 4.46e+00}$	-
f_{12}	$6.22e+04 \pm 6.29e+04$	$1.28e+05 \pm 1.38e+05$	+	$4.39e+04 \pm 9.90e+03$	=	$\mathbf{1.69e+04 \pm 9.96e+03}$	-
f_{13}	$-1.27e+02 \pm 4.85e-01$	$-1.27e+02 \pm 5.72e-01$	+	$\mathbf{-1.27e+02 \pm 6.29e-01}$	-	$-1.25e+02 \pm 7.15e+00$	=
f_{14}	$-2.87e+02 \pm 3.48e-01$	$-2.85e+02 \pm 2.14e-01$	+	$\mathbf{-2.87e+02 \pm 2.10e-01}$	=	$-2.87e+02 \pm 2.76e-01$	+
f_{15}	$1.44e+03 \pm 2.82e+00$	$1.45e+03 \pm 5.22e+00$	+	$\mathbf{2.61e+02 \pm 6.41e+01}$	-	$4.44e+02 \pm 5.92e+01$	-
f_{16}	$1.60e+03 \pm 1.80e+01$	$1.62e+03 \pm 3.60e+01$	+	$3.78e+02 \pm 3.08e+01$	-	$\mathbf{2.43e+02 \pm 6.41e+01}$	-
f_{17}	$1.59e+03 \pm 1.31e+01$	$1.74e+03 \pm 3.52e+01$	+	$4.35e+02 \pm 2.95e+01$	-	$\mathbf{2.83e+02 \pm 8.37e+01}$	-
f_{18}	$\mathbf{9.10e+02 \pm 1.14e-10}$	$9.10e+02 \pm 1.76e-10$	+	$9.56e+02 \pm 2.06e+01$	+	$9.41e+02 \pm 8.99e+00$	+
f_{19}	$\mathbf{9.10e+02 \pm 9.02e-11}$	$9.10e+02 \pm 9.76e-11$	+	$9.50e+02 \pm 2.46e+01$	+	$9.40e+02 \pm 1.61e+01$	+
f_{20}	$\mathbf{9.10e+02 \pm 1.52e-10}$	$9.10e+02 \pm 1.42e-10$	+	$9.53e+02 \pm 2.55e+01$	+	$9.37e+02 \pm 2.42e+01$	+
f_{21}	$1.73e+03 \pm 1.25e+01$	$1.75e+03 \pm 2.04e+01$	+	$\mathbf{8.62e+02 \pm 1.08e+01}$	-	$9.53e+02 \pm 1.65e+02$	-
f_{22}	$2.64e+03 \pm 8.82e+01$	$3.02e+03 \pm 1.79e+02$	+	$1.43e+03 \pm 1.63e+01$	-	$\mathbf{1.30e+03 \pm 1.87e+01}$	-
f_{23}	$1.70e+03 \pm 6.58e+00$	$1.75e+03 \pm 2.10e+01$	+	$\mathbf{8.94e+02 \pm 3.01e-01}$	-	$1.09e+03 \pm 1.72e+02$	-
f_{24}	$1.69e+03 \pm 1.37e+01$	$1.74e+03 \pm 1.31e+01$	+	$\mathbf{4.64e+02 \pm 5.88e+00}$	-	$5.31e+02 \pm 1.44e+02$	-
f_{25}	$\mathbf{1.80e+03 \pm 3.89e+02}$	$3.03e+04 \pm 9.29e+04$	+	$1.89e+03 \pm 6.79e+01$	+	$1.86e+03 \pm 5.21e+01$	+

Table E.22. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = VISPO) for VISPO against population-based algorithms on BBOB2010 in 100 dimensions

	VISPO	ISPO		CLPSO		JADE	
f_1	$7.95e+01 \pm 3.98e-04$	$\mathbf{7.95e+01 \pm 1.34e-13}$	-	$7.95e+01 \pm 4.12e-10$	-	$8.77e+01 \pm 7.64e+00$	+
f_2	$-2.10e+02 \pm 6.25e-13$	$\mathbf{-2.10e+02 \pm 2.30e-14}$	-	$-2.10e+02 \pm 9.39e-07$	+	$5.73e+04 \pm 8.69e+04$	+
f_3	$\mathbf{-4.62e+02 \pm 2.54e-01}$	$6.38e+01 \pm 1.12e+02$	+	$-4.62e+02 \pm 4.45e-01$	+	$-3.11e+02 \pm 4.95e+01$	+
f_4	$\mathbf{-4.61e+02 \pm 8.19e-01}$	$2.49e+02 \pm 1.60e+02$	+	$-4.55e+02 \pm 1.50e+00$	+	$-1.43e+02 \pm 9.83e+01$	+
f_5	$\mathbf{-9.21e+00 \pm 0.00e+00}$	$-9.21e+00 \pm 0.00e+00$	=	$2.58e+02 \pm 1.17e+01$	+	$1.24e+02 \pm 5.08e+01$	+
f_6	$\mathbf{1.40e+02 \pm 4.95e+01}$	$2.58e+02 \pm 1.02e+02$	+	$4.25e+02 \pm 2.65e+01$	+	$3.92e+02 \pm 1.18e+02$	+
f_7	$4.80e+02 \pm 8.21e+01$	$2.03e+03 \pm 5.20e+02$	+	$\mathbf{2.25e+02 \pm 1.07e+01}$	-	$3.08e+02 \pm 6.50e+01$	-
f_8	$\mathbf{2.02e+02 \pm 5.27e+01}$	$2.41e+02 \pm 5.11e+01$	+	$2.82e+02 \pm 3.53e+01$	+	$7.24e+03 \pm 6.15e+03$	+
f_9	$2.36e+02 \pm 3.63e+01$	$2.24e+02 \pm 2.30e+01$	-	$\mathbf{2.21e+02 \pm 5.45e-01}$	-	$1.78e+03 \pm 1.33e+03$	+
f_{10}	$3.66e+04 \pm 8.76e+03$	$\mathbf{3.13e+04 \pm 8.42e+03}$	-	$5.32e+05 \pm 6.46e+04$	+	$1.94e+05 \pm 8.87e+04$	+
f_{11}	$6.16e+02 \pm 4.08e+02$	$1.20e+03 \pm 1.34e+02$	+	$3.18e+02 \pm 1.67e+01$	-	$\mathbf{2.11e+02 \pm 2.76e+01}$	-
f_{12}	$\mathbf{-6.11e+02 \pm 1.60e+01}$	$-6.08e+02 \pm 2.07e+01$	=	$-6.03e+02 \pm 5.15e+00$	+	$2.22e+07 \pm 2.13e+07$	+
f_{13}	$\mathbf{3.29e+01 \pm 4.88e+00}$	$3.31e+01 \pm 3.79e+00$	=	$5.08e+01 \pm 2.08e+00$	+	$7.69e+02 \pm 2.46e+02$	+
f_{14}	$\mathbf{-5.23e+01 \pm 2.85e-04}$	$-5.23e+01 \pm 5.54e-04$	+	$-5.23e+01 \pm 2.77e-03$	+	$-4.65e+01 \pm 3.89e+00$	+
f_{15}	$2.43e+03 \pm 3.42e+02$	$8.24e+03 \pm 1.16e+03$	+	$2.10e+03 \pm 3.63e+01$	-	$\mathbf{1.59e+03 \pm 8.67e+01}$	-
f_{16}	$1.10e+02 \pm 7.29e+00$	$1.20e+02 \pm 7.73e+00$	+	$\mathbf{9.53e+01 \pm 2.04e+00}$	-	$1.01e+02 \pm 3.46e+00$	-
f_{17}	$-8.18e+00 \pm 1.46e+00$	$2.56e+01 \pm 2.46e+01$	+	$-1.21e+01 \pm 2.96e-01$	-	$\mathbf{-1.45e+01 \pm 5.96e-01}$	-
f_{18}	$1.51e+01 \pm 5.02e+00$	$1.72e+02 \pm 1.43e+02$	+	$1.12e+00 \pm 1.09e+00$	-	$\mathbf{-8.79e+00 \pm 2.11e+00}$	-
f_{19}	$-8.42e+01 \pm 1.26e+01$	$1.54e+01 \pm 2.22e+01$	+	$-9.48e+01 \pm 2.29e-01$	-	$\mathbf{-9.50e+01 \pm 2.26e-01}$	-
f_{20}	$\mathbf{-5.46e+02 \pm 7.07e-02}$	$-5.45e+02 \pm 8.56e-02$	+	$-5.45e+02 \pm 6.24e-02$	+	$-5.12e+02 \pm 9.92e+01$	+
f_{21}	$5.24e+01 \pm 1.38e+01$	$5.28e+01 \pm 1.40e+01$	=	$\mathbf{4.17e+01 \pm 1.21e+00}$	-	$4.93e+01 \pm 6.25e+00$	=
f_{22}	$-9.80e+02 \pm 1.58e+01$	$-9.84e+02 \pm 1.18e+01$	-	$\mathbf{-9.98e+02 \pm 5.95e-01}$	-	$-9.94e+02 \pm 6.66e+00$	-
f_{23}	$\mathbf{9.77e+00 \pm 6.17e-01}$	$1.06e+01 \pm 7.08e-01$	+	$1.03e+01 \pm 2.74e-01$	+	$1.07e+01 \pm 3.78e-01$	+
f_{24}	$1.42e+03 \pm 2.04e+02$	$3.71e+03 \pm 3.63e+02$	+	$1.21e+03 \pm 4.50e+01$	-	$\mathbf{1.03e+03 \pm 4.44e+01}$	-

Table E.23. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = VISPO) for VISPO against population-based algorithms on CEC2010 in 1000 dimensions

	VISPO	ISPO		CLPSO		JADE	
f_1	$2.80e-10 \pm 5.24e-11$	$\mathbf{7.26e-32 \pm 7.23e-31}$	-	$6.15e+05 \pm 1.42e+05$	+	$1.40e+10 \pm 6.91e+09$	+
f_2	$\mathbf{6.75e-02 \pm 2.45e-01}$	$1.39e+04 \pm 4.51e+02$	+	$1.99e+02 \pm 1.27e+01$	+	$4.56e+03 \pm 1.04e+03$	+
f_3	$\mathbf{2.98e-09 \pm 1.35e-10}$	$1.99e+01 \pm 1.19e-02$	+	$9.04e-01 \pm 1.06e-01$	+	$1.76e+01 \pm 6.75e-01$	+
f_4	$7.43e+12 \pm 2.61e+12$	$5.18e+12 \pm 1.72e+12$	-	$1.22e+13 \pm 2.82e+12$	+	$\mathbf{2.62e+12 \pm 1.03e+12}$	-
f_5	$7.95e+08 \pm 1.23e+08$	$8.50e+08 \pm 1.30e+08$	+	$1.93e+08 \pm 2.20e+07$	-	$\mathbf{8.58e+07 \pm 1.77e+07}$	-
f_6	$1.98e+07 \pm 6.21e+04$	$1.98e+07 \pm 6.80e+04$	=	$\mathbf{9.51e+03 \pm 2.09e+04}$	-	$3.48e+06 \pm 1.40e+06$	-
f_7	$2.03e+10 \pm 9.63e+09$	$2.53e+10 \pm 1.57e+10$	+	$\mathbf{5.53e+08 \pm 1.30e+08}$	-	$3.37e+09 \pm 3.66e+09$	-
f_8	$1.11e+09 \pm 2.28e+09$	$4.97e+08 \pm 6.14e+08$	=	$\mathbf{7.53e+07 \pm 2.77e+07}$	-	$6.31e+13 \pm 1.80e+14$	+
f_9	$6.51e+07 \pm 6.82e+06$	$\mathbf{5.06e+07 \pm 5.40e+06}$	-	$9.54e+08 \pm 5.86e+07$	+	$1.67e+10 \pm 5.87e+09$	+
f_{10}	$7.88e+03 \pm 3.42e+02$	$1.49e+04 \pm 5.59e+02$	+	$\mathbf{6.36e+03 \pm 1.92e+02}$	-	$7.50e+03 \pm 1.07e+03$	-
f_{11}	$1.99e+02 \pm 2.31e-01$	$2.18e+02 \pm 1.84e-01$	+	$\mathbf{9.71e+01 \pm 7.27e+00}$	-	$1.94e+02 \pm 7.49e+00$	-
f_{12}	$\mathbf{1.19e+05 \pm 1.93e+04}$	$1.57e+05 \pm 2.76e+04$	+	$7.77e+05 \pm 3.78e+04$	+	$2.32e+06 \pm 4.55e+05$	+
f_{13}	$4.03e+03 \pm 4.28e+03$	$\mathbf{2.06e+03 \pm 1.98e+03}$	=	$9.09e+03 \pm 2.13e+03$	+	$8.02e+10 \pm 4.76e+10$	+
f_{14}	$1.47e+08 \pm 1.06e+07$	$\mathbf{1.14e+08 \pm 8.85e+06}$	-	$1.45e+09 \pm 9.35e+07$	+	$1.31e+10 \pm 4.64e+09$	+
f_{15}	$1.54e+04 \pm 4.97e+02$	$1.55e+04 \pm 4.72e+02$	=	$1.25e+04 \pm 3.21e+02$	-	$\mathbf{8.51e+03 \pm 1.03e+03}$	-
f_{16}	$3.98e+02 \pm 3.21e-01$	$3.97e+02 \pm 2.62e-01$	-	$\mathbf{2.42e+02 \pm 1.61e+01}$	-	$3.83e+02 \pm 1.19e+01$	-
f_{17}	$\mathbf{2.45e+05 \pm 3.44e+04}$	$3.53e+05 \pm 4.34e+04$	+	$1.80e+06 \pm 6.51e+04$	+	$2.63e+06 \pm 7.56e+05$	+
f_{18}	$1.85e+04 \pm 1.16e+04$	$\mathbf{1.05e+04 \pm 5.02e+03}$	-	$7.08e+04 \pm 1.22e+04$	+	$4.42e+11 \pm 1.91e+11$	+
f_{19}	$6.21e+06 \pm 5.17e+05$	$3.82e+07 \pm 6.30e+06$	+	$5.82e+06 \pm 2.81e+05$	-	$\mathbf{3.59e+06 \pm 7.17e+05}$	-
f_{20}	$\mathbf{6.58e+02 \pm 1.75e+02}$	$7.61e+02 \pm 2.30e+02$	+	$3.65e+04 \pm 7.28e+03$	+	$5.48e+11 \pm 2.10e+11$	+

E.4 Extended numerical results for PMS

Table E.24. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against popular meta-heuristics on CEC2005 in 30 dimensions.

	PMS	3SOME		CLPSO		JADE	
f_1	$\mathbf{-4.50e+02 \pm 6.97e-14}$	$-4.50e+02 \pm 7.59e-14$	=	$-4.50e+02 \pm 6.12e-05$	+	$-3.51e+02 \pm 1.99e+02$	+
f_2	$-4.50e+02 \pm 3.79e-10$	$\mathbf{-4.50e+02 \pm 2.23e-12}$	=	$6.44e+03 \pm 9.22e+02$	+	$3.08e+02 \pm 7.32e+02$	+
f_3	$\mathbf{1.97e+05 \pm 1.50e+05}$	$2.80e+05 \pm 1.70e+05$	+	$2.55e+07 \pm 6.25e+06$	+	$3.71e+06 \pm 1.77e+06$	+
f_4	$2.16e+04 \pm 7.94e+03$	$2.27e+04 \pm 1.13e+04$	+	$1.43e+04 \pm 1.70e+03$	-	$\mathbf{2.27e+03 \pm 1.68e+03}$	-
f_5	$1.08e+04 \pm 6.01e+03$	$1.08e+04 \pm 3.27e+03$	=	$7.14e+03 \pm 5.68e+02$	-	$\mathbf{3.91e+03 \pm 9.18e+02}$	-
f_6	$\mathbf{5.28e+02 \pm 2.71e+02}$	$5.43e+02 \pm 2.64e+02$	=	$6.08e+02 \pm 4.97e+01$	+	$5.07e+06 \pm 1.53e+07$	+
f_7	$-1.80e+02 \pm 1.43e-02$	$\mathbf{-1.80e+02 \pm 1.13e-02}$	=	$1.41e+11 \pm 7.45e+10$	+	$1.69e+13 \pm 1.78e+13$	+
f_8	$-1.20e+02 \pm 1.88e-03$	$\mathbf{-1.20e+02 \pm 7.40e-04}$	=	$-1.19e+02 \pm 5.05e-02$	=	$-1.19e+02 \pm 5.75e-02$	=
f_9	$-1.18e+02 \pm 5.17e-01$	$\mathbf{-1.18e+02 \pm 7.73e-14}$	-	$-1.15e+02 \pm 8.97e-01$	+	$-1.17e+02 \pm 1.22e+00$	+
f_{10}	$2.76e+02 \pm 2.45e+01$	$2.87e+02 \pm 2.99e+01$	+	$2.73e+02 \pm 1.22e+01$	=	$\mathbf{2.02e+02 \pm 2.20e+01}$	-
f_{11}	$1.23e+02 \pm 5.69e+00$	$1.22e+02 \pm 4.59e+00$	=	$1.19e+02 \pm 1.69e+00$	-	$\mathbf{1.16e+02 \pm 4.48e+00}$	-
f_{12}	$\mathbf{1.02e+03 \pm 2.18e+03}$	$1.27e+03 \pm 3.33e+03$	=	$4.46e+04 \pm 9.39e+03$	+	$1.72e+04 \pm 1.54e+04$	+
f_{13}	$-1.19e+02 \pm 4.66e+00$	$-1.25e+02 \pm 1.19e+00$	-	$-1.20e+02 \pm 8.21e-01$	=	$\mathbf{-1.26e+02 \pm 9.64e-01}$	-
f_{14}	$-2.86e+02 \pm 5.60e-01$	$-2.86e+02 \pm 3.31e-01$	=	$\mathbf{-2.87e+02 \pm 1.65e-01}$	=	$-2.87e+02 \pm 2.02e-01$	=
f_{15}	$\mathbf{1.44e+03 \pm 8.47e-01}$	$1.44e+03 \pm 8.77e-01$	=	$1.46e+03 \pm 2.18e+00$	+	$1.45e+03 \pm 3.45e+00$	+
f_{16}	$1.60e+03 \pm 2.52e+01$	$1.62e+03 \pm 2.78e+01$	+	$1.61e+03 \pm 4.94e+00$	+	$\mathbf{1.56e+03 \pm 6.50e+00}$	-
f_{17}	$1.67e+03 \pm 1.90e+01$	$1.67e+03 \pm 1.88e+01$	=	$1.67e+03 \pm 6.15e+00$	=	$\mathbf{1.59e+03 \pm 7.53e+00}$	-
f_{18}	$\mathbf{9.10e+02 \pm 5.70e-12}$	$9.10e+02 \pm 5.90e-12$	=	$9.10e+02 \pm 3.54e-05$	=	$9.10e+02 \pm 2.62e-01$	=
f_{19}	$9.10e+02 \pm 5.63e-12$	$\mathbf{9.10e+02 \pm 5.17e-12}$	=	$9.10e+02 \pm 3.06e-05$	=	$9.10e+02 \pm 1.31e-01$	=
f_{20}	$9.10e+02 \pm 5.52e-12$	$\mathbf{9.10e+02 \pm 4.76e-12}$	=	$9.10e+02 \pm 3.09e-05$	=	$9.10e+02 \pm 1.40e-01$	=
f_{21}	$1.72e+03 \pm 1.51e+01$	$1.73e+03 \pm 1.27e+01$	+	$1.72e+03 \pm 3.69e+00$	=	$\mathbf{1.69e+03 \pm 4.32e+00}$	-
f_{22}	$2.62e+03 \pm 8.11e+01$	$2.66e+03 \pm 8.39e+01$	+	$2.55e+03 \pm 1.91e+01$	-	$\mathbf{2.29e+03 \pm 3.44e+01}$	-
f_{23}	$1.72e+03 \pm 1.07e+01$	$1.72e+03 \pm 9.82e+00$	=	$1.77e+03 \pm 5.94e+00$	+	$\mathbf{1.70e+03 \pm 4.48e+00}$	-
f_{24}	$1.72e+03 \pm 1.39e+01$	$1.72e+03 \pm 1.13e+01$	=	$1.70e+03 \pm 6.21e+00$	-	$\mathbf{1.66e+03 \pm 1.40e+01}$	-
f_{25}	$\mathbf{1.65e+03 \pm 2.55e+02}$	$1.66e+03 \pm 3.97e+02$	=	$1.89e+03 \pm 8.46e+01$	+	$1.86e+03 \pm 4.65e+01$	+

Table E.25. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against popular meta-heuristics on BBOB2010 in 100 dimensions.

	PMS	3SOME		CLPSO		JADE	
f_1	$7.95e+01 \pm 3.65e-14$	$\mathbf{7.95e+01 \pm 3.29e-14}$	=	$7.95e+01 \pm 4.12e-10$	=	$8.77e+01 \pm 7.64e+00$	+
f_2	$-2.10e+02 \pm 7.13e-14$	$\mathbf{-2.10e+02 \pm 5.69e-14}$	-	$-2.10e+02 \pm 9.39e-07$	=	$5.73e+04 \pm 8.69e+04$	+
f_3	$-3.67e+02 \pm 1.66e+01$	$-4.39e+02 \pm 7.28e+00$	-	$\mathbf{-4.62e+02 \pm 4.45e-01}$	-	$-3.11e+02 \pm 4.95e+01$	+
f_4	$-3.40e+02 \pm 2.05e+01$	$-4.27e+02 \pm 8.70e+00$	-	$\mathbf{-4.55e+02 \pm 1.50e+00}$	-	$-1.43e+02 \pm 9.83e+01$	+
f_5	$\mathbf{-9.21e+00 \pm 3.51e-12}$	$7.40e+00 \pm 1.65e+02$	+	$2.58e+02 \pm 1.17e+01$	+	$1.24e+02 \pm 5.08e+01$	+
f_6	$4.05e+01 \pm 4.61e+00$	$\mathbf{3.59e+01 \pm 8.86e-08}$	-	$4.25e+02 \pm 2.65e+01$	+	$3.92e+02 \pm 1.18e+02$	+
f_7	$3.70e+02 \pm 8.80e+01$	$5.97e+02 \pm 2.83e+02$	+	$\mathbf{2.25e+02 \pm 1.07e+01}$	-	$3.08e+02 \pm 6.50e+01$	-
f_8	$2.21e+02 \pm 5.96e+01$	$\mathbf{1.83e+02 \pm 3.31e+01}$	-	$2.82e+02 \pm 3.53e+01$	+	$7.24e+03 \pm 6.15e+03$	+
f_9	$\mathbf{1.72e+02 \pm 2.30e+01}$	$1.76e+02 \pm 1.36e+01$	=	$2.21e+02 \pm 5.45e-01$	+	$1.78e+03 \pm 1.33e+03$	+
f_{10}	$\mathbf{2.37e+03 \pm 6.64e+02}$	$2.68e+03 \pm 6.96e+02$	+	$5.32e+05 \pm 6.46e+04$	+	$1.94e+05 \pm 8.87e+04$	+
f_{11}	$6.51e+02 \pm 8.17e+01$	$3.83e+02 \pm 8.22e+01$	-	$3.18e+02 \pm 1.67e+01$	-	$\mathbf{2.11e+02 \pm 2.76e+01}$	-
f_{12}	$\mathbf{-6.11e+02 \pm 1.52e+01}$	$-6.09e+02 \pm 1.83e+01$	=	$-6.03e+02 \pm 5.15e+00$	+	$2.22e+07 \pm 2.13e+07$	+
f_{13}	$3.60e+01 \pm 6.01e+00$	$\mathbf{3.35e+01 \pm 4.87e+00}$	=	$5.08e+01 \pm 2.08e+00$	+	$7.69e+02 \pm 2.46e+02$	+
f_{14}	$\mathbf{-5.23e+01 \pm 1.80e-05}$	$-5.23e+01 \pm 5.47e-05$	=	$-5.23e+01 \pm 2.77e-03$	=	$-4.65e+01 \pm 3.89e+00$	+
f_{15}	$2.39e+03 \pm 6.32e+02$	$4.53e+03 \pm 5.89e+02$	+	$2.10e+03 \pm 3.63e+01$	-	$\mathbf{1.59e+03 \pm 8.67e+01}$	-
f_{16}	$\mathbf{9.23e+01 \pm 9.50e+00}$	$9.51e+01 \pm 6.11e+00$	=	$9.53e+01 \pm 2.04e+00$	=	$1.01e+02 \pm 3.46e+00$	=
f_{17}	$-1.96e+00 \pm 9.61e+00$	$-2.63e-02 \pm 3.97e+00$	+	$-1.21e+01 \pm 2.96e-01$	-	$\mathbf{-1.45e+01 \pm 5.96e-01}$	-
f_{18}	$4.89e+01 \pm 4.41e+01$	$4.55e+01 \pm 1.54e+01$	=	$1.12e+00 \pm 1.09e+00$	-	$\mathbf{-8.79e+00 \pm 2.11e+00}$	-
f_{19}	$-5.08e+01 \pm 4.81e+01$	$-9.08e+01 \pm 3.39e+00$	=	$-9.48e+01 \pm 2.29e-01$	-	$\mathbf{-9.50e+01 \pm 2.26e-01}$	-
f_{20}	$-5.45e+02 \pm 1.28e-01$	$\mathbf{-5.46e+02 \pm 9.61e-02}$	-	$-5.45e+02 \pm 6.24e-02$	+	$-5.12e+02 \pm 9.92e+01$	+
f_{21}	$5.05e+01 \pm 1.12e+01$	$5.19e+01 \pm 1.21e+01$	=	$\mathbf{4.17e+01 \pm 1.21e+00}$	-	$4.93e+01 \pm 6.25e+00$	=
f_{22}	$-9.83e+02 \pm 1.29e+01$	$-9.82e+02 \pm 1.47e+01$	=	$\mathbf{-9.98e+02 \pm 5.95e-01}$	-	$-9.94e+02 \pm 6.66e+00$	-
f_{23}	$8.70e+00 \pm 8.16e-01$	$\mathbf{8.21e+00 \pm 4.93e-01}$	=	$1.03e+01 \pm 2.74e-01$	+	$1.07e+01 \pm 3.78e-01$	+
f_{24}	$2.14e+03 \pm 6.37e+02$	$2.79e+03 \pm 4.75e+02$	+	$1.21e+03 \pm 4.50e+01$	-	$\mathbf{1.03e+03 \pm 4.44e+01}$	-

Table E.26. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against popular meta-heuristics on CEC2008 in 1000 dimensions.

	PMS	3SOME		CLPSO		JADE	
f_1	$\mathbf{-4.50e+02 \pm 4.20e-13}$	$-4.50e+02 \pm 5.54e-08$	=	$-2.98e+02 \pm 3.50e+01$	+	$1.00e+06 \pm 3.23e+05$	+
f_2	$-3.88e+02 \pm 5.17e+01$	$\mathbf{-4.50e+02 \pm 3.04e-02}$	-	$-3.72e+02 \pm 6.49e-01$	=	$-3.21e+02 \pm 8.63e+00$	+
f_3	$\mathbf{1.34e+03 \pm 5.37e+02}$	$1.38e+03 \pm 8.44e+01$	=	$4.02e+03 \pm 2.64e+02$	+	$4.26e+11 \pm 2.13e+11$	+
f_4	$\mathbf{-3.30e+02 \pm 1.33e-12}$	$-3.30e+02 \pm 3.35e-04$	+	$-7.28e+01 \pm 1.44e+01$	+	$4.45e+03 \pm 1.00e+03$	+
f_5	$-1.80e+02 \pm 1.50e-02$	$\mathbf{-1.80e+02 \pm 7.92e-03}$	=	$-1.78e+02 \pm 2.17e-01$	+	$8.45e+03 \pm 3.11e+03$	+
f_6	$-1.38e+02 \pm 6.36e+00$	$\mathbf{-1.40e+02 \pm 5.50e-04}$	-	$-1.40e+02 \pm 9.84e-05$	-	$-1.22e+02 \pm 6.01e-01$	+
f_7	$-1.37e+04 \pm 2.80e+02$	$\mathbf{-1.39e+04 \pm 6.56e+01}$	-	$-1.33e+04 \pm 4.19e+01$	+	$-1.19e+04 \pm 4.24e+02$	+

Table E.27. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against popular meta-heuristics on CEC2010 in 1000 dimensions.

	PMS	3SOME		CLPSO		JADE	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$1.88e-02 \pm 5.82e-02$	+	$6.15e+05 \pm 1.42e+05$	+	$1.40e+10 \pm 6.91e+09$	+
f_2	$\mathbf{1.48e-13 \pm 2.79e-13}$	$1.97e+01 \pm 2.87e+01$	+	$1.99e+02 \pm 1.27e+01$	+	$4.56e+03 \pm 1.04e+03$	+
f_3	$4.52e-01 \pm 2.76e+00$	$\mathbf{3.93e-01 \pm 3.61e-01}$	-	$9.04e-01 \pm 1.06e-01$	+	$1.76e+01 \pm 6.75e-01$	+
f_4	$\mathbf{5.27e+11 \pm 2.72e+11}$	$8.57e+12 \pm 2.82e+12$	+	$1.22e+13 \pm 2.82e+12$	+	$2.62e+12 \pm 1.03e+12$	+
f_5	$4.78e+08 \pm 1.40e+08$	$7.17e+08 \pm 1.22e+08$	+	$1.93e+08 \pm 2.20e+07$	-	$\mathbf{8.58e+07 \pm 1.77e+07}$	-
f_6	$1.92e+07 \pm 2.24e+06$	$1.98e+07 \pm 1.16e+05$	+	$\mathbf{9.51e+03 \pm 2.09e+04}$	-	$3.48e+06 \pm 1.40e+06$	-
f_7	$\mathbf{1.02e+08 \pm 2.57e+08}$	$1.57e+09 \pm 4.02e+08$	+	$5.53e+08 \pm 1.30e+08$	+	$3.37e+09 \pm 3.66e+09$	+
f_8	$1.17e+08 \pm 1.30e+08$	$4.80e+08 \pm 1.78e+09$	+	$\mathbf{7.53e+07 \pm 2.77e+07}$	=	$6.31e+13 \pm 1.80e+14$	+
f_9	$\mathbf{6.19e+06 \pm 2.80e+06}$	$4.01e+08 \pm 6.95e+07$	+	$9.54e+08 \pm 5.86e+07$	+	$1.67e+10 \pm 5.87e+09$	+
f_{10}	$\mathbf{5.25e+03 \pm 2.09e+03}$	$6.75e+03 \pm 3.73e+02$	+	$6.36e+03 \pm 1.92e+02$	=	$7.50e+03 \pm 1.07e+03$	+
f_{11}	$1.85e+02 \pm 3.03e+01$	$1.99e+02 \pm 7.14e-01$	+	$\mathbf{9.71e+01 \pm 7.27e+00}$	-	$1.94e+02 \pm 7.49e+00$	=
f_{12}	$\mathbf{1.06e+03 \pm 7.23e+02}$	$1.59e+05 \pm 7.73e+04$	+	$7.77e+05 \pm 3.78e+04$	+	$2.32e+06 \pm 4.55e+05$	+
f_{13}	$\mathbf{1.18e+03 \pm 6.33e+02}$	$1.49e+04 \pm 5.82e+03$	+	$9.09e+03 \pm 2.13e+03$	+	$8.02e+10 \pm 4.76e+10$	+
f_{14}	$\mathbf{1.44e+07 \pm 5.47e+06}$	$1.22e+08 \pm 3.18e+07$	+	$1.45e+09 \pm 9.35e+07$	+	$1.31e+10 \pm 4.64e+09$	+
f_{15}	$1.20e+04 \pm 3.87e+03$	$1.38e+04 \pm 5.34e+02$	=	$1.25e+04 \pm 3.21e+02$	+	$\mathbf{8.51e+03 \pm 1.03e+03}$	-
f_{16}	$3.27e+02 \pm 9.13e+01$	$3.71e+02 \pm 7.82e+01$	+	$\mathbf{2.42e+02 \pm 1.61e+01}$	-	$3.83e+02 \pm 1.19e+01$	=
f_{17}	$\mathbf{1.37e+03 \pm 8.23e+02}$	$2.77e+05 \pm 2.15e+05$	+	$1.80e+06 \pm 6.51e+04$	+	$2.63e+06 \pm 7.56e+05$	+
f_{18}	$\mathbf{2.41e+03 \pm 1.00e+03}$	$2.68e+04 \pm 1.42e+04$	+	$7.08e+04 \pm 1.22e+04$	+	$4.42e+11 \pm 1.91e+11$	+
f_{19}	$1.47e+05 \pm 4.93e+04$	$\mathbf{1.44e+05 \pm 1.73e+04}$	=	$5.82e+06 \pm 2.81e+05$	+	$3.59e+06 \pm 7.17e+05$	+
f_{20}	$\mathbf{9.57e+02 \pm 5.37e+02}$	$1.13e+03 \pm 1.26e+02$	=	$3.65e+04 \pm 7.28e+03$	+	$5.48e+11 \pm 2.10e+11$	+

Table E.28. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against state-of-the-art algorithms on CEC2005 in 30 dimensions.

	PMS	CCPSO2		MA-LSCh-CMA		MDE-pBX	
f_1	$-4.50e+02 \pm 6.97e-14$	$-4.50e+02 \pm 6.40e-13$	+	$-4.50e+02 \pm 8.36e-10$	+	$\mathbf{-4.50e+02 \pm 1.52e-13}$	-
f_2	$\mathbf{-4.50e+02 \pm 3.79e-10}$	$-4.38e+02 \pm 3.92e+01$	+	$-4.50e+02 \pm 1.47e-02$	+	$-4.50e+02 \pm 2.54e-03$	+
f_3	$1.97e+05 \pm 1.50e+05$	$1.56e+06 \pm 9.03e+05$	+	$\mathbf{1.45e+05 \pm 4.32e+05}$	-	$2.81e+05 \pm 1.99e+05$	+
f_4	$2.16e+04 \pm 7.94e+03$	$1.81e+04 \pm 3.74e+03$	-	$3.21e+02 \pm 5.06e+02$	-	$\mathbf{-1.29e+02 \pm 9.67e+02}$	-
f_5	$1.08e+04 \pm 6.01e+03$	$9.18e+03 \pm 1.59e+03$	=	$\mathbf{5.60e+02 \pm 5.63e+02}$	-	$2.74e+03 \pm 6.34e+02$	-
f_6	$5.28e+02 \pm 2.71e+02$	$4.63e+02 \pm 5.05e+01$	-	$\mathbf{4.00e+02 \pm 3.07e+01}$	-	$4.33e+02 \pm 4.81e+01$	-
f_7	$\mathbf{-1.80e+02 \pm 1.43e-02}$	$-1.80e+02 \pm 1.86e-02$	+	$-1.41e+02 \pm 1.16e+02$	=	$2.03e+06 \pm 1.88e+07$	+
f_8	$\mathbf{-1.20e+02 \pm 1.88e-03}$	$-1.19e+02 \pm 5.34e-02$	+	$-1.20e+02 \pm 9.08e-03$	+	$-1.19e+02 \pm 4.23e-01$	+
f_9	$-1.18e+02 \pm 5.17e-01$	$-1.18e+02 \pm 1.98e-01$	=	$\mathbf{-3.30e+02 \pm 7.35e-01}$	-	$-1.17e+02 \pm 1.14e+00$	+
f_{10}	$2.76e+02 \pm 2.45e+01$	$2.55e+02 \pm 1.98e+01$	-	$\mathbf{-2.96e+02 \pm 2.17e+01}$	-	$2.23e+02 \pm 2.44e+01$	-
f_{11}	$1.23e+02 \pm 5.69e+00$	$1.18e+02 \pm 2.48e+00$	-	$1.14e+02 \pm 3.18e+00$	-	$\mathbf{1.11e+02 \pm 4.59e+00}$	-
f_{12}	$1.02e+03 \pm 2.18e+03$	$3.36e+03 \pm 4.99e+03$	+	$\mathbf{2.46e+02 \pm 1.07e+03}$	-	$3.77e+03 \pm 3.87e+03$	+
f_{13}	$-1.19e+02 \pm 4.66e+00$	$\mathbf{-1.27e+02 \pm 1.90e-01}$	-	$-1.27e+02 \pm 1.96e+00$	-	$-1.19e+02 \pm 2.28e+00$	=
f_{14}	$-2.86e+02 \pm 5.60e-01$	$-2.87e+02 \pm 2.89e-01$	-	$-2.87e+02 \pm 3.18e-01$	-	$\mathbf{-2.87e+02 \pm 4.50e-01}$	-
f_{15}	$1.44e+03 \pm 8.47e-01$	$1.44e+03 \pm 1.25e-01$	=	$\mathbf{4.27e+02 \pm 2.92e+01}$	-	$1.46e+03 \pm 6.43e+00$	+
f_{16}	$1.60e+03 \pm 2.52e+01$	$1.58e+03 \pm 9.17e+00$	-	$\mathbf{2.61e+02 \pm 1.62e+02}$	-	$1.58e+03 \pm 1.11e+01$	-
f_{17}	$1.67e+03 \pm 1.90e+01$	$1.70e+03 \pm 1.29e+01$	+	$\mathbf{3.05e+02 \pm 1.68e+02}$	-	$1.62e+03 \pm 9.04e+00$	-
f_{18}	$9.10e+02 \pm 5.70e-12$	$\mathbf{9.10e+02 \pm 1.58e-12}$	=	$9.11e+02 \pm 3.88e+01$	+	$9.10e+02 \pm 8.31e-11$	+
f_{19}	$9.10e+02 \pm 5.63e-12$	$9.10e+02 \pm 1.41e-12$	=	$\mathbf{9.07e+02 \pm 4.09e+01}$	-	$9.10e+02 \pm 2.42e-10$	+
f_{20}	$9.10e+02 \pm 5.52e-12$	$9.10e+02 \pm 1.65e-12$	=	$\mathbf{9.09e+02 \pm 3.94e+01}$	-	$9.10e+02 \pm 3.41e-11$	+
f_{21}	$1.72e+03 \pm 1.51e+01$	$1.71e+03 \pm 5.62e+00$	=	$\mathbf{8.82e+02 \pm 1.13e+02}$	-	$1.70e+03 \pm 5.51e+00$	-
f_{22}	$2.62e+03 \pm 8.11e+01$	$2.49e+03 \pm 2.85e+01$	-	$\mathbf{1.25e+03 \pm 1.31e+01}$	-	$2.41e+03 \pm 4.97e+01$	-
f_{23}	$1.72e+03 \pm 1.07e+01$	$1.71e+03 \pm 4.86e+00$	=	$\mathbf{9.00e+02 \pm 6.27e+01}$	-	$1.70e+03 \pm 5.28e+00$	-
f_{24}	$1.72e+03 \pm 1.39e+01$	$1.70e+03 \pm 7.97e+00$	-	$\mathbf{4.60e+02 \pm 0.00e+00}$	-	$1.67e+03 \pm 1.55e+01$	-
f_{25}	$\mathbf{1.65e+03 \pm 2.55e+02}$	$1.85e+03 \pm 9.19e+01$	+	$1.89e+03 \pm 8.19e+00$	+	$1.83e+03 \pm 1.55e+02$	+

Table E.29. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against state-of-the-art algorithms on BBOB2010 in 100 dimensions.

	PMS	CCPSO2		MA-LSCh-CMA		MDE-pBX	
f_1	$\mathbf{7.95e+01 \pm 3.65e-14}$	$7.95e+01 \pm 1.81e-13$	=	$7.95e+01 \pm 6.55e-10$	+	$7.95e+01 \pm 7.60e-05$	+
f_2	$\mathbf{-2.10e+02 \pm 7.13e-14}$	$-2.10e+02 \pm 1.77e-12$	+	$-2.10e+02 \pm 5.31e-06$	+	$-2.10e+02 \pm 6.06e-03$	+
f_3	$-3.67e+02 \pm 1.66e+01$	$\mathbf{-4.54e+02 \pm 8.40e+00}$	-	$-4.14e+02 \pm 9.03e+00$	-	$3.29e+01 \pm 7.94e+01$	+
f_4	$-3.40e+02 \pm 2.05e+01$	$\mathbf{-4.40e+02 \pm 1.31e+01}$	-	$-3.82e+02 \pm 1.14e+01$	-	$4.03e+02 \pm 1.31e+02$	+
f_5	$-9.21e+00 \pm 3.51e-12$	$-9.21e+00 \pm 1.20e-03$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	-	$-3.09e-02 \pm 1.27e+01$	+
f_6	$4.05e+01 \pm 4.61e+00$	$1.25e+02 \pm 4.02e+01$	+	$\mathbf{3.59e+01 \pm 1.60e-03}$	-	$8.03e+01 \pm 3.32e+01$	+
f_7	$3.70e+02 \pm 8.80e+01$	$4.38e+02 \pm 4.90e+01$	+	$\mathbf{1.33e+02 \pm 7.77e+00}$	-	$3.70e+02 \pm 7.43e+01$	=
f_8	$2.21e+02 \pm 5.96e+01$	$2.70e+02 \pm 3.42e+01$	+	$\mathbf{1.59e+02 \pm 1.45e+01}$	-	$3.40e+02 \pm 6.77e+01$	+
f_9	$\mathbf{1.72e+02 \pm 2.30e+01}$	$2.30e+02 \pm 2.78e+01$	+	$1.80e+02 \pm 8.49e+00$	+	$2.52e+02 \pm 3.75e+01$	+
f_{10}	$\mathbf{2.37e+03 \pm 6.64e+02}$	$2.61e+04 \pm 6.64e+03$	+	$2.40e+03 \pm 1.44e+03$	=	$1.64e+04 \pm 7.99e+03$	+
f_{11}	$6.51e+02 \pm 8.17e+01$	$6.21e+02 \pm 1.95e+02$	-	$1.85e+02 \pm 1.85e+01$	-	$\mathbf{9.16e+01 \pm 7.45e+00}$	-
f_{12}	$-6.11e+02 \pm 1.52e+01$	$-6.13e+02 \pm 1.20e+01$	=	$\mathbf{-6.21e+02 \pm 6.95e-01}$	-	$-5.99e+02 \pm 7.07e+01$	+
f_{13}	$3.60e+01 \pm 6.01e+00$	$3.31e+01 \pm 4.20e+00$	-	$\mathbf{3.11e+01 \pm 1.76e+00}$	-	$3.47e+01 \pm 6.70e+00$	-
f_{14}	$-5.23e+01 \pm 1.80e-05$	$-5.23e+01 \pm 2.34e-04$	+	$\mathbf{-5.23e+01 \pm 6.00e-05}$	-	$-5.23e+01 \pm 2.55e-03$	+
f_{15}	$2.39e+03 \pm 6.32e+02$	$2.33e+03 \pm 2.32e+02$	-	$\mathbf{1.41e+03 \pm 1.88e+02}$	-	$1.66e+03 \pm 1.10e+02$	-
f_{16}	$9.23e+01 \pm 9.50e+00$	$9.88e+01 \pm 4.27e+00$	+	$\mathbf{8.38e+01 \pm 1.79e+00}$	-	$8.85e+01 \pm 4.46e+00$	-
f_{17}	$-1.96e+00 \pm 9.61e+00$	$-8.29e+00 \pm 1.62e+00$	-	$\mathbf{-1.69e+01 \pm 4.95e-02}$	-	$-1.35e+01 \pm 4.83e-01$	-
f_{18}	$4.89e+01 \pm 4.41e+01$	$1.61e+01 \pm 6.61e+00$	-	$\mathbf{-1.64e+01 \pm 1.89e-01}$	-	$-4.84e+00 \pm 1.68e+00$	-
f_{19}	$-5.08e+01 \pm 4.81e+01$	$-9.46e+01 \pm 1.29e+00$	-	$\mathbf{-1.00e+02 \pm 1.17e+00}$	-	$-1.00e+02 \pm 7.13e-01$	-
f_{20}	$-5.45e+02 \pm 1.28e-01$	$\mathbf{-5.46e+02 \pm 6.73e-02}$	-	$-5.45e+02 \pm 9.48e-02$	+	$-5.44e+02 \pm 1.14e-01$	+
f_{21}	$5.05e+01 \pm 1.12e+01$	$\mathbf{4.41e+01 \pm 3.38e+00}$	-	$4.55e+01 \pm 7.35e+00$	-	$4.49e+01 \pm 5.88e+00$	-
f_{22}	$-9.83e+02 \pm 1.29e+01$	$\mathbf{-9.95e+02 \pm 5.83e+00}$	-	$-9.92e+02 \pm 8.63e+00$	-	$-9.92e+02 \pm 9.11e+00$	-
f_{23}	$8.70e+00 \pm 8.16e-01$	$9.39e+00 \pm 4.22e-01$	+	$\mathbf{7.66e+00 \pm 2.66e-01}$	-	$9.34e+00 \pm 7.99e-01$	+
f_{24}	$2.14e+03 \pm 6.37e+02$	$1.19e+03 \pm 1.53e+02$	-	$\mathbf{4.15e+02 \pm 1.12e+02}$	-	$4.75e+02 \pm 4.72e+01$	-

Table E.30. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against state-of-the-art algorithms on CEC2008 in 1000 dimensions.

	PMS	CCPSO2		MA-LSCh-SSW		MDE-pBX	
f_1	$\mathbf{-4.50e+02 \pm 4.20e-13}$	$-4.50e+02 \pm 3.54e-12$	+	$6.18e+04 \pm 6.12e+05$	+	$1.20e+05 \pm 4.41e+04$	+
f_2	$-3.88e+02 \pm 5.17e+01$	$\mathbf{-4.09e+02 \pm 2.46e+01}$	-	$-2.66e+02 \pm 1.81e+00$	+	$-3.33e+02 \pm 4.09e+00$	+
f_3	$\mathbf{1.34e+03 \pm 5.37e+02}$	$1.80e+03 \pm 1.12e+02$	+	$1.45e+11 \pm 1.01e+12$	+	$3.13e+10 \pm 1.65e+10$	+
f_4	$\mathbf{-3.30e+02 \pm 1.33e-12}$	$-2.00e+02 \pm 1.07e+02$	+	$1.53e+04 \pm 1.02e+03$	+	$7.60e+03 \pm 2.55e+02$	+
f_5	$-1.80e+02 \pm 1.50e-02$	$\mathbf{-1.80e+02 \pm 2.98e-03}$	=	$-1.72e+02 \pm 1.09e+01$	+	$1.08e+03 \pm 4.60e+02$	+
f_6	$-1.38e+02 \pm 6.36e+00$	$\mathbf{-1.40e+02 \pm 9.34e-11}$	-	$-1.20e+02 \pm 3.19e-01$	+	$-1.21e+02 \pm 5.10e-02$	+
f_7	$-1.37e+04 \pm 2.80e+02$	$\mathbf{-1.44e+04 \pm 7.46e+01}$	-	$-9.97e+03 \pm 8.06e+02$	+	$-1.11e+04 \pm 1.63e+02$	+

Table E.31. Average Fitness ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = PMS) for PMS against state-of-the-art algorithms on CEC2010 in 1000 dimensions.

	PMS	CCPSO2		MA-LSCh-SSW		MDE-pBX	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$6.47e-14 \pm 1.41e-13$	+	$1.39e+11 \pm 3.63e+10$	+	$1.05e+09 \pm 6.58e+08$	+
f_2	$\mathbf{1.48e-13 \pm 2.79e-13}$	$1.36e+02 \pm 1.11e+02$	+	$1.46e+04 \pm 1.15e+03$	+	$7.02e+03 \pm 2.38e+02$	+
f_3	$4.52e-01 \pm 2.76e+00$	$\mathbf{7.34e-11 \pm 1.05e-10}$	-	$2.00e+01 \pm 3.56e-01$	+	$1.93e+01 \pm 4.76e-02$	+
f_4	$\mathbf{5.27e+11 \pm 2.72e+11}$	$2.14e+12 \pm 1.27e+12$	+	$5.91e+14 \pm 6.23e+14$	+	$3.21e+12 \pm 9.76e+11$	+
f_5	$4.78e+08 \pm 1.40e+08$	$3.92e+08 \pm 7.98e+07$	-	$7.76e+08 \pm 6.82e+07$	+	$\mathbf{1.54e+08 \pm 2.77e+07}$	-
f_6	$1.92e+07 \pm 2.24e+06$	$1.71e+07 \pm 4.45e+06$	-	$2.03e+07 \pm 2.27e+05$	+	$\mathbf{3.65e+06 \pm 1.75e+06}$	-
f_7	$1.02e+08 \pm 2.57e+08$	$7.60e+09 \pm 9.72e+09$	+	$2.81e+11 \pm 4.11e+11$	+	$\mathbf{6.79e+06 \pm 1.01e+07}$	-
f_8	$1.17e+08 \pm 1.30e+08$	$\mathbf{5.46e+07 \pm 4.16e+07}$	-	$2.34e+16 \pm 2.01e+16$	+	$2.03e+08 \pm 1.63e+08$	+
f_9	$\mathbf{6.19e+06 \pm 2.80e+06}$	$5.01e+07 \pm 7.68e+06$	+	$7.94e+09 \pm 5.14e+10$	+	$1.68e+09 \pm 1.00e+09$	+
f_{10}	$5.25e+03 \pm 2.09e+03$	$\mathbf{4.57e+03 \pm 2.75e+02}$	=	$1.47e+04 \pm 2.77e+02$	+	$7.33e+03 \pm 2.55e+02$	+
f_{11}	$\mathbf{1.85e+02 \pm 3.03e+01}$	$2.00e+02 \pm 5.98e+00$	=	$2.19e+02 \pm 1.96e+00$	+	$2.06e+02 \pm 2.40e+00$	+
f_{12}	$\mathbf{1.06e+03 \pm 7.23e+02}$	$6.12e+04 \pm 8.14e+04$	+	$2.54e+05 \pm 9.03e+04$	+	$2.92e+05 \pm 6.60e+04$	+
f_{13}	$1.18e+03 \pm 6.33e+02$	$\mathbf{1.14e+03 \pm 5.42e+02}$	=	$3.43e+10 \pm 3.34e+11$	+	$2.88e+09 \pm 3.17e+09$	+
f_{14}	$\mathbf{1.44e+07 \pm 5.47e+06}$	$1.60e+08 \pm 3.35e+07$	+	$8.71e+09 \pm 5.52e+10$	+	$1.04e+09 \pm 1.97e+08$	+
f_{15}	$1.20e+04 \pm 3.87e+03$	$9.31e+03 \pm 5.52e+02$	-	$1.48e+04 \pm 1.02e+03$	=	$\mathbf{7.44e+03 \pm 2.80e+02}$	-
f_{16}	$\mathbf{3.27e+02 \pm 9.13e+01}$	$3.95e+02 \pm 1.45e+00$	=	$3.97e+02 \pm 3.26e+00$	+	$3.84e+02 \pm 1.22e+00$	=
f_{17}	$\mathbf{1.37e+03 \pm 8.23e+02}$	$1.41e+05 \pm 1.44e+05$	+	$1.63e+06 \pm 8.92e+06$	+	$4.35e+05 \pm 8.33e+04$	+
f_{18}	$\mathbf{2.41e+03 \pm 1.00e+03}$	$5.62e+03 \pm 4.13e+03$	+	$1.15e+10 \pm 8.80e+10$	+	$3.73e+10 \pm 1.95e+10$	+
f_{19}	$\mathbf{1.47e+05 \pm 4.93e+04}$	$1.14e+06 \pm 1.22e+06$	+	$1.50e+07 \pm 9.29e+06$	+	$9.22e+05 \pm 1.06e+05$	+
f_{20}	$\mathbf{9.57e+02 \pm 5.37e+02}$	$1.42e+03 \pm 1.19e+02$	+	$9.68e+10 \pm 8.63e+11$	+	$4.18e+10 \pm 2.02e+10$	+

E.5 Extended numerical results for SPAM

Table E.32. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on CEC2013 in 10 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$\mathbf{0.00e+00 \pm 0.00e+00}$	$0.00e+00 \pm 0.00e+00$	=	$3.37e-03 \pm 2.16e-02$	+	$0.00e+00 \pm 0.00e+00$	=
f_2	$\mathbf{0.00e+00 \pm 0.00e+00}$	$2.06e+03 \pm 4.90e+03$	+	$1.61e+06 \pm 1.21e+06$	+	$4.39e+02 \pm 4.16e+03$	+
f_3	$\mathbf{4.45e+01 \pm 2.61e+02}$	$9.43e+04 \pm 4.32e+05$	+	$6.19e+07 \pm 1.02e+08$	+	$3.19e+04 \pm 1.94e+05$	=
f_4	$\mathbf{0.00e+00 \pm 0.00e+00}$	$1.38e+00 \pm 7.24e+00$	+	$1.05e+04 \pm 2.93e+03$	+	$6.06e+01 \pm 4.97e+02$	+
f_5	$2.36e-08 \pm 2.77e-08$	$\mathbf{0.00e+00 \pm 8.73e-14}$	-	$1.25e-02 \pm 2.41e-02$	+	$8.59e-09 \pm 4.44e-08$	-
f_6	$3.94e+00 \pm 4.67e+00$	$5.71e+00 \pm 4.83e+00$	=	$4.56e+00 \pm 4.46e+00$	+	$\mathbf{5.14e-02 \pm 4.11e-01}$	-
f_7	$6.67e+01 \pm 5.65e+01$	$7.99e+00 \pm 1.04e+01$	-	$4.16e+01 \pm 1.25e+01$	-	$\mathbf{1.86e+00 \pm 3.52e+00}$	-
f_8	$\mathbf{2.03e+01 \pm 1.88e-01}$	$2.05e+01 \pm 1.07e-01$	+	$2.04e+01 \pm 8.82e-02$	+	$2.04e+01 \pm 1.01e-01$	+
f_9	$6.09e+00 \pm 3.38e+00$	$2.11e+00 \pm 1.47e+00$	-	$5.58e+00 \pm 1.01e+00$	-	$\mathbf{1.78e+00 \pm 9.45e-01}$	-
f_{10}	$\mathbf{1.49e-02 \pm 1.24e-02}$	$1.16e-01 \pm 9.01e-02$	+	$2.08e+00 \pm 1.15e+00$	+	$1.47e-01 \pm 1.68e-01$	+
f_{11}	$4.17e+00 \pm 1.67e+00$	$2.86e+00 \pm 1.86e+00$	-	$3.44e+00 \pm 2.02e+00$	-	$\mathbf{1.96e+00 \pm 1.16e+00}$	-
f_{12}	$1.44e+01 \pm 6.82e+00$	$1.13e+01 \pm 4.99e+00$	-	$3.18e+01 \pm 9.05e+00$	+	$\mathbf{5.25e+00 \pm 3.31e+00}$	-
f_{13}	$8.02e+01 \pm 9.29e+01$	$1.89e+01 \pm 9.18e+00$	-	$4.05e+01 \pm 8.05e+00$	=	$\mathbf{7.75e+00 \pm 4.15e+00}$	-
f_{14}	$1.33e+02 \pm 9.00e+01$	$1.09e+02 \pm 1.07e+02$	-	$8.02e+01 \pm 5.47e+01$	-	$\mathbf{4.22e+01 \pm 6.04e+01}$	-
f_{15}	$7.14e+02 \pm 1.99e+02$	$7.42e+02 \pm 2.77e+02$	=	$1.02e+03 \pm 2.89e+02$	+	$\mathbf{5.78e+02 \pm 2.04e+02}$	-
f_{16}	$\mathbf{3.19e-01 \pm 2.33e-01}$	$5.80e-01 \pm 4.09e-01$	+	$1.31e+00 \pm 2.81e-01$	+	$4.88e+00 \pm 1.13e+00$	+
f_{17}	$\mathbf{1.12e+01 \pm 3.79e+00}$	$1.32e+01 \pm 1.82e+00$	+	$1.74e+01 \pm 2.48e+00$	+	$2.62e+02 \pm 3.30e+01$	+
f_{18}	$6.83e+01 \pm 8.47e+01$	$\mathbf{2.01e+01 \pm 5.56e+00}$	-	$5.92e+01 \pm 8.00e+00$	-	$3.43e+02 \pm 4.64e+01$	+
f_{19}	$9.05e-01 \pm 3.26e-01$	$\mathbf{6.09e-01 \pm 2.13e-01}$	-	$1.01e+00 \pm 3.27e-01$	+	$2.76e+02 \pm 1.37e+02$	+
f_{20}	$3.92e+00 \pm 4.48e-01$	$\mathbf{2.87e+00 \pm 6.22e-01}$	-	$3.60e+00 \pm 2.21e-01$	-	$4.99e+00 \pm 5.33e-02$	+
f_{21}	$\mathbf{2.43e+02 \pm 1.19e+02}$	$3.98e+02 \pm 1.99e+01$	+	$3.88e+02 \pm 3.73e+01$	+	$6.36e+02 \pm 4.30e+01$	+
f_{22}	$2.70e+02 \pm 2.26e+02$	$1.79e+02 \pm 1.41e+02$	-	$\mathbf{1.13e+02 \pm 6.86e+01}$	-	$7.07e+02 \pm 2.14e+02$	+
f_{23}	$9.97e+02 \pm 3.34e+02$	$9.21e+02 \pm 3.68e+02$	=	$1.31e+03 \pm 2.94e+02$	+	$\mathbf{8.83e+02 \pm 9.94e+01}$	-
f_{24}	$\mathbf{1.37e+02 \pm 3.81e+01}$	$2.03e+02 \pm 1.05e+01$	+	$2.08e+02 \pm 1.99e+01$	+	$2.56e+02 \pm 1.08e+01$	+
f_{25}	$\mathbf{1.99e+02 \pm 3.06e+01}$	$2.01e+02 \pm 8.76e+00$	+	$2.15e+02 \pm 8.90e+00$	+	$2.48e+02 \pm 5.43e+00$	+
f_{26}	$\mathbf{1.48e+02 \pm 4.29e+01}$	$1.55e+02 \pm 4.37e+01$	=	$1.69e+02 \pm 2.62e+01$	+	$3.35e+02 \pm 4.65e+01$	+
f_{27}	$3.56e+02 \pm 8.99e+01$	$\mathbf{3.13e+02 \pm 4.68e+01}$	-	$4.46e+02 \pm 6.43e+01$	+	$1.03e+03 \pm 1.27e+02$	+
f_{28}	$\mathbf{2.38e+02 \pm 9.25e+01}$	$3.01e+02 \pm 6.82e+01$	+	$3.87e+02 \pm 1.80e+02$	+	$1.30e+03 \pm 7.99e+01$	+

Table E.33. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on CEC2005 in 30 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$\mathbf{0.00e+00 \pm 4.96e-14}$	$5.68e-14 \pm 4.13e-13$	+	$1.71e-13 \pm 4.62e-13$	+	$8.90e-09 \pm 8.36e-10$	+
f_2	$\mathbf{0.00e+00 \pm 5.45e-14}$	$9.75e-04 \pm 4.18e-03$	+	$9.47e+00 \pm 1.66e+01$	+	$1.57e-03 \pm 1.47e-02$	+
f_3	$\mathbf{1.87e+03 \pm 1.39e+03}$	$3.23e+05 \pm 2.14e+05$	+	$1.53e+06 \pm 6.74e+05$	+	$1.46e+05 \pm 4.32e+05$	+
f_4	$2.47e+04 \pm 9.22e+03$	$\mathbf{2.31e+02 \pm 6.07e+02}$	-	$1.80e+04 \pm 4.47e+03$	-	$7.71e+02 \pm 5.06e+02$	-
f_5	$\mathbf{8.21e+02 \pm 4.69e+02}$	$2.99e+03 \pm 5.85e+02$	+	$9.44e+03 \pm 1.72e+03$	+	$8.70e+02 \pm 5.63e+02$	=
f_6	$4.69e+01 \pm 9.38e+01$	$3.38e+01 \pm 3.37e+01$	-	$6.99e+01 \pm 5.60e+01$	+	$\mathbf{9.54e+00 \pm 3.07e+01}$	-
f_7	$\mathbf{6.41e-04 \pm 2.19e-03}$	$3.16e+04 \pm 2.04e+05$	+	$2.49e-02 \pm 2.10e-02$	+	$3.86e+01 \pm 1.16e+02$	+
f_8	$\mathbf{2.00e+01 \pm 8.76e-04}$	$2.07e+01 \pm 4.12e-01$	+	$2.10e+01 \pm 6.31e-02$	+	$2.00e+01 \pm 9.08e-03$	+
f_9	$\mathbf{2.13e-10 \pm 5.06e-11}$	$4.02e+01 \pm 1.09e+01$	+	$5.67e-01 \pm 6.74e-01$	+	$2.09e-01 \pm 7.35e-01$	+
f_{10}	$6.83e+01 \pm 3.22e+01$	$7.31e+01 \pm 1.88e+01$	+	$2.01e+02 \pm 5.34e+01$	+	$\mathbf{3.36e+01 \pm 2.17e+01}$	-
f_{11}	$\mathbf{1.35e+01 \pm 5.15e+00}$	$2.10e+01 \pm 4.91e+00$	+	$2.80e+01 \pm 2.36e+00$	+	$2.36e+01 \pm 3.18e+00$	+
f_{12}	$\mathbf{4.29e+02 \pm 7.94e+02}$	$3.95e+03 \pm 3.90e+03$	+	$3.51e+03 \pm 3.78e+03$	+	$7.06e+02 \pm 1.07e+03$	=
f_{13}	$2.79e+00 \pm 6.01e-01$	$3.49e+00 \pm 1.16e+00$	+	$\mathbf{8.97e-01 \pm 1.83e-01}$	-	$3.07e+00 \pm 1.96e+00$	=
f_{14}	$4.39e+01 \pm 2.75e-01$	$\mathbf{4.28e+01 \pm 4.14e-01}$	-	$4.30e+01 \pm 2.93e-01$	-	$4.29e+01 \pm 3.18e-01$	-
f_{15}	$\mathbf{1.28e+02 \pm 1.13e+02}$	$3.55e+02 \pm 7.82e+01$	+	$2.44e+02 \pm 1.45e+02$	+	$3.07e+02 \pm 2.92e+01$	+
f_{16}	$\mathbf{1.36e+02 \pm 5.52e+01}$	$1.79e+02 \pm 1.40e+02$	=	$2.88e+02 \pm 1.05e+02$	+	$1.41e+02 \pm 1.62e+02$	+
f_{17}	$2.22e+02 \pm 5.44e+01$	$\mathbf{1.81e+02 \pm 1.38e+02}$	-	$3.77e+02 \pm 7.25e+01$	+	$1.85e+02 \pm 1.68e+02$	-
f_{18}	$9.03e+02 \pm 3.04e+01$	$\mathbf{8.99e+02 \pm 5.98e+01}$	-	$9.30e+02 \pm 1.31e+01$	+	$9.01e+02 \pm 3.88e+01$	-
f_{19}	$\mathbf{8.95e+02 \pm 4.00e+01}$	$8.99e+02 \pm 5.81e+01$	+	$9.28e+02 \pm 1.41e+01$	+	$8.97e+02 \pm 4.09e+01$	+
f_{20}	$9.00e+02 \pm 5.68e+01$	$\mathbf{8.97e+02 \pm 5.96e+01}$	-	$9.24e+02 \pm 2.88e+01$	+	$8.99e+02 \pm 3.94e+01$	-
f_{21}	$\mathbf{4.99e+02 \pm 9.02e+00}$	$5.63e+02 \pm 1.91e+02$	+	$5.04e+02 \pm 3.07e+01$	+	$5.22e+02 \pm 1.13e+02$	+
f_{22}	$9.04e+02 \pm 2.92e+01$	$9.74e+02 \pm 2.88e+01$	+	$1.06e+03 \pm 2.97e+01$	+	$\mathbf{8.89e+02 \pm 1.31e+01}$	-
f_{23}	$\mathbf{5.32e+02 \pm 2.00e+01}$	$5.91e+02 \pm 1.48e+02$	+	$5.32e+02 \pm 2.00e+01$	+	$5.40e+02 \pm 6.27e+01$	=
f_{24}	$2.00e+02 \pm 3.80e-11$	$2.10e+02 \pm 9.81e+01$	+	$2.00e+02 \pm 1.67e-06$	+	$\mathbf{2.00e+02 \pm 0.00e+00}$	-
f_{25}	$\mathbf{1.24e+03 \pm 4.43e+02}$	$1.59e+03 \pm 1.32e+02$	+	$1.59e+03 \pm 6.89e+01$	+	$1.63e+03 \pm 8.19e+00$	+

Table E.34. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on CEC2013 in 50 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$2.27e-13 \pm 2.27e-14$	$5.58e-10 \pm 4.68e-09$	+	$1.36e-12 \pm 3.64e-12$	+	$\mathbf{0.00e+00 \pm 0.00e+00}$	-
f_2	$\mathbf{3.76e+04 \pm 1.97e+04}$	$8.12e+05 \pm 4.07e+05$	+	$4.43e+06 \pm 2.70e+06$	+	$4.14e+05 \pm 4.67e+05$	+
f_3	$7.86e+06 \pm 1.32e+07$	$1.63e+08 \pm 2.00e+08$	+	$3.47e+09 \pm 3.70e+09$	+	$\mathbf{3.91e+06 \pm 1.07e+07}$	-
f_4	$1.20e+03 \pm 4.28e+03$	$\mathbf{9.70e+02 \pm 7.38e+02}$	-	$1.08e+05 \pm 3.85e+04$	+	$2.85e+04 \pm 8.23e+03$	+
f_5	$\mathbf{4.29e-07 \pm 5.98e-08}$	$2.11e-06 \pm 1.97e-05$	+	$5.73e-05 \pm 5.68e-04$	+	$1.19e-06 \pm 4.52e-06$	+
f_6	$\mathbf{2.61e+01 \pm 1.90e+01}$	$5.81e+01 \pm 2.32e+01$	+	$4.86e+01 \pm 1.64e+01$	+	$3.77e+01 \pm 1.14e+01$	+
f_7	$4.78e+01 \pm 2.15e+01$	$6.94e+01 \pm 1.27e+01$	+	$1.44e+02 \pm 2.81e+01$	+	$\mathbf{1.42e+01 \pm 5.65e+00}$	-
f_8	$\mathbf{2.11e+01 \pm 6.29e-02}$	$2.12e+01 \pm 4.66e-02$	+	$2.12e+01 \pm 3.78e-02$	+	$2.12e+01 \pm 3.48e-02$	+
f_9	$4.77e+01 \pm 3.86e+00$	$4.42e+01 \pm 8.00e+00$	-	$5.88e+01 \pm 3.72e+00$	+	$\mathbf{2.48e+01 \pm 1.27e+01}$	-
f_{10}	$\mathbf{1.08e-02 \pm 7.37e-03}$	$4.03e-01 \pm 5.14e-01$	+	$2.08e-01 \pm 2.82e-01$	+	$1.58e-01 \pm 1.41e-01$	+
f_{11}	$5.03e+01 \pm 1.03e+01$	$1.23e+02 \pm 2.63e+01$	+	$\mathbf{9.48e-01 \pm 8.55e-01}$	-	$1.86e+01 \pm 6.22e+00$	-
f_{12}	$3.17e+02 \pm 1.21e+02$	$1.59e+02 \pm 3.72e+01$	-	$4.63e+02 \pm 9.91e+01$	+	$\mathbf{1.39e+02 \pm 8.05e+01}$	-
f_{13}	$4.88e+02 \pm 8.32e+01$	$3.18e+02 \pm 5.58e+01$	-	$5.70e+02 \pm 8.52e+01$	+	$\mathbf{1.83e+02 \pm 5.41e+01}$	-
f_{14}	$1.28e+03 \pm 2.81e+02$	$2.74e+03 \pm 8.02e+02$	+	$\mathbf{6.98e+00 \pm 3.64e+00}$	-	$1.97e+03 \pm 2.41e+03$	+
f_{15}	$\mathbf{6.48e+03 \pm 5.37e+02}$	$7.58e+03 \pm 8.91e+02$	+	$8.25e+03 \pm 8.01e+02$	+	$7.62e+03 \pm 9.47e+02$	+
f_{16}	$\mathbf{8.59e-02 \pm 3.78e-02}$	$1.85e+00 \pm 8.03e-01$	+	$2.71e+00 \pm 5.64e-01$	+	$7.43e+00 \pm 9.69e-01$	+
f_{17}	$9.42e+01 \pm 9.78e+00$	$1.76e+02 \pm 3.52e+01$	+	$\mathbf{5.17e+01 \pm 4.23e-01}$	-	$2.79e+02 \pm 7.07e+01$	+
f_{18}	$5.08e+02 \pm 8.68e+01$	$\mathbf{1.94e+02 \pm 3.77e+01}$	-	$5.08e+02 \pm 1.03e+02$	=	$3.95e+02 \pm 5.05e+00$	-
f_{19}	$5.05e+00 \pm 9.04e-01$	$4.27e+01 \pm 2.78e+01$	+	$\mathbf{1.46e+00 \pm 2.30e-01}$	-	$4.71e+02 \pm 5.91e+01$	+
f_{20}	$2.43e+01 \pm 5.55e-01$	$\mathbf{2.00e+01 \pm 1.03e+00}$	-	$2.32e+01 \pm 7.70e-01$	-	$2.50e+01 \pm 1.34e-05$	+
f_{21}	$4.37e+02 \pm 3.36e+02$	$8.54e+02 \pm 3.66e+02$	+	$\mathbf{4.06e+02 \pm 3.22e+02}$	=	$9.64e+02 \pm 1.81e+02$	+
f_{22}	$1.93e+03 \pm 4.31e+02$	$3.09e+03 \pm 9.77e+02$	+	$\mathbf{9.40e+01 \pm 9.52e+01}$	-	$7.62e+02 \pm 2.05e+02$	-
f_{23}	$\mathbf{8.78e+03 \pm 1.26e+03}$	$9.03e+03 \pm 1.29e+03$	=	$1.07e+04 \pm 1.17e+03$	+	$8.88e+03 \pm 1.67e+03$	=
f_{24}	$3.25e+02 \pm 2.43e+01$	$\mathbf{2.85e+02 \pm 1.44e+01}$	-	$3.61e+02 \pm 1.04e+01$	+	$8.62e+02 \pm 1.08e+02$	+
f_{25}	$\mathbf{3.68e+02 \pm 1.34e+01}$	$3.68e+02 \pm 1.45e+01$	=	$4.00e+02 \pm 1.07e+01$	+	$6.79e+02 \pm 3.82e+01$	+
f_{26}	$2.24e+02 \pm 6.92e+01$	$3.48e+02 \pm 8.03e+01$	+	$\mathbf{2.22e+02 \pm 6.42e+01}$	-	$5.98e+02 \pm 7.45e+01$	+
f_{27}	$1.28e+03 \pm 1.97e+02$	$1.24e+03 \pm 1.37e+02$	=	$1.82e+03 \pm 1.67e+02$	+	$\mathbf{1.18e+03 \pm 3.70e+02}$	=
f_{28}	$1.42e+03 \pm 1.96e+03$	$7.52e+02 \pm 1.06e+03$	-	$\mathbf{5.77e+02 \pm 7.84e+02}$	-	$1.40e+03 \pm 3.98e+02$	-

Table E.35. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on BBOB2010 in 100 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$\mathbf{2.42e-13 \pm 2.13e-13}$	$7.73e-06 \pm 7.60e-05$	+	$4.12e-13 \pm 1.97e-13$	+	$9.30e-09 \pm 6.55e-10$	+
f_2	$3.21e-07 \pm 7.33e-08$	$1.11e-03 \pm 6.06e-03$	+	$\mathbf{9.66e-13 \pm 1.77e-12}$	-	$1.44e-06 \pm 5.31e-06$	+
f_3	$9.60e+01 \pm 1.24e+01$	$4.95e+02 \pm 7.94e+01$	+	$\mathbf{8.09e+00 \pm 8.40e+00}$	-	$4.86e+01 \pm 9.03e+00$	-
f_4	$1.28e+02 \pm 1.86e+01$	$8.65e+02 \pm 1.31e+02$	+	$\mathbf{2.24e+01 \pm 1.31e+01}$	-	$8.02e+01 \pm 1.14e+01$	-
f_5	$1.31e-04 \pm 1.27e-05$	$9.18e+00 \pm 1.27e+01$	+	$2.41e-04 \pm 1.20e-03$	+	$\mathbf{1.90e-13 \pm 0.00e+00}$	-
f_6	$\mathbf{3.93e-08 \pm 3.87e-08}$	$4.44e+01 \pm 3.32e+01$	+	$8.94e+01 \pm 4.02e+01$	+	$3.25e-04 \pm 1.60e-03$	=
f_7	$5.07e+01 \pm 1.35e+01$	$2.77e+02 \pm 7.43e+01$	+	$3.45e+02 \pm 4.90e+01$	+	$\mathbf{4.04e+01 \pm 7.77e+00}$	-
f_8	$3.45e+01 \pm 1.31e+01$	$1.91e+02 \pm 6.77e+01$	+	$1.20e+02 \pm 3.42e+01$	+	$\mathbf{9.41e+00 \pm 1.45e+01}$	-
f_9	$\mathbf{4.70e+01 \pm 1.14e+01}$	$1.28e+02 \pm 3.75e+01$	+	$1.06e+02 \pm 2.78e+01$	+	$5.57e+01 \pm 8.49e+00$	+
f_{10}	$\mathbf{6.98e+02 \pm 2.21e+02}$	$1.65e+04 \pm 7.99e+03$	+	$2.62e+04 \pm 6.64e+03$	+	$2.45e+03 \pm 1.44e+03$	+
f_{11}	$8.15e+01 \pm 2.76e+01$	$\mathbf{1.54e+01 \pm 7.45e+00}$	-	$5.45e+02 \pm 1.95e+02$	+	$1.09e+02 \pm 1.85e+01$	+
f_{12}	$\mathbf{1.97e-02 \pm 6.48e-02}$	$2.21e+01 \pm 7.07e+01$	+	$7.95e+00 \pm 1.20e+01$	+	$1.43e-01 \pm 6.95e-01$	=
f_{13}	$\mathbf{7.74e-01 \pm 9.16e-01}$	$4.76e+00 \pm 6.70e+00$	+	$3.16e+00 \pm 4.20e+00$	+	$1.12e+00 \pm 1.76e+00$	=
f_{14}	$4.21e-05 \pm 8.74e-06$	$2.73e-03 \pm 2.55e-03$	+	$1.32e-03 \pm 2.34e-04$	+	$\mathbf{3.74e-05 \pm 6.00e-05}$	-
f_{15}	$\mathbf{2.68e+02 \pm 3.86e+01}$	$6.57e+02 \pm 1.10e+02$	+	$1.33e+03 \pm 2.32e+02$	+	$4.14e+02 \pm 1.88e+02$	+
f_{16}	$\mathbf{2.35e+00 \pm 8.17e-01}$	$1.71e+01 \pm 4.46e+00$	+	$2.74e+01 \pm 4.27e+00$	+	$1.25e+01 \pm 1.79e+00$	+
f_{17}	$7.93e+00 \pm 3.86e+00$	$3.42e+00 \pm 4.83e-01$	-	$8.65e+00 \pm 1.62e+00$	+	$\mathbf{7.23e-02 \pm 4.95e-02}$	-
f_{18}	$1.65e+01 \pm 9.86e+00$	$1.21e+01 \pm 1.68e+00$	-	$3.30e+01 \pm 6.61e+00$	+	$\mathbf{5.77e-01 \pm 1.89e-01}$	-
f_{19}	$\mathbf{1.66e+00 \pm 3.11e-01}$	$2.42e+00 \pm 7.13e-01$	+	$7.90e+00 \pm 1.29e+00$	+	$2.16e+00 \pm 1.17e+00$	=
f_{20}	$1.27e+00 \pm 1.47e-01$	$2.11e+00 \pm 1.14e-01$	+	$\mathbf{4.95e-01 \pm 6.73e-02}$	-	$1.54e+00 \pm 9.48e-02$	+
f_{21}	$3.42e+00 \pm 3.50e+00$	$4.15e+00 \pm 5.88e+00$	=	$\mathbf{3.36e+00 \pm 3.38e+00}$	=	$4.76e+00 \pm 7.35e+00$	=
f_{22}	$6.27e+00 \pm 7.42e+00$	$7.94e+00 \pm 9.11e+00$	+	$\mathbf{5.11e+00 \pm 5.83e+00}$	-	$7.58e+00 \pm 8.63e+00$	+
f_{23}	$\mathbf{6.62e-01 \pm 2.75e-01}$	$2.47e+00 \pm 7.99e-01$	+	$2.52e+00 \pm 4.22e-01$	+	$7.85e-01 \pm 2.66e-01$	+
f_{24}	$\mathbf{3.11e+02 \pm 5.78e+01}$	$3.72e+02 \pm 4.72e+01$	+	$1.08e+03 \pm 1.53e+02$	+	$3.13e+02 \pm 1.12e+02$	+

Table E.36. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on CEC2008 in 1000 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$\mathbf{5.12e-13 \pm 6.32e-13}$	$1.21e+05 \pm 4.41e+04$	+	$1.01e-11 \pm 3.57e-12$	+	$6.23e+04 \pm 6.12e+05$	+
f_2	$\mathbf{1.32e-01 \pm 3.27e-02}$	$1.17e+02 \pm 4.09e+00$	+	$4.10e+01 \pm 2.46e+01$	+	$1.84e+02 \pm 1.81e+00$	+
f_3	$\mathbf{8.97e+02 \pm 2.83e+01}$	$3.13e+10 \pm 1.65e+10$	+	$1.41e+03 \pm 1.12e+02$	+	$1.45e+11 \pm 1.01e+12$	+
f_4	$\mathbf{7.17e+01 \pm 1.16e+01}$	$7.93e+03 \pm 2.55e+02$	+	$1.30e+02 \pm 1.07e+02$	+	$1.56e+04 \pm 1.02e+03$	+
f_5	$\mathbf{7.56e-05 \pm 7.36e-04}$	$1.26e+03 \pm 4.60e+02$	+	$8.59e-04 \pm 2.98e-03$	+	$8.17e+00 \pm 1.09e+01$	+
f_6	$1.22e+01 \pm 9.51e+00$	$1.93e+01 \pm 5.10e-02$	+	$\mathbf{7.23e-11 \pm 9.34e-11}$	-	$1.99e+01 \pm 3.19e-01$	+

Table E.37. Average Error ± Standard Deviation and Wilcoxon Rank-Sum Test (reference = SPAM) for SPAM against MDE-pBX, CCPSO2 and MA-LSCh on CEC2010 in 1000 dimensions.

	SPAM	MDE-pBX		CCPSO2		MA-LSCh	
f_1	$1.92e-06 \pm 1.33e-07$	$1.05e+09 \pm 6.58e+08$	+	$\mathbf{6.47e-14 \pm 1.41e-13}$	-	$2.45e+11 \pm 4.07e+10$	+
f_2	$\mathbf{6.15e+01 \pm 1.28e+01}$	$7.02e+03 \pm 2.38e+02$	+	$1.36e+02 \pm 1.11e+02$	+	$1.97e+04 \pm 1.74e+03$	+
f_3	$8.06e+00 \pm 9.49e+00$	$1.93e+01 \pm 4.76e-02$	+	$\mathbf{7.34e-11 \pm 1.05e-10}$	-	$2.02e+01 \pm 4.21e-01$	+
f_4	$\mathbf{1.14e+11 \pm 4.32e+10}$	$3.21e+12 \pm 9.76e+11$	+	$2.14e+12 \pm 1.27e+12$	+	$1.24e+15 \pm 5.69e+14$	+
f_5	$6.78e+08 \pm 9.14e+07$	$\mathbf{1.54e+08 \pm 2.77e+07}$	-	$3.92e+08 \pm 7.98e+07$	-	$8.09e+08 \pm 6.58e+07$	+
f_6	$1.98e+07 \pm 7.24e+04$	$\mathbf{3.65e+06 \pm 1.75e+06}$	-	$1.71e+07 \pm 4.45e+06$	-	$2.03e+07 \pm 2.24e+05$	+
f_7	$\mathbf{4.17e+05 \pm 5.05e+05}$	$6.79e+06 \pm 1.01e+07$	+	$7.60e+09 \pm 9.72e+09$	+	$7.64e+11 \pm 5.09e+11$	+
f_8	$\mathbf{3.17e+06 \pm 2.82e+06}$	$2.03e+08 \pm 1.63e+08$	+	$5.46e+07 \pm 4.16e+07$	+	$5.34e+16 \pm 3.07e+16$	+
f_9	$\mathbf{3.21e+06 \pm 3.48e+05}$	$1.68e+09 \pm 1.00e+09$	+	$5.01e+07 \pm 7.68e+06$	+	$2.81e+11 \pm 4.08e+10$	+
f_{10}	$5.37e+03 \pm 3.84e+02$	$7.33e+03 \pm 2.55e+02$	+	$\mathbf{4.57e+03 \pm 2.75e+02}$	-	$1.85e+04 \pm 1.76e+03$	+
f_{11}	$2.00e+02 \pm 2.22e+01$	$2.06e+02 \pm 2.40e+00$	+	$\mathbf{2.00e+02 \pm 5.98e+00}$	-	$2.20e+02 \pm 3.44e+00$	+
f_{12}	$\mathbf{3.76e-03 \pm 1.40e-03}$	$2.92e+05 \pm 6.60e+04$	+	$6.12e+04 \pm 8.14e+04$	+	$1.69e+07 \pm 4.07e+06$	+
f_{13}	$1.22e+03 \pm 8.23e+02$	$2.88e+09 \pm 3.17e+09$	+	$\mathbf{1.14e+03 \pm 5.42e+02}$	=	$1.27e+12 \pm 5.77e+11$	+
f_{14}	$\mathbf{4.65e+06 \pm 4.18e+05}$	$1.04e+09 \pm 1.97e+08$	+	$1.60e+08 \pm 3.35e+07$	+	$3.23e+11 \pm 3.62e+10$	+
f_{15}	$1.00e+04 \pm 1.04e+03$	$\mathbf{7.44e+03 \pm 2.80e+02}$	-	$9.31e+03 \pm 5.52e+02$	-	$1.86e+04 \pm 1.58e+03$	+
f_{16}	$\mathbf{2.64e+02 \pm 1.26e+02}$	$3.84e+02 \pm 1.22e+00$	=	$3.95e+02 \pm 1.45e+00$	=	$4.00e+02 \pm 7.55e+00$	+
f_{17}	$\mathbf{4.36e-01 \pm 1.13e-01}$	$4.35e+05 \pm 8.33e+04$	+	$1.41e+05 \pm 1.44e+05$	+	$4.64e+07 \pm 1.30e+07$	+
f_{18}	$\mathbf{1.71e+03 \pm 1.09e+03}$	$3.73e+10 \pm 1.95e+10$	+	$5.62e+03 \pm 4.13e+03$	+	$5.38e+12 \pm 7.78e+11$	+
f_{19}	$\mathbf{3.32e+05 \pm 3.25e+04}$	$9.22e+05 \pm 1.06e+05$	+	$1.14e+06 \pm 1.22e+06$	+	$1.24e+08 \pm 3.39e+07$	+
f_{20}	$\mathbf{8.92e+02 \pm 3.28e+01}$	$4.18e+10 \pm 2.02e+10$	+	$1.42e+03 \pm 1.19e+02$	+	$6.07e+12 \pm 8.88e+11$	+

Bibliography

Abdi, H. (2010), 'Holms sequential bonferroni procedure', *Encyclopedia of research design* pp. 573–577.

Acampora, G., Cadenas, J. M., Loia, V. & Ballester, E. M. (2011), 'A multi-agent memetic system for human-based knowledge selection', *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **41**(5), 946–960.

Acampora, G., Gaeta, M. & Loia, V. (2011), 'Hierarchical optimization of personalized experiences for e-learning systems through evolutionary models', *Neural Computing and Applications* **20**(5), 641–657.

Akay, B. & Karaboga, D. (2012), 'Artificial bee colony algorithm for large-scale problems and engineering design optimization', *Journal of Intelligent Manufacturing* **23**(4), 1001–1014.

Auger, A. & Hansen, N. (2005), A restart cma evolution strategy with increasing population size, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 1769–1776.

Auger, A. & Teytaud, O. (2010), 'Continuous lunches are free plus the design of optimal optimization algorithms', *Algorithmica* **57**(1), 21–146.

Baluja, S. (1994), 'Population-based incremental learning', *Techical reports* .

Battiti, R. & Campigotto, P. (2012), An investigation of reinforcement learning for reactive search optimization, *in* Y. Hamadi, E. Monfroy & F. Saubion, eds, 'Autonomous Search', Springer, pp. 131–160.

Bazaraa, M. S., Sherali, H. D. & Shetty, C. M. (2006), *Nonlinear Programming: Theory And Algorithms*, Wiley-Interscience.

Beni, G. & Wang, J. (1989), Swarm intelligence in cellular robotic systems, *in* 'NATO Advanced Workshop on Robots and Biological Systems'.

Birattari, M., Yuan, Z., Balaprakash, P. & Stützle, T. (2010), F-race and iterated f-race: An overview, *in* T. Bartz-Beielstein, M. Chiarandini, L. Paquete & M. Preuss, eds, 'Experimental Methods for the Analysis of Optimization Algorithms', Springer.

Bremermann, H. J. (1962), Optimization through evolution and recombination, *in* M. C. Yovits, G. T. Jacobi & G. D. Golstine, eds, 'Proceedings of the Conference on Self-Organizing Systems – 1962', Spartan Books, Washington, DC, pp. 93–106.

Brent, R. P. (1973), *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, N.J.

Brest, J., Greiner, S., Bošković, B., Mernik, M. & Žumer, V. (2006), 'Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems', *IEEE Transactions on Evolutionary Computation* **10**(6), 646–657.

Brest, J., Korosec, P., Silc, J., Zamuda, A., Boskovic, B. & Maucec, M. S. (2013), 'Differential evolution and differential ant-stigmergy on dynamic optimisation problems', *Int. J. Systems Science* **44**(4), 663–679.

Brest, J. & Maucec, M. S. (2011), 'Self-adaptive differential evolution algorithm using population size reduction and three strategies', *Soft Compututing* **15**(11), 2157–2174.

Brest, J. & Maučec, M. S. (2008), 'Population size reduction for the differential evolution algorithm', *Applied Intelligence* **29**(3), 228–247.

Burke, E. K. & Bykov, Y. (2008), A late acceptance strategy in hill-climbing for exam timetabling problems, *in* 'PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling'.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E. & Woodward, J. (2010), Classification of hyper-heuristic approaches, *in* 'Handbook of Meta-Heuristics', Springer, pp. 449–468.

Burke, E. K., Kendall, G. & Soubeiga, E. (2003), 'A tabu search hyperheuristic for timetabling and rostering', *Journal of Heuristics* **9**(6), 451–470.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S. & Qu, R. (2007), 'A graph-based hyperheuristic for educational timetabling problems', *European Journal of Operational Research* **176**, 177–192.

Caponio, A., Cascella, G. L., Neri, F., Salvatore, N. & Sumner, M. (2007), 'A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives', *IEEE Transactions on System Man and Cybernetics-part B* **37**(1), 28–41.

Caponio, A., Kononova, A. & Neri, F. (2010), Differential evolution with scale factor local search for large scale problems, *in* Y. Tenne & C.-K. Goh, eds, 'Computational Intelligence in Expensive Optimization Problems', Vol. 2 of *Studies in Evolutionary Learning and Optimization*, Springer, chapter 12, pp. 297–323.

Caraffini, F., Iacca, G., Neri, F. & Mininno, E. (2012*a*), The importance of being structured: A comparative study on multi stage memetic approaches, *in* 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1–8.

Caraffini, F., Iacca, G., Neri, F. & Mininno, E. (2012*b*), Three variants of three stage optimal memetic exploration for handling non-separable fitness landscapes, *in* 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1–8.

Caraffini, F., Iacca, G., Neri, F., Picinali, L. & Mininno, E. (2013), A cma-es super-fit scheme for the re-sampled inheritance search, *in* 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1123–1130.

Caraffini, F., Neri, F., Cheng, J., Zhang, G., Picinali, L. & G. Iacca, E. M. (2013), Super-fit multicriteria adaptive differential evolution, *in* 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1678–1685.

Caraffini, F., Neri, F., Gongora, M. & Passow, B. (2013), Re-sampling search: A seriously simple memetic approach with a high performance, *in* 'IEEE Symposium Series on Computational Intelligence, Workshop on Memetic Computing', pp. 52–59.

Caraffini, F., Neri, F., Iacca, G. & Mol, A. (2012), 'Parallel memetic structures', *Information Sciences* **227**(0), 60–82.

Caraffini, F., Neri, F., Iacca, G. & Mol, A. (2013), 'Parallel memetic structures', *Information Sciences* **227**(0), 60 – 82.

Caraffini, F., Neri, F., Passow, B. N. & Iacca, G. (2013*a*), 'Re-sampled inheritance search: high performance despite the simplicity', *Soft Computing* pp. 1–22.

Caraffini, F., Neri, F., Passow, B. N. & Iacca, G. (2013*b*), 'Re-sampled inheritance search: High performance despite the simplicity', *Soft Computing* pp. 1–22.

Caraffini, F., Neri, F. & Picinali, L. (2014), 'An analysis on separability for memetic computing automatic design', *Information Sciences* **265**(0), 1 – 22.

Caraffini, F., Neri, F. & Poikolainen, I. (2013), Micro-differential evolution with extra moves along the axes, *in* 'IEEE Symposium Series on Computational Intelligence, Symposium on Differential Evolution', pp. 46–53.

Chabuk, T., Reggia, J., Lohn, J. & Linden, D. (2012), 'Causally-guided evolutionary optimization and its application to antenna array design', *Integrated Computer-Aided Engineering* **19**, 11–124.

Chen, W.-N., Zhang, J., Lin, Y., Chen, N., Zhan, Z.-H., Chung, H.-H., Li, Y. & Shi, Y.-H. (2013), 'Particle swarm optimization with an aging leader and challengers', *Evolutionary Computation, IEEE Transactions on* **17**(2), 241–258.

Chen, Y.-P. (2012), *Exploitation of Linkage Learning in Evolutionary Algorithms*, Vol. 3, Springer.

Clerc, M. & Kennedy, J. (2002), 'The particle swarm-explosion, stability and convergence in a multidimensional complex space', *IEEE Transactions on Evolutionary Computation* **6**(1), 58–73.

Cody, W. J. (1969), 'Rational chebyshev approximations for the error function', *Mathematics of Computation* **23**(107), 631–637.

Colorni, A., Dorigo, M., Maniezzo, V. et al. (1991), Distributed optimization by ant colonies, *in* 'Proceedings of the first European conference on artificial life', Vol. 142, Paris, France, pp. 134–142.

Cowling, P., Kendall, G. & Soubeiga, E. (2000), A hyperheuristic approach to scheduling: a sales summit, *in* 'Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling', Vol. 2079 of *Lecture Notes in Computer Science*, Springer, pp. 176–190.

Cox, D. R. & Hinkley, D. (1974), *Theoretical Statistics*, Chapman & Hall.

Cyber Dyne Srl Home Page (2012), 'Kimeme'. `http://cyberdynesoft.it/`.

Das, S. & Suganthan, P. (2011), 'Differential evolution: A survey of the state-of-the-art', *Evolutionary Computation, IEEE Transactions on* **15**(1), 4–31.

Dasgupta, S., Das, S., Biswas, A. & Abraham, A. (2009), 'On stability and convergence of the population-dynamics in differential evolution', *AI Communications - The European Journal on Artificial Intelligence* **22**(1), 1–20.

Davis, L. et al. (1991), *Handbook of genetic algorithms*, Vol. 115, Van Nostrand Reinhold New York.

Dawkins, R. (1976), *The Selfish Gene*, Press, Oxford University.

Dowsland, K. A., Soubeiga, E. & Burke, E. (2007), 'A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation', *European Journal of Operational Research* **179**(3), 759–774.

Eiben, A. E., Hinterding, R. & Michalewicz, Z. (1999), 'Parameter control in evolutionary algorithms', *Evolutionary Computation, IEEE Transactions on* **3**(2), 124–141.

Eiben, A. E., Michalewicz, Z., Schoenauer, M. & Smith, J. (2007), Parameter control in evolutionary algorithms, *in* F. Lobo, C. Lima & Z. Michalewicz, eds, 'Parameter Setting in Evolutionary Algorithms', Vol. 54 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, pp. 19–46.

Eiben, A. E. & Smith, J. E. (2007), *Introduction to evolutionary computing*, Vol. 2, Springer Berlin.

Epstein, S. L. & Petrovic, S. (2012), Learning a mixture of search heuristics, *in* Y. Hamadi, E. Monfroy & F. Saubion, eds, 'Autonomous Search', Springer, pp. 97–127.

Eshelman, L. J. & Schaffer, J. D. (1992), Real-coded genetic algorithms and interval-schemata, *in* L. Whitley, ed., 'Foundations of Genetic Algorithms II', Morgan Kaufmann, San Mateo CA, pp. 187–202.

Fernandes, C. & Rosa, A. (2001), A study on non-random mating and varying population size in genetic algorithms using a royal road function, *in* 'Evolutionary Computation, 2001. Proceedings of the 2001 Congress on', Vol. 1, IEEE, pp. 60–66.

Fister, I., Jr., I. F., Brest, J. & Zumer, V. (2012), Memetic artificial bee colony algorithm for large-scale global optimization, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 1–8.

Fogel, L. J., Owens, A. J. & Walsh, M. J. (1966), *Artificial Intelligence through Simulated Evolution*, John Wiley.

García-Martínez, C. & Lozano, M. (2008), Local search based on genetic algorithms, *in* 'Advances in metaheuristics for hard optimization', Springer, pp. 199–221.

García, S., Fernández, A., Luengo, J. & Herrera, F. (2008), 'A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability', *Soft Computing* **13**(10), 959–977.

García, S., Molina, D., Lozano, M. & Herrera, F. (2009), 'A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization', *Journal of Heuristics* **15**(6), 617–644.

Gautschi, W. (1972), Error function and fresnel integrals, *in* M. Abramowitz & I. A. Stegun, eds, 'Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables', chapter 7, pp. 297–309.

Geoffry E. Hinton and, S. J. N. (1987), 'How learning can guide evolution', *Complex systems* **1**(3), 495–502.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., Reading, MA, USA.

Gross, D. & Harris, C. M. (1985), *Fundamentals of Queueing Theory*, Wiley, NY.

Hamadi, Y., Jabbour, S. & Sais, J. (2012), Control-based clause sharing in parallel sat solving, *in* Y. Hamadi, E. Monfroy & F. Saubion, eds, 'Autonomous Search', Springer, pp. 245–267.

Hamadi, Y., Monfroy, E. & Saubion, F. (2012), *Autonomous Search*, Springer.

Hansen, N. (2012), 'The CMA evolution strategy'. http://www.lri.fr/ hansen/cmaesintro.html.

Hansen, N., Auger, A., Finck, S. & Ros, R. (2010), Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup, Research Report RR-7215, INRIA.

Hansen, N., Auger, A., Finck, S., Ros, R. et al. (2010), Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions, Technical Report RR-6829, INRIA.

Hansen, N., Müller, S. D. & Koumoutsakos, P. (2003), 'Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)', *Evolutionary Computation* **11**(1), 1–18.

Hansen, N. & Ostermeier, A. (1996), Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, *in* 'Proceedings of the IEEE International Conference on Evolutionary Computation', pp. 312–317.

Hansen, N. & Ostermeier, A. (2001), 'Completely derandomized self-adaptation in evolution strategies', *Evolutionary Computation* **9**(2), 159–195.

Hansen, P. & Mladenović, N. (2001), 'Variable neighborhood search: Principles and applications', *European journal of operational research* **130**(3), 449–467.

Hansen, P., Mladenovi, N. & Moreno Prez, J. (2008), 'Variable neighbourhood search: methods and applications', *4OR* **6**(4), 319–360.

Harik, G. R., Lobo, F. G. & Goldberg, D. E. (1999), 'The compact genetic algorithm', *IEEE Transactions on Evolutionary Computation* **3**(4), 287–297.

Harik, G. R., Lobo, F. G. & Sastry, K. (2006), Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA), *in* M. Pelikan, K. Sastry & E. Cantú-Paz, eds, 'Scalable Optimization via Probabilistic Modeling', Vol. 33 of *Studies in Computational intelligence*, Springer, pp. 39–61.

Hart, W. E., Krasnogor, N. & Smith, J. E. (2004), Memetic evolutionary algorithms, *in* W. E. Hart, N. Krasnogor & J. E. Smith, eds, 'Recent Advances in Memetic Algorithms', Springer, Berlin, Germany, pp. 3–27.

Hart, W. E., Krasnogor, N. & Smith, J. E. (2005), *Recent advances in memetic algorithms*, Vol. 166, Springer Verlag.

Hauke, J. & Kossowski, T. (2011), 'Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data', *Quaestiones Geographicae* (2), 87–93.

Herrera, F., Lozano, M. & Sánchez, A. M. (2003), 'A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study', *International Journal of Intelligent Systems* **18**, 309–338.

Holland, J. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Holm, S. (1979), 'A simple sequentially rejective multiple test procedure', *Scandinavian Journal of Statistics* **6**(2), 65–70.

Hooke, R. & Jeeves, T. A. (1961), 'Direct search solution of numerical and statistical problems', *Journal of the ACM* **8**, 212–229.

Hoos, H. H. (2012), 'Programming by optimization', *Communication of the ACM* **55**(22), 70–80.

Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2010), 'Tradeoffs in the empirical evaluation of competing algorithm designs', *Annals of Mathematics and Artificial Intelligence* **60**(1-2), 65–89.

Hutter, F., Xu, L., Hoos, H. H. & Leyton-Brown, K. (2014), 'Algorithm runtime prediction: Methods & evaluation', *Artificial Intelligence* **206**, 79–111.

Iacca, G., Caraffini, F. & Neri, F. (2012), 'Compact differential evolution light: high performance despite limited memory requirement and modest computational overhead', *Journal of Computer Science and Technology* **27**(5), 1056–1076.

Iacca, G., Caraffini, F. & Neri, F. (2013), 'Memory-saving memetic computing for path-following mobile robots', *Applied Soft Computing* **13**(4), 2003–2016.

Iacca, G., Caraffini, F. & Neri, F. (2014), 'Multy-strategy coevolving aging particle optimization', *International Journal of Neural Systems* **24**(01), 1450008.

Iacca, G., Caraffini, F., Neri, F. & Mininno, E. (2012), Robot base disturbance optimization with compact differential evolution light, *in* 'EvoApplications', pp. 285–294.

Iacca, G., Caraffini, F., Neri, F. & Mininno, E. (2013), Single particle algorithms for continuous optimization, *in* 'Evolutionary Computation (CEC), 2013 IEEE Congress on', pp. 1610–1617.

Iacca, G., Mallipeddi, R., Mininno, E., Neri, F. & Suganthan, P. N. (2011), Super-fit and population size reduction mechanisms in compact differential evolution, *in* 'Proceedings of IEEE Symposium on Memetic Computing', pp. 21–28.

Iacca, G., Neri, F., Caraffini, F., & Suganthan, P. N. (2014), A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms, *in* 'EvoApplications', p. To appear.

Iacca, G., Neri, F. & Mininno, E. (2012), Compact bacterial foraging optimization, *in* 'Proceedings of the 2012 International Conference on Swarm and Evolutionary Computation', Springer-Verlag, pp. 84–92.

Iacca, G., Neri, F., Mininno, E., Ong, Y.-S. & Lim, M.-H. (2012), 'Ockhams razor in memetic computing: three stage optimal memetic exploration', *Information Sciences* **188**, 17–43.

Igel, C., Suttorp, T. & Hansen, N. (2006), A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', ACM Press, pp. 453–460.

Islam, S., Das, S., Ghosh, S., Roy, S. & Suganthan, P. (2012), 'An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization', *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **42**(2), 482–500.

Kendall, G., Soubeiga, E. & Cowling, P. (2002), Choice function and random hyperheuristics, *in* 'Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, SEAL', Springer, pp. 667–671.

Kennedy, J. & Eberhart, R. C. (1995), Particle swarm optimization, *in* 'Proceedings of IEEE International Conference on Neural Networks', pp. 1942–1948.

Kirkpatrick, S., Gelatt, C. D. J. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *Science* (220), 671–680.

Kononova, A. V., Hughes, K. J., Pourkashanian, M. & Ingham, D. B. (2007), Fitness diversity based adaptive memetic algorithm for solving inverse problems of chemical kinetics, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 2366–2373.

Korošec, P. & Šilc, J. (2008), The differential ant-stigmergy algorithm for large scale real-parameter optimization, *in* 'ANTS '08: Proceedings of the 6th international conference on Ant Colony Optimization and Swarm Intelligence', Lecture Notes in Computer Science, Springer, pp. 413–414.

Koza, J. R. (1992), *Genetic Programming: vol. 1, On the programming of computers by means of natural selection*, Vol. 1, MIT press.

Krasnogor, N. (2004), Toward robust memetic algorithms, *in* W. E. Hart, N. Krasnogor & J. E. Smith, eds, 'Recent Advances in Memetic Algorithms', Studies in Fuzzines and Soft Computing, Springer, Berlin, Germany, pp. 185–207.

Krasnogor, N. & Smith, J. (2005), 'A tutorial for competent memetic algorithms: model, taxonomy, and design issues', *IEEE Transactions on Evolutionary Computation* **9**, 474–488.

Lagarias, J. C., Reeds, J. A., Wright, M. H. & Wright, P. E. (1998), 'Convergence properties of the nelder-mead simplex method in low dimensions', *SIAM Journal on Optimization* **9**, 112–147.

Li, X. & Yao, X. (2012), 'Cooperatively coevolving particle swarms for large scale optimization', *Evolutionary Computation, IEEE Transactions on* **16**(2), 210–224.

Liang, J. J., Qin, A. K., Suganthan, P. N. & Baskar, S. (2006), 'Comprehensive learning particle swarm optimizer for global optimization of multimodal functions', *IEEE Transactions on Evolutionary Computation* **10**(3), 281–295.

Liang, J. J., Qu, B. Y., Suganthan, P. N. & Hernndez-Daz, A. G. (2013), Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization, Technical Report 201212, Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore.

Lin, S.-F. & Cheng, Y.-C. (2011), A separability detection approach to cooperative particle swarm optimization, pp. 1141–1145.

Liu, J. & Lampinen, J. (2005), 'A fuzzy adaptive differential evolution algorithm', *Soft Computing - A Fusion of Foundations, Methodologies and Applications* **9**(6), 448–462.

Lopez-Ibanez, M., Dubois-Lacoste, J., Stützle, T. & Birattari, M. (2011), The irace package, iterated race for automatic algorithm configuration, Technical Report TR/IRIDIA/2011-004, IRIDIA, Universite Libre de Bruxelles, Belgium.

Loshchilov, I., Schoenauer, M. & Sebag, M. (2012), Alternative restart strategies for cma-es, *in* C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia & M. Pavone, eds, 'Parallel Problem Solving from Nature - PPSN XII', Vol. 7491 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 296–305.

Lozano, M., Molina, D. & Herrera, F. (2011), 'Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems', *Soft Computing* **15**(11), 2085–2087.

Luke, S. & Spector, L. (1997), 'A comparison of crossover and mutation in genetic programming', *Genetic Programming* **97**, 240–248.

Mallipeddi, R., Mallipeddi, S. & Suganthan, P. N. (2010), 'Ensemble strategies with adaptive evolutionary programming', *Information Sciences* **180**(9), 1571–1581.

Mann, H. B., Whitney, D. R. et al. (1947), 'On a test of whether one of two random variables is stochastically larger than the other', *The annals of mathematical statistics* **18**(1), 50–60.

Marchiori, E. & Steenbeek, A. (2000), An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling, *in* 'Scheduling, in Real World Applications of Evolutionary Computing', Lecture Notes in Computer Science, Springer, pp. 367–381.

Maturana, J., Fialho, A., Saubion, F., Schoenauer, M., Lardeux, F. & Sebag, M. (2012), Adaptive operator selection and management in evolutionary algorithms, *in* Y. Hamadi, E. Monfroy & F. Saubion, eds, 'Autonomous Search', Springer, pp. 161–189.

Meuth, R., M. Lim, H., Ong, Y. S. & Wunsch-II, D. C. (2009), 'A proposition on memes and meta-memes in computing for higher-order learning', *Memetic Computing Journal* **1**(2), 85–100.

Mininno, E., Cupertino, F. & Naso, D. (2008*a*), 'Real-valued compact genetic algorithms for embedded microcontroller optimization', *IEEE Transactions on Evolutionary Computation* **12**(2), 203–219.

Mininno, E., Cupertino, F. & Naso, D. (2008*b*), 'Real-valued compact genetic algorithms for embedded microcontroller optimization', *Evolutionary Computation, IEEE Transactions on* **12**(2), 203–219.

Mininno, E., Neri, F., Cupertino, F. & Naso, D. (2011), 'Compact differential evolution', *IEEE Transactions on Evolutionary Computation* **15**(1), 32–54.

Molina, D., Lozano, M., García-Martínez, C. & Herrera, F. (2010*a*), 'Memetic algorithms for continuous optimisation based on local search chains', *Evolutionary Computing* **18**(1), 27–63.

Molina, D., Lozano, M., García-Martínez, C. & Herrera, F. (2010*b*), 'Memetic algorithms for continuous optimization based on local search chains', *Evolutionary Computation* **18**(1), 27–63.

Molina, D., Lozano, M. & Herrera, F. (2010), Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization, *in* 'IEEE Congress on Evolutionary Computation', pp. 1–8.

Montes de Oca, M., Stutzle, T., Birattari, M. & Dorigo, M. (2009), 'Frankenstein's pso: A composite particle swarm optimization algorithm', *IEEE Transactions on Evolutionary Computation* **13**(5), 1120–1132.

Moscato, P. (1989), On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Technical Report 826.

Moscato, P. & Norman, M. (1989), A competitive and cooperative approach to complex combinatorial search, Technical Report 790.

Mühlenbein, H. & Schlierkamp-Voosen, D. (1993), 'Predictive models for the breeder genetic algorithm in continuous parameter optimization', *Evolutionary computation* **1**(1), 25–49.

Nannen, V., Smit, S. K. & Eiben, A. E. (2008), Costs and benefits of tuning parameters of evolutionary algorithms, *in* G. Rudolph, T. Jansen, S. Lucas, C. Poloni & N. Beume, eds, 'Parallel Problem Solving from Nature PPSN X', Vol. 5199 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 528–538.

Nelder, A. & Mead, R. (1965), 'A simplex method for function optimization', *Computation Journal* **Vol 7**, 308–313.

Neri, F. & Cotta, C. (2012), 'Memetic algorithms and memetic computing optimization: A literature review', *Swarm and Evolutionary Computation* **2**, 1–14.

Neri, F., Cotta, C. & Moscato, P. (2011), *Handbook of Memetic Algorithms*, Vol. 379 of *Studies in Computational Intelligence*, Springer.

Neri, F., Mininno, E. & Iacca, G. (2013), 'Compact particle swarm optimization', *Information Sciences* **239**(0), 96 – 121.

Neri, F. & Tirronen, V. (2008), On memetic differential evolution frameworks: a study of advantages and limitations in hybridization, *in* 'Proceedings of the IEEE World Congress on Computational Intelligence', pp. 2135–2142.

Neri, F. & Tirronen, V. (2010), 'Recent advances in differential evolution: A review and experimental analysis', *Artificial Intelligence Review* **33**(1–2), 61–106.

Neri, F., Toivanen, J., Cascella, G. L. & Ong, Y. S. (2007), 'An adaptive multimeme algorithm for designing HIV multidrug therapies', *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **4**(2), 264–278.

Neri, F., Weber, M., Caraffini, F. & Poikolainen, I. (2012), Meta-lamarckian learning in three stage optimal memetic exploration, *in* 'Computational Intelligence (UKCI), 2012 12th UK Workshop on', pp. 1–8.

Nguyen, T. T. & Yao, X. (2008), 'An experimental study of hybridizing cultural algorithms and local search', *International Journal of Neural Systems* **18**(1), 1–17.

Nikolaus, H. (2005), 'The cma evolution strategy: A tutorial', *Vu le* **29**.

Nikolaus, H. & Stefan, K. (2004), Evaluating the cma evolution strategy on multimodal test functions, *in* 'Parallel Problem Solving from Nature-PPSN VIII', Springer, pp. 282–291.

Olorunda, O. & Engelbrecht, A. (2007), Differential evolution in high-dimensional search spaces, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 1934–1941.

Ong, Y. S. & Keane, A. J. (2004), 'Meta-lamarkian learning in memetic algorithms', *IEEE Transactions on Evolutionary Computation* **8**(2), 99–110.

Ong, Y.-S., Lim, M.-H. & Chen, X. (2009), Research frontier: Towards memetic computing, Technical Report C2i-1209, School of Computer Engineering, Nanyang Technological University, Singapore.

Ong, Y.-S., Lim, M.-H. & Chen, X. (2010), 'Memetic computation-past, present and future', *IEEE Computational Intelligence Magazine* **5**(2), 24–31.

Ong, Y. S., Lim, M. H., Zhu, N. & Wong, K. W. (2006), 'Classification of adaptive memetic algorithms: A comparative study', *IEEE Transactions On Systems, Man and Cybernetics - Part B* **36**(1), 141–152.

Özcan, E., Bilgin, B. & Korkmaz, E. E. (2008), 'A comprehensive analysis of hyper-heuristics', *Intelligent Data Analysis* **12**(1), 3–23.

Palmer, J. R. (1969), 'An improved procedure for orthogonalising the search vectors in rosenbrock's and swann's direct search optimisation methods', *The Computer Journal* **12**(1), 69–71.

Parsopoulos, K. E. (2009), Cooperative micro-differential evolution for high-dimensional problems, *in* 'Proceedings of the conference on Genetic and evolutionary computation', pp. 531–538.

Passino, K. M. (2002), 'Biomimicry of bacterial foraging for distributed optimization and control', *Control Systems, IEEE* **22**(3), 52–67.

Pearson, K. (1903), 'Mathematical contributions to the theory of evolution. XI. on the influence of natural selection on the variability and correlation of organs', *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* **200**, 1–66.

Pelikan, M., Goldberg, D. & Lobo, F. (2000), A survey of optimization by building and using probabilistic models, *in* 'American Control Conference, 2000. Proceedings of the 2000', Vol. 5, pp. 3289–3293.

Peng, F., Tang, K., Chen, G. & Yao, X. (2010), 'Population-based algorithm portfolios for numerical optimization', *IEEE Transactions on Evolutionary Computation* **14**(5), 782–800.

Pérez, J. A. M., Hansen, P. & Mladenovic, N. (2004), *Parallel variable neighborhood search*, Groupe d'études et de recherche en analyse des décisions.

Poikolainen, I., Caraffini, F., Neri, F. & Weber, M. (2012), Handling non-separability in three stage memetic exploration, *in* 'Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications', pp. 195–205.

Poikolainen, I., Iacca, G., Caraffini, F. & Neri, F. (2013), 'Focusing the search: a progressively shrinking memetic computing framework', *International Journal of Innovative Computing and Applications* **5**(3), 127–142.

Poli, R. (2008), 'Analysis of the publications on the applications of particle swarm optimisation', *Journal of Artificial Evolution and Applications* **2008**, 4:1–4:10.

Poli, R. & Graff, M. (2009), There is a free lunch for hyper-heuristics, genetic programming and computer scientists, *in* L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco & M. Ebner, eds, 'Genetic Programming', Vol. 5481 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 195–207.

Potter, M. A. & De Jong, K. (2000), 'Cooperative coevolution: An architecture for evolving coadapted subcomponents', *Evolutionary Computation* **8**(1), 1–29.

Potter, M. A. & De Jong, K. A. (1994), A cooperative coevolutionary approach to function optimization, *in* 'Proceedings of the Third Conference on Parallel Problem Solving from Nature', Springer-Verlag, pp. 249–257.

Powell, M. J. D. (1964), 'An efficient method for finding the minimum of a function of several variables without calculating derivatives', *The Computer Journal* **7**(2), 155–162.

Press, W. H. (2007), *Numerical recipes 3rd edition: The art of scientific computing*, Cambridge university press.

Price, K. V., Storn, R. & Lampinen, J. (2005), *Differential Evolution: A Practical Approach to Global Optimization*, Springer.

Prügel-Bennett, A. (2010), 'Benefits of a population: Five mechanisms that advantage population-based algorithms', *IEEE Transactions on Evolutionary Computation* **14**(4), 500–517.

Qin, A. K., Huang, V. L. & Suganthan, P. N. (2009), 'Differential evolution algorithm with strategy adaptation for global numerical optimization', *IEEE Transactions on Evolutionary Computation* **13**, 398–417.

Qin, A. K. & Suganthan, P. N. (2005), Self-adaptive differential evolution algorithm for numerical optimization, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', Vol. 2, pp. 1785–1791.

Rahnamayan, S. & Tizhoosh, H. R. (2008), Image thresholding using micro opposition-based differential evolution (micro-ode), *in* 'Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on', pp. 1409–1416.

Rainer, S. & Kenneth, P. (1997), 'Differential evolution a simple and efficient heuristic for global optimization over continuous spaces', *Journal of Global Optimization* **11**(4), 341–359.

Rechenberg, I. (1971), Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, PhD thesis, Technical University of Berlin.

Ren, Z., Jiang, H., Xuan, J. & Luo, Z. (2012), 'Hyper-heuristics with low level parameter adaptation', *Evolutionary Computation* **20**(2), 189–227.

Rinaldi, F. (2012), 'A class of derivative-free nonmonotone algorithms for unconstrained optimization', *Seminario Dottorato 2012/13* p. 107.

Ros, R. & Hansen, N. (2008), A simple modification in cma-es achieving linear time and space complexity, *in* 'Proceesdings of the Parallel Problem Solving in Nature', pp. 296–305.

Rosenbrock, H. H. (1960), 'An automatic method for finding the greatest or least value of a function', *The Computer Journal* **3**(3), 175–184.

S. Bhatnagar, H. L. P. & Prashanth., L. A. (2003), *Introduction to Stochastic Search and Optimization*, 1 edn, John Wiley & Sons, Inc., New York, NY, USA.

Salvatore, N., Caponio, A., Neri, F., Stasi, S. & Cascella, G. L. (2010), 'Optimization of delayed-state kalman-filter-based algorithm via differential evolution for sensorless control of induction motors', *Industrial Electronics, IEEE Transactions on* **57**(1), 385–394.

Schwefel, H. (1981), *Numerical Optimization of Computer Models*, Wiley, Chichester, England, UK.

Schwefel, H.-P. (1965), Kybernetische Evolutioals Strategie der experimentellen Forschung in der Strömungstechnik, PhD thesis, Technical University of Berlin, Hermann Föttinger–Institute for Fluid Dynamics.

Shi, Y.-J., Teng, H.-F. & Li, Z.-Q. (2005), Cooperative co-evolutionary differential evolution for function optimization, *in* 'Advances in Natural Computation', Vol. 3611 of *Lecture Notes in Computer Science*, Springer, pp. 1080–1088.

Smith, J. E. (2007), 'Coevolving memetic algorithms: A review and progress report', *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **37**(1), 6–17.

Smith, J. E. (2012), 'Estimating meme fitness in adaptive memetic algorithms for combinatorial problems', *Evolutionary Computation* **20**(2), 165–188.

Sofge, D., De Jong, K. & Schultz, A. (2002), A blended population approach to cooperative coevolution for decomposition of complex problems, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 413–418.

Solis, F. J. & Wets, R. J.-B. (1981), 'Minimization by random search techniques', *Mathematics of Operations Research* **6**(1), 19–30.

Spall, J. (1987), A stochastic approximation technique for generating maximum likelihood parameter estimates, *in* 'American Control Conference, 1987', pp. 1161–1167.

Spall, J. C. (1992), 'Multivariate stochastic approximation using a simultaneous perturbation gradient approximation', *IEEE Transactions on Automatic Control* **372**, 332–341.

Spall, J. C. (1998), 'Implementation of the simultaneous perturbation algorithm for stochastic optimization', *Aerospace and Electronic Systems, IEEE Transactions on* **34**(3), 817–823.

Spearman, C. (1904), 'The proof and measurement of association between two things', *The American Journal of Psychology* **15**(1), 72–101.

Spendley, W., Hext, G. R. & Himsworth, F. R. (1962), 'Sequential application of simplex designs in optimisation and evolutionary operation', *Technometrics* **4**(4), 441–461.

Storn, R. (1996), Differential evolution design of an IIR-filter, *in* 'Proceedings of IEEE International Conference on Evolutionary Computation', pp. 268–273.

Storn, R. (1999), 'System design by constraint adaptation and differential evolution', *IEEE Transactions on Evolutionary Computation* **3**(1), 22–34.

Storn, R. (2005), 'Designing nonstandard filters with differential evolution', *IEEE Signal Processing Magazine* **22**(1), 103–106.

Storn, R. & Price, K. (1995), Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, ICSI.

Sudholt, D. (2009), 'The impact of parametrization in memetic evolutionary algorithms', *Theoretical Computer Science* **410**(26), 2511–2528.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A. & Tiwari, S. (2005), Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Technical Report 2005005, Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India.

Swagatam, D. & Suganthan, P. (2010), 'Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems', *Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India* .

Takahama, T. & Sakai, S. (2010), Solving nonlinear optimization problems by differential evolution with a rotation-invariant crossover operation using gram-schmidt process, *in* 'Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on', pp. 526–533.

Tan, Y. & Zhou, Y. (2011), Parallel particle swarm optimization algorithm based on graphic processing units, *in* 'Handbook of Swarm Intelligence'.

Tang, K., Yao, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M. & Yang, Z. (2007), Benchmark functions for the CEC 2008 special session and competition on large scale global optimization, Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China.

Tao, H., Zain, J. M., Ahmed, M. M., Abdalla, A. N. & Jing, W. (2012), 'A wavelet-based particle swarm optimization algorithm for digital image watermarking', *Integrated Computer-Aided Engineering* **19**(1), 81–91.

Thierens, D. (2004), *in* 2, ed., 'Genetic and Evolutionary Computation GECCO 2004', Vol. 3103 of *Lecture Notes in Computer Science*.

Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K. & Rossi, T. (2008), 'An enhanced memetic differential evolution in filter design for defect detection in paper production', *Evolutionary Computation* **16**, 529–555.

Trosset, M. W. (1997), 'I know it when i see it: toward a definition of direct search methods', *SIAG/OPT Views and News* **9**, 7–10.

Tseng, L.-Y. & Chen, C. (2008), Multiple trajectory search for large scale global optimization, *in* 'Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on', pp. 3052–3059.

Van den Bergh, F. & Engelbrecht, A. P. (2004), 'A cooperative approach to particle swarm optimization', *IEEE Transactions on Evolutionary Computation* **8**(3), 225–239.

Vrugt, J. A., Robinson, B. A. & Hyman, J. M. (2009), 'Self-adaptive multimethod search for global optimization in real-parameter spaces', *IEEE Transactions on Evolutionary Computation* **13**(2), 243–259.

Wang, H., Moon, I., Yang, S. & Wang, D. (2012), 'A memetic particle swarm optimization algorithm for multimodal optimization problems', *Information Sciences* **197**(0), 38 – 52.

Whitley, D., Gordon, V. & Mathias, K. (1994), Lamarckian evolution, the baldwin effect and function optimization, *in* Y. Davidor, H.-P. Schwefel & R. Mnner, eds, 'Parallel Problem Solving from Nature PPSN II', Vol. 866 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 5–15.

Wilcoxon, F. (1945), 'Individual comparisons by ranking methods', *Biometrics Bulletin* **1**(6), 80–83.

Wolpert, D. & Macready, W. (1997), 'No free lunch theorems for optimization', *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Wu, Y., McCall, J. A. W., Corne, D. & Regnier-Coudert, O. (2012), Landscape analysis for hyperheuristic bayesian network structure learning on unseen problems, *in* 'IEEE Congress on Evolutionary Computation', pp. 1–8.

Xinchao, Z. (2011), 'Simulated annealing algorithm with adaptive neighborhood', *Applied Soft Compututing* **11**, 1827–1836.

Xu, L., Hutter, F., Hoos, H. & Leyton-Brown, K. (2008), 'SATzilla: Portfolio-based algorithm selection for SAT', *Journal of Artificial Intelligence Research* **32**, 565–606.

Yang, Z., Tang, K. & Yao, X. (2007), Differential evolution for high-dimensional function optimization, *in* 'Proceedings of the IEEE Congress on Evolutionary Computation', pp. 3523–3530.

Yang, Z., Tang, K. & Yao, X. (2008), 'Large scale evolutionary optimization using cooperative coevolution', *Information Sciences* **178**(15), 2985–2999.

Yuhui, S. & Eberhart, R. C. (1998), Parameter selection in particle swarm optimization, *in* V. Porto, N. Saravanan, D. Waagen & A. E. Eiben, eds, 'Evolutionary Programming VII', Vol. 1447 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 591–600.

Zamuda, A., Brest, J., Bošković, B. & Žumer, V. (2008*a*), High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, *in* 'Proceedings of the IEEE World Congress on Computational Intelligence', pp. 2032–2039.

Zamuda, A., Brest, J., Bošković, B. & Žumer, V. (2008*b*), Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution, *in* 'Proceedings of the IEEE World Congress on Computational Intelligence', pp. 3719–3726.

Zhang, J. & Sanderson, A. C. (2009), 'Jade: Adaptive differential evolution with optional external archive', *IEEE Transactions on Evolutionary Computation* **13**(5), 945–958.

Zhen, J., Jiarui, Z., Huilian, L. & Qing-Hua, W. (2010), 'A novel intelligent single particle optimizer', *Chinese Journal of computers, vol33* (3), 556–561.

Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y. & Qian, J.-X. (2003), Empirical study of particle swarm optimizer with an increasing inertia weight, *in* 'Proc. IEEE Congr. Evol. Comput', pp. 221–226.

Zhu, Z., Jia, S. & Ji, Z. (2010), 'Towards a memetic feature selection paradigm', *IEEE Computational Intelligence Magazine* **5**(2), 41–53.

Zhu, Z., Zhou, J., Ji, Z. & Shi, Y.-H. (2011), 'DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm', *IEEE Transactions on Evolutionary Computation* **15**(5), 643–558.