

University of Louisville

ThinkIR: The University of Louisville's Institutional Repository


Electronic Theses and Dissertations

5-2018

Multi self-adapting particle swarm optimization algorithm (MSAPSO).

Gerhard Koch
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Computational Engineering Commons](#), [Dynamical Systems Commons](#), [Industrial Engineering Commons](#), [Statistics and Probability Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Koch, Gerhard, "Multi self-adapting particle swarm optimization algorithm (MSAPSO)." (2018). *Electronic Theses and Dissertations*. Paper 3000.
<https://doi.org/10.18297/etd/3000>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

A MULTI SELF-ADAPTING PARTICLE SWARM OPTIMIZATION ALGORITHM
(MSAPSO)

By

Gerhard Koch

Diplom-Ingenieur (FH), Technische Informatik, Albstadt-Ebingen, 1993
Diplom-Wirtschaftsingenieur (FH), Fernhochschule Hamburg, 2006

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
In fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Industrial Engineering

Department of Industrial Engineering
Speed School of Engineering
University of Louisville
Louisville, Kentucky

May 2018

Copyright 2017 by Gerhard Koch

All rights preserved

A MULTI SELF-ADAPTING PARTICLE SWARM OPTIMIZATION ALGORITHM
(MSAPSO)

By

Gerhard Koch

Diplom-Ingenieur (FH), Technische Informatik, Albstadt-Ebingen, 1993
Diplom-Wirtschaftsingenieur (FH), Fernhochschule Hamburg, 2006

A Dissertation Proposal approved on

December 18th, 2017

by the following Dissertation Committee:

Dissertation Director, Dr. Gail W. Depuy, I.E.

Dr. Suraj Alexander, I.E.

Dr. John Usher, I.E.

Dr. Roman V. Yampolskiy, CECS

DEDICATION

The dissertation dedicates to my wonderful life partner.

Since March 2014, when I started writing my PhD thesis she always believed in what I was doing during my intense studies around my PhD thesis. I want to say thank you for her great patience, understanding and commitment during this time.

In addition, I would like to thank you my family, who finally put me in a position to reach the goal of achieving academic excellence expressed by this PhD thesis.

I am also very much obliged to my actual employer Daimler AG, especially my team, who released me one day per week from my regular work to finalize this PhD thesis.

ACKNOWLEDGEMENTS

The structure and content of the doctoral thesis was strongly supported by my dissertation advisor Prof. Dr. Gail Depuy. She supported and advised me a lot on the organizational aspects as well as the overall consistency of the PhD thesis. Prof. Dr. Thomas Riedel from the mathematics department of the University of Louisville educated me on various aspects of the MSAPSO convergence analysis relevant and essential for the presented PhD work.

To the other members of my dissertation committee I would like to express my most profound thanks: Dr. John Usher and Dr. Yampolskiy, who have generously supported me with their mathematical and computer science expertise. This knowhow further improved the quality of this work. I really appreciated their continuous contribution in their field of expertise. Also, it is a great pleasure for me that Dr. Suraj Alexander as an experienced reviewer is member of the professors PhD panel.

My special thanks and appreciation to Prof. Dr. Joachim Gerlach from the University of Applied Science in Albstadt-Ebingen, (Germany) for his very good support on the algorithmic modelling part of the MSAPSO. He was a student colleague when I studied computer science during the 90's and now he acts as a professor for technical computer science at the same university. To all supporters above, I want to express my deepest appreciation.

ABSTRACT

A MULTI SELF-ADAPTING PARTICLE SWARM OPTIMIZATION ALGORITHM (MSAPSO)

Gerhard Koch
December 18, 2017

The performance and stability of the Particle Swarm Optimization algorithm depends on parameters that are typically tuned manually or adapted based on knowledge from empirical parameter studies. Such parameter selection is ineffectual when faced with a broad range of problem types, which often hinders the adoption of PSO to real world problems.

This dissertation develops a dynamic self-optimization approach for the respective parameters (inertia weight, social and cognition). The effects of self-adaption for the optimal balance between superior performance (convergence) and the robustness (divergence) of the algorithm with regard to both simple and complex benchmark functions is investigated. This work creates a swarm variant which is parameter-less, which means that it is virtually independent of the underlying examined problem type. As PSO variants always have the issue, that they can be stuck-in-local-optima, as second main topic the MSAPSO algorithm do have a highly flexible escape-lmin-strategy embedded, which works dimension-less.

The MSAPSO algorithm outperforms other PSO variants and also other swarm inspired approaches such as Memetic Firefly algorithm with these two major algorithmic elements (parameter-less approach, dimension-less escape-lmin-strategy).

The average performance increase in two dimensions is at least fifteen percent with regard to the compared swarm variants. In higher dimensions (≥ 250) the performance gain accumulates to about fifty percent in average. At the same time the error-proneness of MSAPSO is in average similar or even significant better when converging to the respective global optima's.

TABLE OF CONTENTS

DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
LIST OF FIGURES	xii
LIST OF EQUATIONS	xviii
1 INTRODUCTION.....	1
1.1 Background PSO.....	2
1.2 Present state of research.....	8
1.3 Problem statement.....	15
1.4 Research contribution.....	16
1.5 Scope of PhD work	21

2	LITERATURE REVIEW.....	23
2.1	Traditional research branches PSO	25
2.1.1	Base Model - Standard Particle Swarm Algorithm (SPSO)	25
2.1.2	Parameter optimized PSO's	30
2.1.3	Information optimized PSO's	39
2.1.4	Operator optimized PSO's	43
2.2	Self-Optimization as new PSO research branch	49
2.2.1	Principles of Self-Organizing Systems	49
2.2.2	Self-Optimizing PSO's	52
2.2.3	Self-Optimization in alternate bio-inspired algorithm's	70
2.2.4	Dynamics in Social Systems - Bandura's Learning Theory	77
2.3	Summary of Findings	78
3	CONVERGENCE FRAMEWORK OF THE MSAPSO	80
3.1	Self-Adaptive behavior of MSAPSO	81
3.2	General convergence analysis PSO and considerations	82
3.3	Assumptions for convergence analysis	83

3.4	Order-1 convergence analysis – Difference & Matrix Model.....	85
3.4.1	Final order-1 convergence room of MSAPSO.....	92
3.4.2	Visualization of order-1 convergence.....	93
3.4.3	Harmonic oscillation curve order-1 convergence.....	94
3.5	Order-2 convergence Analysis based on Martinez approach.....	97
3.5.1	Visualization of the order-2 convergence system.....	112
3.5.2	Harmonic oscillation curve order-2 convergence.....	113
3.5.3	Mathematical proof of order-1 order-2 convergence zone collapse.....	115
3.5.4	Application of convergence study to MSAPSO.....	116
3.5.5	Final order-2 convergence room of MSAPSO.....	123
3.6	MSAPSO Stability Line for a universal stable convergence.....	123
3.6.1	Regression analysis of the stability property of MSAPSO.....	126
3.6.2	Mathematical proof of the MSAPSO stability curve.....	127
4	DESIGN OF MSAPSO ALGORITHM.....	129
4.1	Self-Adaptiveness of MSAPSO.....	130
4.2	MSAPSO Stability Line Formula.....	131
4.3	Variation options of MSAPSO Stability Line.....	132

4.3.1	Variation with success-based inertia-weight strategy.....	137
4.4	Reasoning of chosen MSAPSO probability distributions.....	138
4.5	Start values of MSAPSO.....	140
4.5.1	Reasoning of MSAPSO start inertia weight	140
4.5.2	Swarm size in different dimensions.....	141
4.6	Success definition of the MSAPSO	142
4.6.1	Swarm success per iteration.....	142
4.6.2	Swarm success in consecutive iterations	147
4.7	Escape Local Minima Strategy	148
4.7.1	Definition of stuck-into-local-optima	149
4.7.2	Concept of the Hyper-Middle-Point in the search space	149
4.7.3	Gathering information about Gbest-position without vector calculation .	152
4.7.4	Detecting swarm radius to size the local search hyper cube.....	153
4.7.5	Detecting the global search hyper cubes.....	154
4.7.6	HMP centric search cube	156
4.7.7	Final escape strategy for 2D to N-dimensions.....	157
4.8	Dynamic inertia weight strategy	157

4.8.1	Search Space characterization unimodal and multimodal	158
4.8.2	Sliding concept around optimal inertia weight	159
5	EVALUATION MODEL & RESULTS	160
5.1	Performance view of changes in convergence and stability zone	161
5.1.1	Performance influence of inertia weight variation.....	162
5.1.2	Performance influence of probability distributions variation	165
5.1.3	Performance and stability influence of variation of social and cognition	168
5.2	Testing method of MSAPSO against benchmark functions.....	171
5.2.1	Start and convergence settings for MSAPSO algorithm.....	174
5.2.2	Description of test method	175
5.2.3	List of benchmarks and their characteristics.....	176
5.3	Test results MSAPSO versus comparative algorithms	177
5.3.1	Short description of comparative algorithms	177
5.3.2	Preconditions for the long run test	182
5.3.3	Benchmarks and Minimum Tests	184
5.3.4	Benchmarks and Maximum Tests.....	194
5.4	Comparison MSAPSO & MFFA with and without ELM.....	204

5.4.1	Sphere Function without ELM.....	204
5.4.2	Sphere Function with ELM.....	205
5.4.3	Referenced Function comparison MSAPSO versus MFFA	206
5.5	Summary Evaluations Results.....	208
6	RESEARCH CONTRIBUTION OF MSAPSO.....	211
7	FUTURE RESEARCH	214
	REFERENCES	216
	LIST OF ABBREVIATIONS	223
	APPENDIX A – MINIMUM BENCHMARK FUNCTIONS	225
	APPENDIX B – MSAPSO STRUCTURAL DIAGRAM	233
	CURRICULUM VITAE	235

LIST OF FIGURES

Figure 1: Categorization of Natural Computation	8
Figure 2: Proposed Classes of PSO Research Branches.....	12
Figure 3: Ideal convergence behavior of a particle with self-optimization	20
Figure 4: Visualized PSO velocity and position update of particle i	28
Figure 5: CLPSO – dimension surfing based velocity updating of a particle	46
Figure 6: PSO optimal particle’s neighborhood structure	54
Figure 7: MSAPSO order-1 (μ_w, μ_φ) convergence room.....	93
Figure 8: Harmonic oscillation area around equilibrium point.....	96
Figure 9: Set of order-2 convergence curves	112
Figure 10: Same Order-1, Order-2 harmonic oscillation curve	114
Figure 11: 2D view - order-2 convergence limits and μ_φ, μ_w of real algorithm runs ...	121
Figure 12: 3D view - order-2 convergence curves and MSAPSO stability curve	122
Figure 13: Visualization of set of convergence curves – Stability property MSAPSO..	124

Figure 14: Regression curves to proof MSAPSO stability property hypothesis	127
Figure 15: Variation options of MSAPSO Stability Line	132
Figure 16: Lowered convergence and stability curve with $N(0.5,0.475)$	134
Figure 17: Raised convergence and stability curve with $N(0.5,0.075)$	136
Figure 18: Balanced convergence and stability curve with $N(0.5,0.288675),UNIF$...	137
Figure 19: MSAPSO – Graphical Visualization of the “success” of a particle	143
Figure 20: MSAPSO - State transition diagram of particle i – Minimum Problem	146
Figure 21: HMP in the 2D search space with particles close to G_{best}	150
Figure 22: HMP in the 3D search space with particles close to G_{best}	151
Figure 23: Defining local search hyper cube around actual G_{best}	153
Figure 24: Defining one of the four possible global search hyper cubes.....	154
Figure 25: Defining centric hyper cube around HMP	156
Figure 26: Dynamic inertia weight with increments variation along MSF	163
Figure 27: Variation of standard deviation in normal distribution along MSF	166
Figure 28: Variation of social and cognition at optimal inertia along MSF	169
Figure 29: List of benchmarks and their general characteristics	176

Figure 30: List of benchmarks MIN-TEST in respective dimensions	184
Figure 31: Min 2D Long runs with minimum error over all algorithms	186
Figure 32: Min 2D Long MSAPSO compared to other algorithms	187
Figure 33: Min 5D Long runs with minimum error over all algorithms	188
Figure 34: Min 5D Long MSAPSO compared to other algorithms	188
Figure 35: Min 10D Long runs with minimum error over all algorithms	189
Figure 36: Min 10D Long MSAPSO compared to other algorithms	190
Figure 37: Min 30D Long runs with minimum error over all algorithms	191
Figure 38: Min 30D Long MSAPSO compared to other algorithms	191
Figure 39: Min 100D Long runs with minimum error over all algorithms	192
Figure 40: Min 100D Long MSAPSO compared to other algorithms	192
Figure 41: Min 250D Long runs with minimum error over all algorithms	192
Figure 42: Min 250D Long MSAPSO compared to other algorithms	193
Figure 43: Min 500D Long runs with minimum error over all algorithms	193
Figure 44: Min 500D Long MSAPSO compared to other algorithms	193
Figure 45: List of benchmarks MAX-TEST in respective dimensions	194
Figure 46: Max 2D Long runs with minimum error over all algorithms	196

Figure 47: Max 2D Long MSAPSO compared to other algorithms	197
Figure 48: Max 5D Long runs with minimum error over all algorithms.....	198
Figure 49: Max 5D Long MSAPSO compared to other algorithms	198
Figure 50: Max 10D Long runs with minimum error over all algorithms.....	199
Figure 51: Max 10D Long MSAPSO compared to other algorithms	200
Figure 52: Max 30D Long runs with minimum error over all algorithms.....	201
Figure 53: Max 30D Long MSAPSO compared to other algorithms	201
Figure 54: Max 100D Long runs with minimum error over all algorithms.....	202
Figure 55: Max 100D Long MSAPSO compared to other algorithms	202
Figure 56: Max 250D Long runs with minimum error over all algorithms.....	203
Figure 57: Max 250D Long MSAPSO compared to other algorithms	203
Figure 58: Max 500D Long runs with minimum error over all algorithms.....	203
Figure 59: Max 500D Long MSAPSO compared to other algorithms	204
Figure 60: Min 2D-500D Long runs with Sphere Function comparison wo ELM	204
Figure 61: Min 2D-500D Min error with Sphere Function comparison wo ELM	205
Figure 62: Min 2D-500D Long runs with Sphere Function comparison with ELM	205
Figure 63: Min 2D-500D Min error with Sphere Function comparison with ELM	206

Figure 64: Official Performance Data from Yang compared to MSAPSO tests	207
Figure 64: MSAPSO Structural Diagram	224
Figure 65: MSAPSO Escape Local Optima Mode	225

LIST OF EQUATIONS

Equation 1: SPSO – Particles velocity.....	28
Equation 2: SPSO – Particles position.....	28
Equation 3: SPSO – Weighted average position of the two bests	29
Equation 4: SPSO – Velocity clamping.....	30
Equation 5: UPSO – Global PSO variant formula.....	34
Equation 6: UPSO – Local PSO variant formula.....	34
Equation 7: UPSO – Unified Velocity Update of the particle	34
Equation 8: UPSO – Unified Position Update of the particle.....	34
Equation 9: UPSO – Linear increase of unification factor	36
Equation 10: UPSO – Modular increase unification factor	36
Equation 11: UPSO – Exponentially increased unification factor.....	36
Equation 12: UPSO – Sigmoid modulation of the unification factor	37
Equation 13: E-MPSO – Entropy Measures of best positions with regard to particle i ...	41

Equation 14: E-MPSO – Contribution of $f(x)$ of best pos. of particle i to all best pos.....	41
Equation 15: CLPSO – Changed velocity update equation	44
Equation 16: EVPSO – Escape Velocity term for particle	47
Equation 17: APSO – Cognitive factor adaption.....	58
Equation 18: APSO – Social factor adaption in APSO	58
Equation 19: APSO – Step width when calculating $c1$, $c2$	58
Equation 20: ALPSO I – Velocity update of particle i of the own (best) position.....	60
Equation 21: ALPSO I – Velocity update of particle i related to P_{best} closest neighbor	61
Equation 22: ALPSO I – Position of particle i	61
Equation 23: ALPSO I – Velocity update of particle i related to G_{best} position.....	61
Equation 24: SLPSO I – Assigned logarithmic weight to particle	65
Equation 25: SLPSO I – Temporal execution probability of strategy j	66
Equation 26: SLPSO I – Relative execution probability of strategy j	66
Equation 27: SLPSO I – Accumulated importance of strategy j over all particles.....	66
Equation 28: BAT Algorithm – Frequency update of bat at position x_i	72
Equation 29: BAT Algorithm – Velocity update of bat at iteration $t+1$	72
Equation 30: BAT Algorithm – Position update of bat at iteration $t+1$	72

Equation 31: MFFA – Relation Light Intensity and distance r from source	74
Equation 32: MFFA – Light intensity varied by distance r	74
Equation 33: MFFA – Attractiveness from beholders view varied by distance r	74
Equation 34: MFFA – Euclidian distance between a pair of fireflies.....	75
Equation 35: MFFA – Final update equation of a firefly i	75
Equation 36: MFFA – Genotype of a firefly i as a base for self-adaption.....	76
Equation 37: Stagnation condition of the MSAPSO.....	82
Equation 38: Base MSAPSO equations for convergence analysis	85
Equation 39: Base MSAPSO recursive equations	86
Equation 40: Simplified terms of the recursive order-1 equation.....	87
Equation 41: Expectation values of MSAPSO particles position	87
Equation 42: Dynamical Form of the MSAPSO order-1 convergence.....	87
Equation 43: Expectation values μ_w, μ_ϕ of the dynamical order-1 convergence.....	89
Equation 44: μ_w, μ_ϕ based form of the dynamical order-1 convergence	89
Equation 45: Characteristic Polynomial of the MSAPSO dynamical system	90
Equation 46: Determinant of the order-1 MSAPSO dynamical system	90
Equation 47: Eigenvalues of the order-1 MSAPSO dynamical system.....	91

Equation 48: Order-1 convergence limit MSAPSO dynamical system.....	93
Equation 49: Discriminant of the order-1 MSAPSO dynamical system	94
Equation 50: Equilibrium point of particles.....	94
Equation 51: Harmonic oscillation curve around equilibrium point	95
Equation 52: Recalled difference equation MSAPSO for order-1/order-2 convergence .	98
Equation 53: Dynamical Form of the MSAPSO order-2 convergence.....	100
Equation 54: Expectation Values of the order-2 iteration matrix	101
Equation 55: Cubic Eigenvalues equations of the order-2 MSAPSO dynamical system	103
Equation 56: Eigenvalues of the order-2 MSAPSO dynamical system.....	103
Equation 57: Order-2 convergence system for generic probability distributions	111
Equation 58: Discriminant of the order-2 MSAPSO dynamical system	113
Equation 59: Order-2 convergence system for MSAPSO with uniform distribution.....	118
Equation 60: Order-2 convergence system for MSAPSO with normal distribution	120
Equation 61: Final convergence room MSAPSO with uniform, normal distribution	123
Equation 62: MSAPSO Stability Line (MSL) hypothesis	125
Equation 63: Most optimal inertia weight values	140
Equation 64: MSAPSO – 2-View Perspective - Success of a particle i	145

Equation 65: Search space characterization - unimodal and multimodal areas.....	158
Equation 66: Base equations of XPSO	179
Equation 67: Dynamical system of XPSO.....	180
Equation 68: Constriction factor of XPSO	181
Equation 69: Motion equations of XPSO	181

1 INTRODUCTION

The different variants of Particle Swarm Optimization (PSO) invented over the last couple of years have solved many problems and they were quite successful. However, there is still a significant gap to reach the ultimate goal of a self-parametrizing and intelligent optimization algorithm, with the capability to adapt to different benchmark problem surfaces automatically. For example, in the area of highly specialized PSO algorithms the issue is, that there is still a lot of knowledge (empirical studies, researchers experience, etc.) needed about the parameterization of the specialized PSO variants. Also in the field of hybrid algorithms, it is still hard to determine what part of the algorithm is responsible for what positive or negative effect when applied to a set of benchmark problems. In case the problem context changes, a priori parameter assumptions on some previous problem might work not very well anymore. Whatever if it is a specialized PSO or a hybrid algorithm, there is still the central problem, based on what method and/or criteria(s) the overall algorithm need to change or adapt to respond to varying mathematical problem surfaces without using a static algorithmic strategy.

In Yang et al. (2013, p. 11), they describe this problem as “the search for the magic formulas of optimization”, in other words, what an optimal algorithm should bring along. The ideal algorithm from the authors believe, should start with an initial guess for the solution and then get to the optimum in one single step. As the benchmark problem surface

and starting conditions of a problem do vary, they also discuss the principle of self-adaption as an embedded principle of such an ideal algorithm.

1.1 Background PSO

Dr. Kennedy and Dr. Eberhart first introduced the Particle Swarm Optimization (PSO) approach in 1995. The basic nature of the PSO is a population based optimization approach, which is able to solve very complex discrete, continuous and noisy optimization problems. The first roots of the PSO approach appeared based on research of (Reeves, 1983), where he suggested in the field of computer graphics to model objects with particle systems, because he found out that complex objects could not be easily simulated by polygons and surfaces. A second important source of inspiration came from the field of social research, where the PSO algorithm itself used various principles from former social psychology research. Indicatory works was coming from (Reynolds, 1987), where he modeled the collective behavior of a flock of birds. (Heppner & Grenander, 1990) did further work in order to simulate the behavior of animals.

Other important influences were coming from the area of social psychology research were most significant contributions was brought in from (Novak, Latane, & Vallacher, 1994), where they discussed basic concepts of dynamical systems in social psychology and provided also computer modeling of social processes.

All these models led then to a set of rules on swarm intelligence, which strongly influenced the initial PSO approach suggested by (Kennedy & Eberhard, 1995).

In the meantime, many new PSO variants appeared in the research community since the original introduction in 1996 in his standard form. In this context, the following categorization into two main directions of research is possible.

The first category is the development of algorithms, which combines e.g. existing variants of PSO with concepts of other optimization approaches (hybrid algorithms), in order to combine the benefits of different algorithmic approaches. One example could be the synergy of a population-based PSO algorithm with e.g. evolutionary algorithms (EA's). Many other hybrid examples exist for leveraging synergies between different optimization techniques for the purpose of solving complex optimization problems.

In any case, the intent of the first direction is to broaden the applicability of such combined algorithms by using “best of both worlds” and then apply it to a broader set of mathematical problem types and benchmark problems.

The second direction is to research highly specialized algorithms of PSO, which just refer to one specific and/or a limited problem space. The main goal in the area of highly specialized PSO algorithms targets to improve the settings for PSO parameterization for a particular problem set and by that obtain superior performance and stability when solving these optimization problems.

Clearly, there are drawbacks with both approaches. Either there is the choice to perform very well on a small set of problems with a certain PSO variant or to combine characteristics from other optimization approaches such as Genetic Algorithms (GA's) into

the PSO algorithm, which may work more stable on a broader set of problems, but may perform not as well as a specialized PSO version. In addition, it is still difficult to determine in the area of hybrid algorithms, which effect comes from what piece of algorithmic combination.

The intent of this dissertation is to get into a third direction, where the purpose of this new direction is to work out an approach which works on the principle of self-adaption. The exact definition of self-adaption still need to be detailed, but in general it could range from dynamic adaption of PSO parameters to automatic adaption of PSO formulas up to the probabilistic invocation of different PSO variants or combinations of the beforehand mentioned options. All these self-adaption options will have the focus to improve the search methodology and the ability to adapt the PSO algorithm to the changing conditions of the actual examined problem surface. In addition, also the adaption of the social and cognitive learning behavior of the PSO algorithms as a core concept is a valid option in this context.

The described algorithm names Multi Self-Adapting Particle Swarm Optimization (MSAPSO) from now on.

Today the existing PSO variants employ two basic principles, which are:

- Particle Swarm Exploration (Diversification) - detection of the most promising regions in the search space, based on locally available information collected by the particle(s).

- Particle Swarm Exploitation (Intensification) - convergence of particles towards the best solutions, based on globally available information.

These two combined principles are the elementary concept of PSO and it works quite well on a lot of benchmark functions to find global optima's, but the model still have issues, when the PSO algorithm works on a broader set of benchmark problems or when within one benchmark problem there is a frequent change between unimodal and multimodal areas.

The challenges are, that in the exploitation phase the PSO algorithm may end up in premature convergence and in the exploration case there is a chance to see time delay during the convergence when searching for an optimum. In reality, there is a fine and granular line between both phases and one of the key issues of a well performing, stable and self-adaptive algorithm is to meet this optimal balance point independent of the underlying evaluated optimization problem.

Indeed, the No-Free-Lunch Theorem (NFL) seems to play a significant role here, which says that a specific algorithm may perform well on one specific problem but may not perform at all on a slightly varied problem. It seems obvious that there is further research needed to “solve” the above described problem scope of self-adaption of algorithms (e.g. dynamically chosen parameter sets).

As this balance point of exploration and exploitation is flexible by nature (today's researchers try to detect the right parameter sets empirically), there is a need to better understand the underlying principles (e.g. convergence of the overall PSO system when selecting dynamic PSO parameter sets). Self-adaption of PSO parameters is then one way to react to the “dynamics of the balance points”. It can be anticipated that the PSO system,

is influenced by various factors such as the structure of the benchmark problem, used probability distributions, stochastic processes of the system, dimensionality of the problem, chosen parameter sets of the PSO algorithm, etc.

As PSO is a social and/or cognitive algorithm, the question can be raised if and how changes in the strength of collaboration between particles do have an influence with regard to performant and stable system convergence. Secondly how the social dynamics influence the beforehand discussed optimal balance point of exploration and exploitation.

The basic idea of variation of social and cognition reaction and interaction goes back to work of Albert Bandura a social cognitive researcher in the 1960's. He found out that dynamic social learning not just stem from the fact of direct observation and imitation of others, but also depends on the rewarding mechanism with regard to the initiated strength of the social and/or cognitive reaction.

In an experiment called "Doll-experiment", he was able to show that human beings adapt the strength of their social reaction not just based on what they observe, but also on the fact what exactly was rewarded or punished. In the "Doll experiment", thirty-three boys and girls was shown a movie, where a grownup called "Rocky" treats a plastic puppet called Bobo very aggressively. The movie ends in three different variants, where in the first version another person who enters the room rewards the behavior of Rocky, in the second version the same person punishes what Rocky has done and in the third version the behavior stays uncommented. Kids who saw the rewarding of the aggressiveness of Rocky showed also significant increase of their own aggressiveness towards Bobo after entering the room, whereas kids who have sawn the punishment of Rocky had a much lower level

of aggressiveness. The level of aggressiveness was different in the various groups just because of the used rewarding or punishment model.

For the context of PSO this would mean that different strength level of social and cognitive behavior would potentially also have an influence on the reward or success model used. In other words, for example in unimodal problems more aggressiveness would be justifiable because there is more likelihood to find better solution, whereas in multimodal problems the same aggressiveness of individual particles and a 100% of success is contractionary as it will be “punished” being too aggressive by being stuck into local minima. So, this shows that it is not only beneficial to have an adaptive social strategy in the PSO algorithm., it is somehow mandatory, because success is a relative measure, which varies dependent on the underlying benchmark problem.

The parameters set used to control the behavior of the algorithm strongly depend on the type of the problem surface and the actual situation the algorithm is facing in an iteration.

1.2 Present state of research

In Sedighizadeh and Masehian (2009) the authors motivate the need for a natural computation (NC) paradigm composed of Epigenesis, Phylogeny and Ontogeny Algorithms to deal with complex real-world problems having noisy data, inflexible algorithmic structures and multi-dimensionality embedded into the problems. In this context, they divide NC into three main domains:

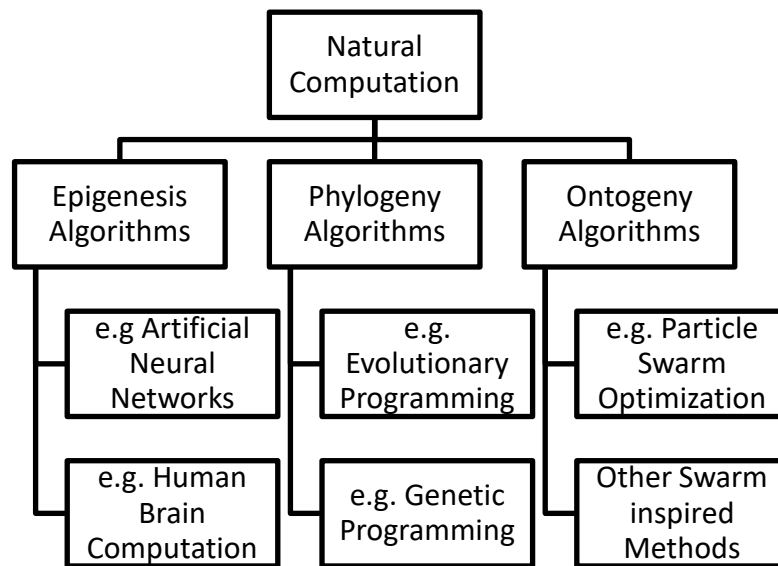


Figure 1: Categorization of Natural Computation

They define the NC categories in more detail as following:

- Epigenesis Algorithms: a complex structure which is able to perform tentative learning (e.g. human's brain, immune system)
- Phylogeny Algorithms: learning and performing is achieved via a competition algorithmic model (e.g. Evolutionary Programming, Genetic Algorithms)

- Ontogeny Algorithms: learning and performing is based on a cooperative strategy (e.g. PSO, MFFA)

In this definition, they set the context for PSO as a way to perform NC in a cooperative way among agents (particles). Based on this overall classification of PSO they describe several characteristics on how the PSO research branch itself could be described. They propose more than twenty aspects how PSO research branches potentially subdivides: For example, by the following characteristic:

- Continuity (continuous, discrete, binary)
- Topology (star, ring, random, etc.)
- Hierarchy (flat, hierarchical)
- Activity (active, passive)
- Compound with other heuristics (Genetic Algorithms (GA), Ant Colony Optimization, Neural Net, etc.)
- Attraction (attractive, repulsive, attractive-repulsive)
- Fuzziness (fuzzy, crisp)
- Divisibility (divided, undivided)
- Velocity Type (restricted, unrestricted, vertical velocity, escape velocity, ...)
- Other

As this approach to categorize seems to massively branch out the different aspects of PSO's, a new categorization method would be helpful, to better reflect the present and future research directions. The following categorization is a proposal on how to better structure

the existing PSO concepts and more important the recent research directions and activities.

A possible categorization is the following:

- Parameter optimized PSO's
 - Individual Parameter Optimization (manual, automatic)
 - Many Parameter Optimization (dependent, independent)
- Operator optimized PSO's
 - Individual Enhancements to the particles update equation
 - Automatic Enhancement to the particles update equations
- Information optimized PSO's
 - Analytic Information Gathering
 - Statistical Information Gathering
 - Historical Information Gathering
 - Prognosis Information based
 - Memetic information based
- Self-Optimizing PSO's
 - Self-Adaptive Parameters
 - Self-Adaptive Update Equations
 - Adaptive Algorithmic Selection
 - Self-Organizing Individual & Group behavior

The individual parameter(s) optimized PSO's take care about manual improvements with regard to parameter settings such as optimal neighborhood topologies, optimal choice of inertia weight, and others before the algorithm actually starts. The automatic parameter

optimization is valid during the PSO algorithm runtime and uses some logic to do so. The same is possible for many parameters optimization. As an add on in the many parameters domain it can be evaluated either manually or in automatic way if the parameters are dependent or independent from each other.

The operators optimized PSO's is about manual or automatic enhancements of the update equations of the particles positions under certain conditions. When conditions are met this will vary the particles trajectory compared to the Standard PSO and instantaneously influence the flight behavior of the particle in the actual iteration.

The third category is about getting useful information during the algorithmic runtime in a way such that it will not consume too many compute cycles. For simple unimodal problems, an analytical way might be more accurate than using statistical methods. Whereas in the case of more complex multimodal scenarios, deriving statistical information seems to be more appropriate because the analytical approach could be very challenging in that case. In higher dimensional problems, even the statistical information gathering might be inefficient, so nature inspired algorithms might be more helpful to get an idea how the search space looks like in order to find the best corresponding solutions.

The last class and this is a new direction is the approach of self-optimizing PSO's, where the core idea is to dynamically adapt the algorithms behavior to the actual problem space without having full control over parameters and even more without having the complete information about all neighboring particles. The promise made by self-optimization is the automatic adaption of e.g. parameters to the actual searched part of the problem space

without employing a fully informed algorithm or a priori parameter optimization. The figure below shows PSO variants that fit into the proposed PSO classes. The MSAPSO discussed in this PhD-document, will exactly work in the context of self-optimization.

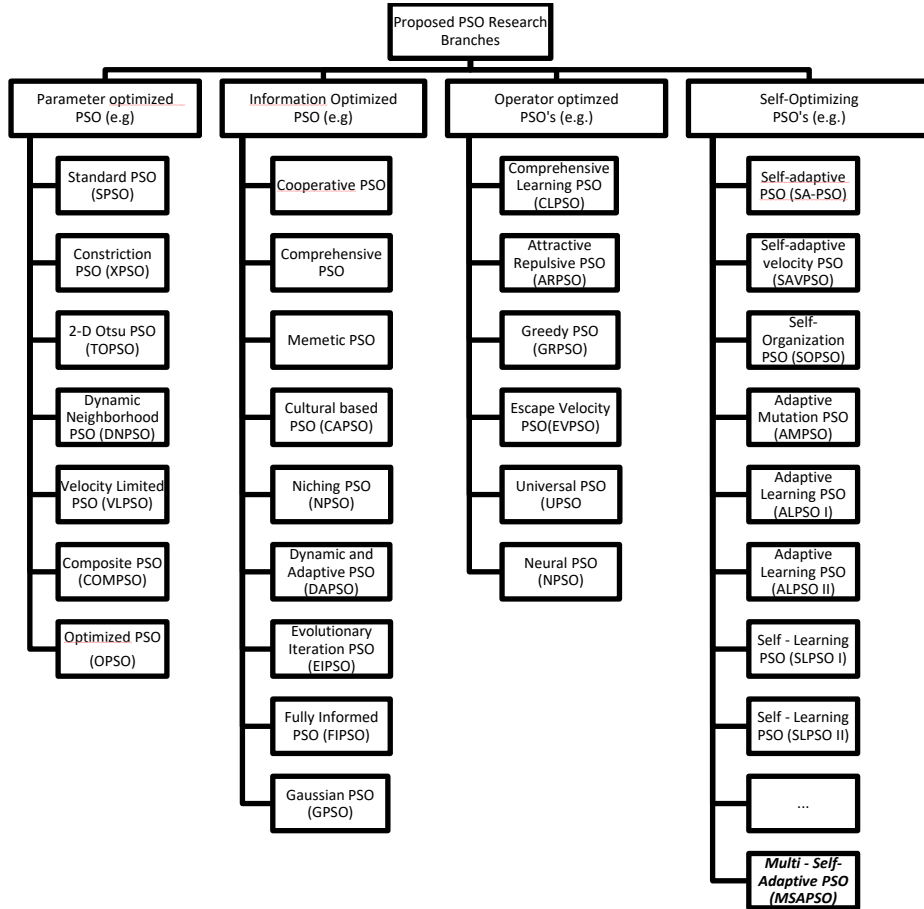


Figure 2: Proposed Classes of PSO Research Branches

Two leading Greece researchers in the area of PSO algorithms, describe the current state of research as follows in Parsopoulos and Vrahatis (2010, p. 269)

” The different PSO variants has been very useful in addressing continuous and integer problems, handling noisy and multi-objective cases, and producing hybrid schemes in combination with specialized techniques or other algorithms in order to detect multiple minimizer (local or global) or control its own parameters¹ “. In fact, they anticipate the following research areas as the main direction of future PSO research:²

- Theoretical Analysis of the PSO and their variants, especially but not limited to
 - The description of the full dynamics of the original PSO algorithm
 - The convergence criteria on complex problems
 - Better control on PSO parameters and building blocks in general optimization problems
- Strategies and operators, for example
 - The determination if actual particle velocities and operators applied to the particles are adequate or not
 - The question, whether it is worthwhile to always use the same strategy for all particles in a swarm
 - The discussion if hybrid methods are useful and if yes to what degree particular algorithm can improve the overall effectiveness
- Self-adaptive models, which claim to be the ultimate approach in dynamic optimization problems, where

¹ cf. Parsopoulos & Vrahatis (2010). Particle Swarm Optimization and Intelligence - Advances and Applications. Information Science Reference., page 269

² cf. *ibid.*, page 270-272

- The topic is to identify relations to other research fields such as artificial intelligence (AI), which is confronted with similar problems.
- The problem on how to develop appropriate proper operators and PSO variants along with an intelligent decision-making scheme that requires almost no parameter adaption und thus minimal user control.
- New variants of PSO suited to modern communication systems

In another scientific paper, (Bai, 2010, p. 182) comments about the future direction of PSO-Research are made, where it is outlined that four major fields of improvements in the next couple of years are expected:³

- The math's basic theory of the PSO algorithm
- Variation of the topology of the particle swarm
- Principles on how to blend PSO with other algorithms
- Further develop the application area of PSO in non-coordinate and scattered systems

The area of self-optimization behavior of PSO's is still at an early stage. The dissertation will focus on this field of research direction. In this context, the PhD work will compare to other fields of swarm research, where self-optimization is also a central principle such as

³ cf. Bai (2010, February). Analysis of Particle Swarm Optimization.
<http://www.ccsenet.org/journal/index.php/cis/article/view/5131/4314>, page 182

Ant-, Bee-, Bat-, Firefly-, Cuckoo-, Glowworm, Flower Pollination- and other swarm algorithms.

1.3 Problem statement

The problem in scope is to “solve” the conflict of extremely specialized PSO algorithms, which just perform on a specific set of problem spaces and at the same time address the issue that generalized PSO algorithms, which are very robust on a broad set of problems will lack similar performance capabilities compared to the specialized PSO versions.

The problem addressed, is to bridge the gap between best performance characteristics and extreme robustness of an algorithmic approach. A proper approach is searched, which unifies best of both worlds for the application on an extremely wide set of mathematical problems in a self-optimizing manner.

In this context MSAPSO shall avoid tuning of PSO influencing parameters, dynamically optimize particles behavior while searching for optimal solutions in the “unknown” and multidimensional search space. Also, the algorithm should be able to react dynamically to environmental changes with regard to the problem surface structure and probability distributions being used, while solving the optimization problem.

1.4 Research contribution

The research contribution focuses in the area of self-optimization of PSO based on a “to be defined” Multi Self-Adaptive Particle Swarm Optimization (MSAPSO) approach. The purpose of the new algorithm is the dynamic adaption on multiple levels at the same time.

These levels of MSAPSO are:

- Bidirectional learning strategy (social & cognition) with varying strength between the individual particle and the swarm itself
- Adaptive inertia weight strategy along different classes of benchmarks
- Dynamic detection of balanced exploration and exploitation points (optimal triples of inertia weight, social and cognition parameters)
- Use of adaptive randomization model with different probability distributions
- Finding dependencies between inertia weight and social & cognition parameters

The MSAPSO behavior can vary by self-adaptive social and cognitive parameters, dynamic inertia weight and the adaptive randomization strategy of the algorithm, during the runtime. The overall intent is to accelerate the convergence speed, while keeping the diversity of the search to avoid trapping into local minima or maxima. The multi-level self-optimization approach promises to generate positive synergies on both aspects (exploration and exploitation behavior of the algorithm) at the same time.

The concept of the dissertation will fall into four parts, where the first part takes care about the definition of a bidirectional learning approach, which is dynamic in nature because the strength of this cooperation strategy depends on the individual and group success of the particles in a varying problem search space (unimodal/multimodal and combinations of it).

An analogy to this aspect is the learning within social groups where it is of course beneficial to learn from the actual best, but still the question remains open to what degree this should happen. Although an individual might be not the best as of now, it could be that in the future very good personal success is possible and because of that, the individual should not just purely believe in the actual best in the group. In addition, the question is, if there is a natural limit of bidirectional learning which makes the overall convergence optimal, independent what the optimization problem is about. Also, the question can be raised how the variation of social collaboration and cognition changes with the increase of the dimensionality of the problem.

The second part will evaluate the concept of optimal balance points between exploration and exploitation, which should be agnostic from the underlying benchmark problem, so it is fully self-adaptive. In this context, there is a need to better understand the dynamics of MSAPSO in general, the convergence behavior, the influence of the problem benchmarks dimensionality and the applied probability distributions in the algorithm as influence factors to the optimal “balance point”.

The third element of the MSAPSO study will focus on the relation of social and cognition and inertia weight strategy. It is interesting to understand how changes in the collaboration

parameters influence the inertia weight parameter and vice versa and if rules can be found such that it can be applied to the overall self-adaptive model of MSAPSO.

The last aspect is the adaptive randomization during the algorithmic runtime. The idea here is that when convergence matures over time, there is decreasing need to equally distribute the particles all over the search space. The overall goal is to find very good solutions at the final convergence. So, based on this fact, there is the option to choose better fitting probability distributions to draw random numbers from, when the algorithm matures as there is better “knowledge” from the examined search space towards the end of the algorithmic runtime.

The MSAPSO algorithm does not limit itself to the four self-optimization aspects as proposed. In the future, many other aspects with regard to self-optimization might appear. If that happens, there is the question which of them contributes to the success, which are contradictory and which are synergetic to each other. Finally, this raises the question of which self-optimization strategy or combinations of it to use for a certain type of a problem surface. It is finally similar to the problem what also a human brain has to solve, which needs to decide dynamically which “algorithms” to use and also the need to dynamically parametrize the respective algorithms in order to best approach the actual faced problem scope.

In Yang et al. (2013, p. 9) the authors describe what an ideal algorithm should do:

„ that the algorithm simply has to tell what the best answer is to any given problem in one step! “⁴ Such algorithm surprisingly does exist in reality. For the special case of a quadratic function, this actually works with a root finding method called Newton-Raphson, which is able to find the global optimum within one-step. As mentioned, this is a special case and of course cannot be generalized. Today, there is no known way to create one universal algorithm that can provide the “one-step” answer to complex problems. On the other side, there should be still enough room for improvement in order to shorten the time of convergence (iterative steps) while still be able to work on a broad set of problems with the “same” algorithm. When we consider the self-optimization as a way to continuously adapt to the given underlying structure of the problem, then there is a chance to reduce the iteration steps dramatically in average. To translate this into the view of a particle, the following graph describes the “ideal” convergence with regard to a hypothetical self-optimization PSO algorithm:

⁴ cf. Yang et al. (2013). Swarm Intelligence and Bio-Inspired Computation. Elsevier Insights., page 9

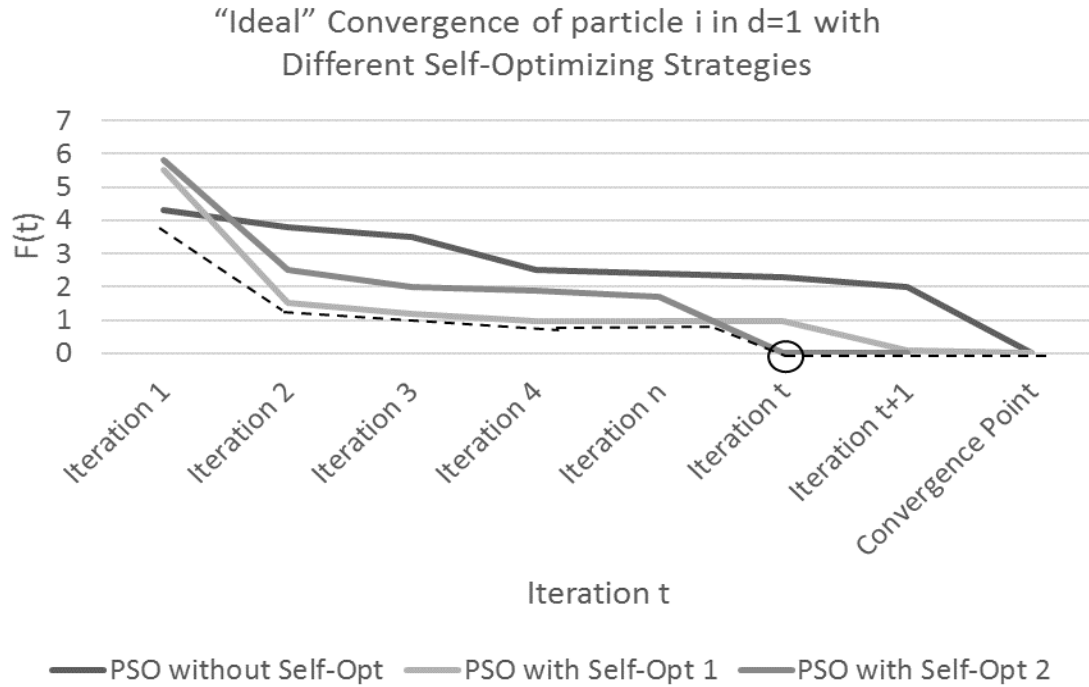


Figure 3: Ideal convergence behavior of a particle with self-optimization

In the graph at time point one, the different PSO variants takes a random and initial guess. In the following iterations, the two self-adaptive strategies realize improvements over the original “PSO algorithm without Self-Opt” (dark line). In iteration t PSO with “Self-Opt 2” (brighter dark line) realizes a benefit over the PSO “Self-Opt 1” strategy (brighter line). The resulting dashed black line would be the “ideal” convergence behavior of a to be defined multi self-adaptive algorithm.

On one hand, the nature of self-optimizing approaches should lead into broader applicability of the algorithm with regard to optimization problems. On the other hand, the

hybridization of different self-optimization strategies is more complex to understand with respect to their specific system and convergence behavior improvements and contributions.

1.5 Scope of PhD work

Although self-adaption in optimization algorithms is a broad and fast-growing research area today, ranging from artificial intelligence concepts into bio-inspired computation approaches the PhD proposal will focus and limit to the research branch of swarm intelligence and more specifically to a self-adaptive and parameter-independent Particle Swarm Optimization (PSO) approach.

In the last twenty years, PSO researchers covered many aspects of the original algorithm. It started with basic research studies conducted by Kennedy in 1995 with regard to the trajectory behavior of particles in the one-dimensional search space.

These studies found out that the stochastic velocity changes of the particles can expand into wider and wider cycles such that it gets uncontrollable, because parameter settings of the systems were exceeded. A simple method to avoid this was the introduction of a so-called velocity clamping. In later studies, new particle parameter concepts such as inertia weight (James Kennedy, 2001, p. 339) and constriction factor (Clerc, Particle Swarm Optimization (L'Optimisation par essaims particulaires), 2005/2006, p. 220) complemented the original PSO.

These parameters had the goal to better control the PSO algorithm behavior and therefore avoid particles velocity explosion. On the other side, this better understanding led to the improved applicability of PSO.

Finally, with the introduction and appearance of many new tuning methods for the PSO parameters, on one hand researchers were able to better analyze the dynamics of PSO, but on the other hand, the tuning of parameters had still to be done “manually” and finally adapted to every class of optimization problem. Because of this tradeoff, a new sub branch of research came up with the intention to automate parameter settings in PSO. At the same time, researchers tried to make the PSO variants and algorithms more self-adaptive, while still keeping the capability of the algorithm to solve a broad variety of problems. The PhD proposal exactly focuses on this aspect of particle swarm research. Conceptually the PhD proposal will clarify how social, cognition and inertia weight parameters are related to each other and how different probability distributions influence this relation during convergence and stagnation of the algorithm. Based on this knowledge MSAPSO should be able to define new criteria(s) for better convergence and also find flexible parameter settings of social and cognition “on the fly” for superior global and local search strategies. Of course, the new self-adaptive PSO approach should be also applicable and useful in the N-dimensional search space. Methodically we will test the effectiveness of MSAPSO by the use a broad set of benchmark functions, where the self-optimizing PSO will show the capabilities of self-adaptiveness, stable and fast convergence, independent of the underlying problem benchmarks examined.

2 LITERATURE REVIEW

The literature review will first cover the Standard Particle Swarm Optimization algorithm (SPSO) as a base model, the key PSO parameters and their contribution to improve the SPSO for specific optimization problems. This can be seen as the “what was going on within the PSO research segment” (vertical review). For the proposed classes of PSO algorithms two examples for every class will show the principles of every proposed research branch. For the section of self-optimizing PSO, a more granular study of the algorithms is necessary, as this is the main direction of the proposed MSAPSO.

Secondly, the literature review will extend horizontally into other related bio- and social-inspired research fields to investigate other bio-inspired concepts. The comparison will focus on: if and how these algorithms use self-optimization principles and can be an inspiration source for the design of the MSAPSO algorithm. Candidates for this comparison are listed below:

- Artificial Bee Colony Algorithm (ABCA)
- Artificial Ant Colony Algorithms (AACA)
- Memetic Firefly Algorithm (MFFA)
- Glowworm Algorithm (GWA)
- Cuckoo Search Algorithm (CSA)

- Bat Artificial Algorithm (BAA)
- Cultural Algorithm (CA)
- other

A more complete list of the above-mentioned algorithms can be found in the following literature sources (Commons Creative - Optimization Algorithms, 2011), (Commons Creative - Evolutionary Algorithms, 2011) and (Swarm Intelligence and Bio-Inspired Computation, 2013, p. 28).

The sections above will be complemented with the literature review of research articles, which relates especially to PSO variants, which embeds the self-optimization aspect. For the moment, self-optimization is not limited to the parameter level, it could also be on the algorithm equation adaption aspect or the flexible PSO variants invocation in a hybrid algorithm schema. The major intent is not to explain in detail the above-mentioned algorithms, but more to discuss the basic ideas. In addition, it is elementary to understand what elements in these algorithms could relate to the concept of self-optimization.

Former social and cognitive theories is also a major source of information for the PhD topic, because it can be the foundation on how the PSO particles relate to each other and what are interesting models of learning and cooperation strategies to apply it to MSAPSO.

The following sections will on one hand, set the foundation for the MSAPSO in terms of how the Standard Particle Swarm Optimization work and on the other hand focus on differentiating aspects of other related PSO variants. Furthermore, the literature review will

highlight and describe important self-optimization concepts from other biologically inspired algorithms as well as relevant social and cognitive theories. All parts of the review should then feed into the core idea of the PhD proposal.

2.1 Traditional research branches PSO

2.1.1 Base Model - Standard Particle Swarm Algorithm (SPSO)

The standard version of the PSO algorithm (SPSO) invented by Kennedy, Eberhardt and Shi in 1995 is a population based optimization algorithm described with all aspects in (James Kennedy, 2001, p. 287). In an iterative way, the algorithm tries to improve initial candidate solution with respect to a given measure of quality. A candidate solution equals a so-called particle. The collection of particles forms the swarm, which moves throughout the search space due to a certain mathematical movement equation. The particles itself do have relationships to other particles in so-called neighborhood topologies. These neighborhoods can be of various forms such as circles, partial-mesh, full-mesh or other connection forms. When updating each individual particle position, it is influenced by the local best position if neighboring particles are “fitter”, but at the same time each particles position is updated with the “effect” of the global best position of the entire swarm. This represents the social and cognitive behavior of the particle swarm. One key source of innovative thinking for the PSO was the Adaptive Culture Model (ACM) which rely on the following basic principles:

- Evaluate - Rate something as positive or negative, attractive or repulsive. Evaluation is the prerequisite for an organism to “learn”, where learning in this context: is a change that enables an organism to actually better rate in average (evaluate) its environment.
- Compare - Based on Festinger’s social comparison theory the principle comparison appeals to others as a kind of a motivation to learn and change the own behavior (Festinger, 1954, pp. 1-16). In fact, in ACM and PSO the individual compares to its neighbors and the global best on the critical measure and imitate only those which are superior to the own performance.
- Imitate - While monkey see, monkey do. In fact, this is not the same than learning through try and error. It is instead learning by direct observation or observational learning originally described in Bandura’s social cognitive theory (Bandura, 1986, p. 21). The difference is that someone can learn, even he has not seen the specific behavior before.

Overtime this method(s) converges each particle to a global optimum in the search space. One key execution principle, which embeds the basic concept of the algorithm, stems from the fact that it will switch between two modes of operation.

- Exploration – evenly distributed global search in the appropriate dimensions of the optimization problem to cover a broad range of the problem search space.
- Exploitation – rapid convergence to a promising optimum. This exploitation phase will occur locally around the Pbest positions, with the goal to find better solutions.

The formal notation of the Standard PSO in the real-numbered space denotes as following:

- \vec{x}_{id} – Position vector of a particle i in any relevant dimension d
- φ_1, φ_2 – random numbers from a chosen probability distribution
- c_1, c_2 – cognitive and social weighting factor, which describe trust into individual and group behavior
- w – inertia weight – preservation of previous velocity of particle i
- $\vec{\Delta x}_{id}$ or \vec{v}_{id} – Change of a particle position in any relevant dimension d or simply velocity of a particle. Velocity is a vector of numbers that adds to the position coordinates in order to move the particle from one-step in time to another step-in time.
- $\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$ – How to actually search the “Search Space” via particle position update equation, which is influenced by the appropriate underlying problem benchmark and the randomization of the particles position.
- \vec{P}_i – Individual best position of a particle i
- \vec{P}_g – Global best position of a particle found so far by the entire particle swarm

The PSO algorithm samples the “Search Space” by modifying the velocity term. A structure of the neighborhood of the particles influences again the search process and has impact to the individual position. In general, the “direction” of movement is a function of movement of the current position and the velocity update, the location of the individual’s previous best and the best position found globally.

Furthermore, the combined “change” is defined as a function of the difference between the individual’s best position and the current position (cognitive portion) as well as the difference of the global best position (social portion) and the current position.

The following things conclusively change during the execution of the PSO algorithm:

- Particles velocity:

$$\vec{v}_i(t) = w\vec{v}_i(t-1) + c_1\phi_1(\vec{P}_i - \vec{x}_i(t-1)) + c_2\phi_2(\vec{P}_g - \vec{x}_i(t-1))$$

Equation 1: SPSO – Particles velocity

- Particles position :

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$$

Equation 2: SPSO – Particles position

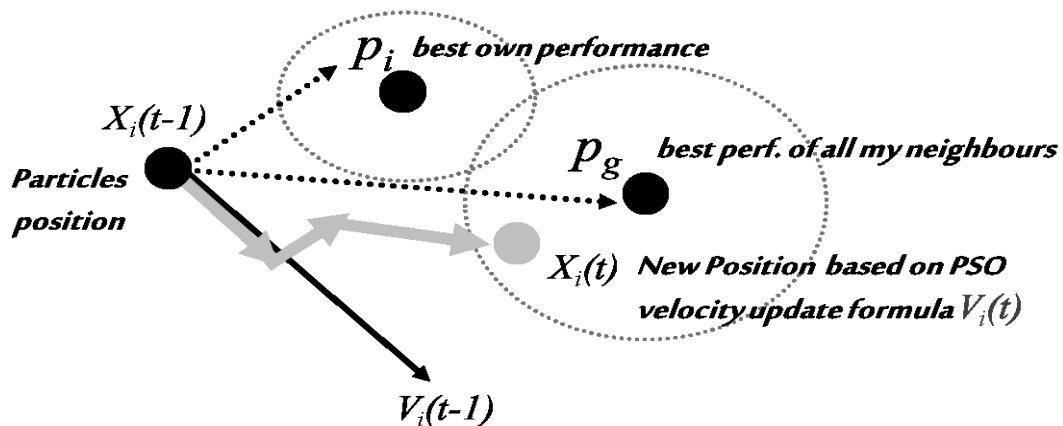


Figure 4: Visualized PSO velocity and position update of particle i

This figure describes the particles velocity and position update for the following iteration. It sums up three different tendencies (best own performance, best performance of all neighbors and the velocity of the last iteration). Based on the PSO update formula as described in Equation 1: SPSO – Particles velocity and Equation 2: SPSO – Particles position, the new position of the particle i is then calculated.

- φ_1, φ_2 – are random numbers from a random distribution usually defined in the interval $[0,1]$ to randomize the PSO algorithm and make it a stochastic optimization approach and process
- Weighted average of the two bests $\vec{P}_i \vec{P}_g$

$$\frac{(\varphi_1 \vec{P}_i + \varphi_2 \vec{P}_g)}{\varphi_1 + \varphi_2}$$

Equation 3: SPSO – Weighted average position of the two bests

The effect of the weighted average is that the particles “cycles” or oscillates around this point during the algorithmic run. The system has a tendency to explode when parameter setting of SPSO are exceeded. Then oscillations of particle’s trajectories become wider and wider unless a method is applied for dampening it. The method to do this is to introduce a so-called velocity clamping.

- V-Max parameter to limit explosion for every individual particle i on each dimension d.

$$\text{If: } V_{id} > V_{\max} \text{ then } V_{id} = V_{\max} \quad \text{else if } V_{id} < -V_{\max} \text{ then } V_{id} = -V_{\max}$$

Equation 4: SPSO – Velocity clamping

In early phases of the Particle Swarm research one of the key issues was the empirical determination of the PSO parameters so it fits to various problems. By adapting the parameters, researchers were able to adapt the algorithm to the explored problems. This was clearly an unpractical way of applying PSO to broader range problems. Therefore, a new research direction was to minimize the tuning of parameters and the parameters itself, such that it corresponds to more potential benchmark problems. In the following sections, an approach is described, which shows the principles of parameter optimized PSO's.

2.1.2 Parameter optimized PSO's

These variants of PSO's primarily deal with the optimization of parameter values itself (manually, experimental, other), which are important to tune and let better perform the PSO algorithm. In addition, the reduction of parameters is in scope. Furthermore, optimal parameters combinations will have the potential to improve PSO convergence. This testing can happen manually or with the help of e.g. Evolutionary Algorithms (EA's). The PSO depends in his standard version on the following factors:

- Cognitive factor $c1$ – believe in own search
- Social factor $c2$ – believe in aggregated search of others
- Inertia weight w – how much speed to take from last iteration

Many PSO variants try to optimize the parameters so it fits to the actual optimization problem. The main problem that exists is that the researchers manually or experimentally adapts parameters to the optimization problems. Due to this approach, it is quite clear that without a proper approach to do this automatically the PSO variants cannot get into the direction of a universal algorithm. Secondly, because of this issue, researchers try to minimize the needed parameters in their algorithmic logic with the purpose to optimize the parameter settings and usage. The idea is the less parameters are needed the less manual or experimental effort comes up to make the PSO variant performing. In the following paragraphs, two examples from this PSO research branch are evaluated.

2.1.2.1 Unified Particle Optimization (UPSO)

UPSO is a parameter optimized PSO, which means that it minimizes the number of parameters needed. The working principles are described by the authors as the inventors of UPSO in (Parsopoulos & Vrahatis, 2010, p. 89). As in the SPSO the UPSO employs two main phases during the iterations and while solving the optimization problem.

- Exploration (detection of the most promising regions of the search space)
- Exploitation (convergence of particles towards the best solutions)

The two phases can take place either once or successively during the execution of the algorithm. For the transition between the two different modes, a so-called unification factor controls the switch over between exploration and exploitation. Before the discussion is made about the unification factor, there is a need to discuss certain characteristics of UPSO.

An important factor in UPSO as in other PSO variants is the neighborhood size. The two subparts in PSO is neighborhood on Gbest (Global Best) and Lbest (Local Best).

- Gbest – the whole swarm is considered as a neighborhood of each particle
- Lbest – where neighborhoods are strictly smaller and local to an individual particle

More specifically the global variant (Gbest) converges faster towards the overall best solution than the local one (Lbest) in the most common neighborhood topologies. Therefore, it is mostly favorable for its exploitation capabilities and characteristics.

On the other hand, the local variant has better exploration capabilities, since information about the best position is gradually communicated to others. (Particles are gradually attracted – this helps avoiding trapping early into suboptimal solutions).

Obviously, also the tradeoff between neighborhood topology and swarm size affects the two different modes of exploration and exploitation, but there is no formal procedure from the authors point of view to optimize it. The most common neighborhood configuration consists of a ring applied to Gbest or Lbest. Under such configuration, the algorithm is biased either towards exploration or towards exploitation, depending on the complexity or “difficulty of the problem”⁵ examined.

USPO main idea was to combine the two-phase’s exploration and exploitation in one generalized manner, such that a new scheme combines the two properties and minimizes

⁵ cf. LeClerc (2005). Particle Swarm Optimization. LAVOISIER. Chapter 1

the parameters. This unification of the exploration and exploitation in fact reduces the number of parameters to configure and adapt.

In this context, the constriction factor χ of (LeClerc, 2005/2006, p. 223) for UPSO was used to control the UPSO convergence behavior.

UPSO unification of global and local PSO defines itself as following, in addition to SPSO definitions:

- N – denotes as the swarm size of the particle swarm
- $G_{ij}(t+1)$ – denotes as the global velocity update of the particle i in dimension j to update particles position x_i for the global PSO variant with constriction coefficient χ
- $p_{gi}(t)$ – denotes as the best position in the neighborhood of x_i for the global update equation
- $L_{ij}(t+1)$ – denotes as the local velocity update of the particle i dimension j to update particles position x_i for the local PSO variant with constriction coefficient χ
- $p_{ij}(t)$ – denotes as the best position in the neighborhood of x_i or global update equation and local update equation
- u – denotes as unification factor $[0,1]$ – with values between zero and one. In fact, u attaches weight to either explorative or exploitation behavior of the algorithm
- $i = 1, 2, \dots, N$ is denoted as the particle i
- $j = 1, 2, \dots, n$ is denoted as the dimension j

The Global PSO variant in UPSO denotes finally as:

$$G_{ij}(t+1) = \chi [v_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p_{gj}(t) - x_{ij}(t))]$$

Equation 5: UPSO – Global PSO variant formula

The Local PSO variant is denoted as:

$$L_{ij}(t+1) = \chi [v_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p_{lj}(t) - x_{ij}(t))]$$

Equation 6: UPSO – Local PSO variant formula

The Unified Particle Swarm (UPSO) then denotes as:

$$v_{ij}(t+1) = u G_{ij}(t+1) + (1 - u) L_{ij}(t+1)$$

Equation 7: UPSO – Unified Velocity Update of the particle

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1)$$

Equation 8: UPSO – Unified Position Update of the particle

Global PSO ($u = 1$) and Local PSO ($u = 0$) are special cases of the UPSO. All other intermediate values define variants of the UPSO, which actually combines the two search directions.

Evidently, lower values of u correspond to distributions biased towards the local best position. Consequently, L_{best} position dominates then in UPSO. Increasing u towards one results in a shift to a more global best position equation and an expansion towards a global search.

Thus, u can control the expansion of new positions for each particle – in fact controlling the exploration/exploitation properties.

In general, when $u < 0.5$ the local search direction is dominant, hence the algorithm is mostly influenced by it. The opposite must hold for $u > 0.5$.

There is an obvious dependency between UPSO swarm dynamics and the unification factor. The unification factor controls the balance between exploitation and exploration.

- Small values of u favor the local algorithm resulting in better exploration
- Larger values of u favor the global algorithm promoting exploitation
- Values around $u = 0.5$ produces more balanced behavior of the algorithm. However, such balanced behavior fails to take advantage of any special structure of the objective function such as convexity, unimodality and others. In such cases more extreme values such as zero or one may exhibit better performance
- Unification factor can be considered at swarm level or at particle level. In the first case, particles have the same behavior for exploration/exploitation (aggregated behavior). In the second case, each particle has its own special exploration/exploitation tradeoff (behavior diversity).

Latest developments of UPSO show initial steps towards self-adaption for the unification factor. Adaptive changes of u were developed with various update methods such as

- Linear Increase: unification factor is linearly increased from zero to one according to the following formula at every iteration t :

$$u(t) = \frac{t}{T_{\max}}$$

Equation 9: UPSO – Linear increase of unification factor

Which corresponds to a small and relatively slow transition from exploration to exploitation

- Modular Increase: Unification factor is increased repeatedly from zero to one every q iterations with the following formula:

$$u(t) = \frac{t \bmod (q+1)}{q}$$

Equation 10: UPSO – Modular increase unification factor

- Exponential Increase: Unification factor is increased from zero to one exponentially according to the following formula:

$$u(t) = \frac{t \log(2.0)}{T_{\max}}$$

Equation 11: UPSO – Exponentially increased unification factor

- Sigmoid Unification Factor: Another type of Unification Factor is the Sigmoid Unification Factor, which works on a Swarm Level. The scheme for the sigmoid transition from exploration into exploitation is the following:

$$u(t) = F_{\text{sig}}\left(t - \frac{T_{\text{max}}}{20}, \lambda\right)$$

With

$$F_{\text{sig}}(x, \lambda) = \frac{1}{1 + \exp(-\lambda x)}$$

Equation 12: UPSO – Sigmoid modulation of the unification factor

The form of the sigmoid transitions depends on the parameter λ , where lower values of λ transition more smoothly than higher values, when changing from exploration into exploitation.

This different scheme for manipulation of the unification factor results in various transition models from exploration to exploitation during the iterations.

UPSO is a step to self-optimization of PSO, but it still embeds a couple problems. The first one is that there are still underlying parameters such as c_1 , c_2 , χ which are still inflexible from a parameter setting point of view. Although these parameters accumulate now in one unification factor, it is still not sufficient, because it will not reduce the amount of investigation for the underlying parameters to make them optimal with regard to the investigated optimization problem. Secondly, the proposed adaption mechanism's does not embed a flexible logic, which fits to various optimization problems in order to switch

automatically over between exploration and exploitation. All adaption functions (linear, sigmoid, and exponential) do assume a tendency, which might work in a couple of cases but for sure be not optimal for a broad range of optimization problems as search spaces within and between problem benchmarks can vary extremely.

2.1.2.2 Composite PSO (COMPSO)

The Composite PSO approach (COMPSO) is described in (Parsopoulos & Vrahatis, 2010, p. 111). COMPSO is an application to PSO based on the Differential Evolution algorithm (DE) introduced by Storn and Price (1997). The idea behind COMPSO is that the three basic parameters of PSO (inertia weight, cognitive and social factor) probe the search space such that the PSO will get optimal with regard to speed and stability of the algorithm.

This is achieved by defining a so-called three-dimensional individual, where the elements of the individual's vector are equal to the three basic PSO parameters. During every iteration, a new swarm S_t is composed with the probing individual q_m where $q_m = \{w_m, c1_m, c_m\}$. This parameter set applies then to the velocity update equation. Subsequently over time, COMPSO tests the best particle x_i with the actual best functional value $f(x_i)$. The individual vector for every particle changes in every iteration using the Differential Evolution algorithm, which executes the mutation of the individuals in every iteration.

Although in this case, COMPSO uses the basic PSO parameters to probe the search space, the randomization for the mutation of individuals still do not have a thought through logic behind. COMPSO has a pure underlying evolutionary concept (DE). A concept of success of a particle would be worthwhile to implement as an algorithmic logic rather than randomly probing the parameter vector. An interesting effect to keep in mind is that COMPSO always assigns larger values to c_1 rather than c_2 . A mathematical understanding of the embedded convergence behavior is not performed in order to better direct the random mechanism of the probing method. Nevertheless, COMPSO shows a promising approach which works on a lot of benchmark problems.

2.1.3 Information optimized PSO's

As described in the introduction chapter, information optimized PSO's general approach is about gathering information from analytics, statistics, and information from other algorithms or the search space in order to improve the classical PSO scheme. On one hand, it is favorable to gather as much information as possible to make better exploration and exploitation decisions. On the other hand, exactly this "collecting" of information consumes a lot compute cycles. In fact, it is the tradeoff to find the right balance when collecting supporting information for the optimization problem versus having a very good performance during the algorithmic run. It very much relates to the No-Free-Lunch theorem (NFL) referenced in (Wolpert, 1996), which says "...that for any algorithm any

elevated performance over one class of problems is exactly paid for in performance over another class”⁶.

When this is translated into the information gathering aspect this would mean then, independent how much information an algorithm gathers in average over all costs function this algorithm will not be better than another algorithm not doing that. On the other site, there is critic from a couple of researchers that in certain areas this is not true as the problem area’s itself imply some structure whereas in NFL argumentation they claim the validity over all problems which would not assume any underlying structure then. As a conclusion, it is still an open debate when and how NFL applies. With regard to information based PSO’s the prerequisite will be that there is some structure where information can be derived from, so it is at least still possible that there would be an information optimized PSO that could perform better on a large variety of problems compared to others. In the following paragraph, again two examples are discussed, which shows the principle of information gathering to improve the classical PSO.

2.1.3.1 Entropy-Based Memetic PSO (E-MPSO)

The basic concept of information theory and information entropy was originally introduced by (Shannon, 1948). Shannon Information Entropy (SIE) is a measure of mess for E-MPSO described in (Parsopoulos & Vrahatis, 2010, p. 105). The main application field of SIE in general within optimization algorithms was consequently that of a diversity metric.

⁶ cf. Holpert (1996). No Free Lunch Theorems for Optimization.
<http://www.no-free-lunch.org/WoMa96a.pdf>.

The definition of E-MPSO is as following:

P – Population

K – Phenotype classes

Q_k – Proportion of P occupied by K

Q_s – User defined selection probability as a threshold

SIE – $SIE_t(P) = - \sum_k Q_k \log Q_k$ representing the amount of chaos in the system

Small values of Q_k correspond to high entropy, whereas high entropy indicates a higher population diversity. In the context of the swarm $S = \{x_1, x_1, \dots, x_1\}$ of N particles and population $P = \{p_1, p_2, \dots, p_n\}$, which are the corresponding best positions, then at a given iteration t, SIE in a particle context is defined as:

$$SIE_t(P) = - \sum_{i=1}^N Q_i \log Q_i$$

Equation 13: E-MPSO – Entropy Measures of best positions with regard to particle i

$$Q_k(t) = \frac{f(p_i(t))}{-\sum_{i=1}^N f(p_i(t))}$$

Equation 14: E-MPSO – Contribution of $f(x)$ of best pos. of particle i to all best pos.

High Values of SIE indicate widely spread functional values of the best positions, while small values show a narrow spread (similar functional values of best positions). SIE can be used as an information so PSO can decide whether it goes for a wide spread search (exploration) or for a narrow search or convergence behavior (exploitation).

It makes a lot of sense for PSO to gather such information to optimize the further logic and decisions within PSO. The disadvantage will be that a lot of computation needs to be done upfront in order to get to that decision point. For every particle, all best positions need to be taken into consideration before the proportion can be calculated and finally the entropy as a measure of particle diversity can be derived. In addition, another weakness is that the proportion Q_s needs to be set manually, so that the useful information gathered is bought via another manual parameter which needs to be tuned empirically.

2.1.3.2 Niching Particle Swarm optimization (NPSO)

Niching PSO described in (Parsopoulos & Vrahatis, 2010, p. 119) is an approach, originally suggested by (Brits, Engelbrecht, & van den Bergh, 2002). NPSO focuses on the so-called “cognitive only model” at the beginning. In that regard initialization of the NPSO plays a central role, because this model assumes independence for the individual particle i while searching locally or individually. NPSO uses a special set of random numbers for that (Faure random numbers).

The NPSO algorithm searches for so-called niches in the swarm. The information, which provides that, is the measurement of the variance of the functional value of a particle for

several iterations. If it falls under a certain value (threshold), the algorithm creates a new sub-swarm with this particle and his closest neighbor. The intent is to close the “niche”.

Although the niching information is useful for both convergence as well as diversity of the swarm, there is still the need to define a manual threshold, which is set by the researcher.

Again, this is an example for the tradeoff between collecting sufficient information to make decisions in PSO versus to introduce new complexity because the information gathered needs new parameters (in this case a threshold parameter).

2.1.4 Operator optimized PSO's

The class of operator optimized PSO's try to improve the update equations of the PSO algorithm itself. The consideration here is that differently structured search spaces of optimization problems should get a representation in how particles surf across these structures. In a fuzzy structure with slight variations with regard to functional values (“many multimodal” problems), probably a uniform and broad flight and speed of the respective particles is more appropriate.

In contrast, in a clear defined structure optimization problem (unimodal problems) with a strong ascending or descending surface the PSO do not have to search broadly but rather very fast because of the nature of the underlying problem surface. In addition, in dynamic optimization problems there may be the need to react fast and quickly according to the speed and direction of the particles and the swarm.

The next two PSO variants will give an example how to solve these challenges with changes in the operator equation of the PSO itself. Two ways seem to be appropriate:

- Individual Enhancements to the particles update equation
- Automatic Enhancement to the particles update equations

2.1.4.1 Comprehensive Learning Particle Swarm optimization (CLPSO)

The CLPSO algorithm describes itself in (Liang & et al., 2006). A major issue what CLPSO addresses is the solution finding efficiency for multimodal problems. The algorithm has the following changed velocity update equation compared to the original PSO. It is therefore an example for an operator optimized PSO.

$$v_i^d = w v_i^d + c^* \varphi_i^d (pbest_{fi(d)} - x_i^d) \quad \forall d = \{1 \dots D\}$$

Equation 15: CLPSO – Changed velocity update equation

with

$f_i(d) = \{f_i(1), f_i(2), \dots, f_i(D)\}$ – defines which particles 'Pbests' with regard to the dimension d the particle i should follow. In this case, the flight is not the classical way, which goes iteration by iteration, but it is rather a method where the particle i parses all dimensions with the PSO algorithm. All dimensions the particle is associated with, potentially determine the overall fitness of a particle. This is a characteristic, which the algorithm can use to find excellent solutions. First, a learning probability defines whether

something is to learn or not. In addition, as long there is something to learn, the algorithm generates two functional values within the same dimension for the same particle normed to the population size p_s . These functional values compare to each other and the largest value is stored in a variable. Then the particle “surfs” on to his next dimension to do the same in the next dimension and so on. All particles will search for their optimum in the same way and subsequently will find the global optimum.

The dimensional flight of the particle does show his advantages in multimodal functions and show a good diversity of the swarm. However, on the other site in high dimensional problems with simpler functions this way of updating the velocity is suboptimal. The main reason for this is that CLPSO do have a larger search range than SPSO. The more complex the problem is (higher dimensions, complex surface) the better CLPSO seems to work. The flowchart is referenced from Liang et al. (2006, p. 283) and the particles’ dimension surfing is there described as following:

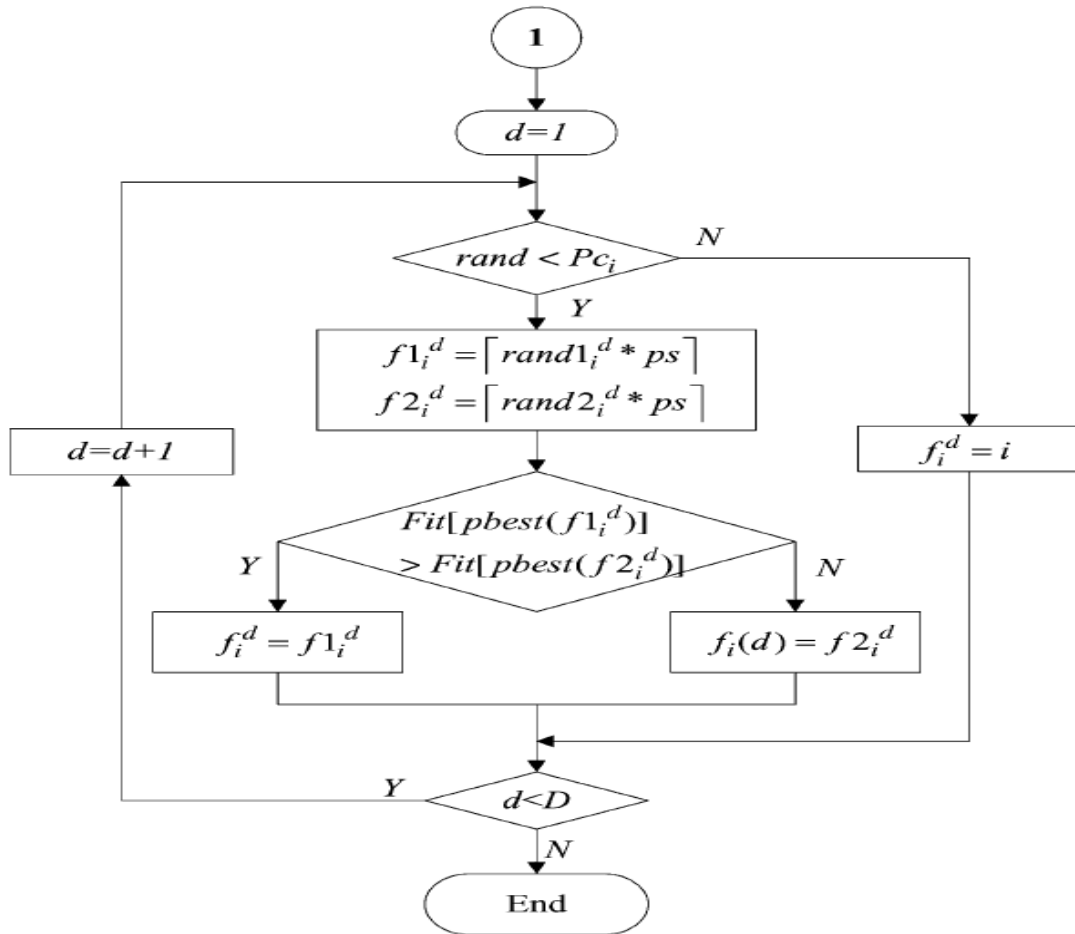


Figure 5: CLPSO – dimension surfing based velocity updating of a particle

2.1.4.2 Escape Velocity Particle Swarm optimization (EVPSO)

The EVPSO algorithm is initially described by a Chinese research team from Wang et al. (2006) and it shows another example of operator optimized PSO's. The major intend of the algorithm is to avoid trapping into local optima. It uses the regular velocity update formula Equation 1: SPSO – Particles velocity and removes the velocity update of the last iteration with the so-called escape velocity:

$$v_e = \begin{cases} v_{i,j}(t), & |v_{i,j}(t)| > e_c \\ r_j \times v_{i,j}(t) \mid \rho |v_{i,j}(t)| < e_c \end{cases}$$

Equation 16: EVPSO – Escape Velocity term for particle

In this context, r_j are random numbers from a uniform distribution within the interval of $[-1, 1]$, ρ is a scaling factor that defines a region relevant for the escape velocity and e_c is a configured parameter which actually decides when the escape case happens ($e_c < 1$). In the situation, when many particles are stuck within a local optimum, the stochastic escape velocity then actually increases the likelihood to create a velocity larger than the basin diameter where the particles are potentially stuck in. Performance of EVPSO directly correlates with the parameters and e_c and ρ . A large value of e_c shortens the time to escape which is equal to again perform a global search. A low value leads exactly to the opposite. For the escape case of the particle i ρ need to have a large value, which actually reduces the escape domain for the particle (particle gets faster out of the basin). Whereas this parameter is optimal for the escape case of the algorithm (large e_c , large ρ), during the regular run these setting can be very suboptimal.

For the desired behavior of a balanced exploration-exploitation, the algorithm works in two phases, at the first stage, e_c is set at a large value, and ρ is assigned a small value (broad exploration to look for good minima), at the last stage, e_c is set at a small value, and ρ is set at a large value (fine granular search).

With these settings the particles make very large movements at the beginning and scan the whole solution space for candidate solutions in the early stage, and they perform a fine grain search in the final stage.

The critic to this approach is the following. Although it might be possible to escape from local minima in early stages of the algorithmic search, this capability “is bought” by the need for manual tuning of e_c , ρ . It is very difficult to find good criteria on how to do the parameter settings automatically when the variety of benchmark problems are considered. Also in dynamic problems where the problem surface changes over time, it is hard to imagine that this approach leads to good results without tuning these parameters repeatedly.

2.2 Self-Optimization as new PSO research branch

2.2.1 Principles of Self-Organizing Systems

In Haken (1983, p. 191) the author describes “Self-Organization” in the context of a group of workers. A process can be seen as self-organized when:

“There are no external orders given, but the workers work together by some kind of mutual understanding each one doing his job so as to produce a product”⁷.

Self-organization in general is a process where some form of global coordination appears out of the local interactions among agents in an initially chaotic system. This process is spontaneous and emergent. There is no master that controls the system itself neither from the inside view nor from the external view.

State changes in the self-organizing system are often triggered by random fluctuations that are amplified by positive feedback. The resulting organization has the characteristic of decentralization among all the elements in the system. As such, it is typically very robust and able to self-repair. Chaos theory discuss the “self-repair capability” as a state of predictability in an ocean of chaos.

⁷ cf. Haken (1983). Synergetics An Introduction Nonequilibrium Phase Transitions and Self-Organisation. Berlin Heidelberg: Springer Verlag., page 191

Furthermore, in Ashby (1962, pp. 255-278) the author describes that:

“Any deterministic dynamic system will automatically evolve towards a state of equilibrium (or in more modern terminology, an attractor). As such it will leave behind all non-attractor states (the attractor's basin), and thus select the best attractor out of all others. Once there, the further evolution of the system is constrained to remain in this condition. This constraint on the system as a whole implies a form of mutual dependency or coordination between its subsystems or components. In Ashby's terms, each subsystem has adapted to the environment formed by all other subsystems.”

In the area of biological systems, examples from bird flocking and other natural inspired systems such as bee and ant colonies also show the relation to self-organizing behavior. Particle Swarm Optimization is also an example for a self-organizing system, especially because of the following definition coming from Camazine et al. (2001, p. 8).

“In biological systems, self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern”⁸

It is obvious that the particles itself with their interactions can represent the lower level in the system. Also, the rules how particles interact (neighborhood structure) and how they exchange information corresponds with this definition. In addition, randomness and

⁸ cf. Camazine et al. (2001). *Self-Organization in Biological Systems*. Princeton University Press, page 8

positive amplification is also part of PSO. Think about, when a particle becomes Gbest: then others are attracted, but also other may attract the actual best individual particle again in the next iteration. Over all iterations, the portion of amplification coming from the Gbest location which finds his way into the system and in other iterations, the system gives it back as portion of amplification back to the individual particle again (self-amplification). The last prove is, whether there is an attractor element in PSO and yes, the attractor is the actual Gbest Position of all particles, so this look similar like the above described equilibrium state in the system.

In the science of Self-Organization from (Haken, 1983), further hints can be found that PSO by its nature qualifies as a self-organizing system. The theory describes the Brownian particle movement (unpredictable model with emergent orders in the system), which has strong similarities to the PSO Model. PSO, which has randomness elements by nature, but with social and cognitive influenced trajectories of the particles also wants to achieve an higher order attractor state (convergence) out of the system (emergence).

It is important to understand what the “inbuilt” capabilities of PSO are, in order to design better PSO algorithms. Based on the above discussion, PSO shall have the characteristic of self-organization by using self-adaptive parameter settings of social, cognition and inertia weight. The parametric rules on a lower level should also attract the self-healing capabilities of the swarm in an emergent way when environmental conditions changes (e.g. complex problem surfaces, dynamic optimization problems, etc.).

2.2.2 Self-Optimizing PSO's

In this section, several PSO variants are discussed, which focuses on the class of self-adaptation approaches and related concepts. In this context, the literature review focuses in the first part on inherent self-adaptation capabilities, such as automatic parameter selection and variation. In the second part, the automated selection of several PSO variants within the algorithmic convergence is discussed.

In order to understand other self-adaptation algorithms in biological optimization such as Bee Algorithms, Evolutionary Programming, Genetic Programming a representative feature analysis is performed to extract relevant ideas from other self-optimizing nature inspired approaches.

In most of the variants of PSO, the parameter definition was a manual and static process. For example, $c_1\varphi_1$ and $c_2\varphi_2$ parameters were “tuned” before the optimization actually has started. In addition, the number of particles during an optimum search was selected as a constant number. Other examples of static setting are the way the particles are interconnected. Often a ring topology between particles is the default choice as a “neighboring topology”.

In early versions of PSO, this seemed to be sufficient to experiment manually with the parameter selection. Later PSO researchers then started to vary parameter during the iterative search process, based on above-mentioned parameters but not limited to.

2.2.2.1 Dynamic Variation of PSO topology

The main objective of MSAPSO is to act as a kind of a universal algorithm with “good to very good performance” characteristics. The algorithm shall work on a very large set of different kind of benchmark problems without having the need to think about the parameter settings anymore.

Although performance is important, the MSAPSO will not focus on the last percentage of convergence tuning. Therefore, it is essential to find a communication structure, which performs well on many optimization problems. In Kennedy and Mendes (2002, p. 1672), the authors theorize that:

“Populations with fewer connections might perform better on highly multimodal problems, while highly interconnected populations would be better for unimodal problems”.

The study shows in different tests that greater connectivity speeds up convergence, nevertheless, it does not tend to improve the population’s ability to discover global optima. In multimodal problems, faster convergence is not necessarily a good idea, thus fewer neighbor connections would be better for complex multi-modal problems. In unimodal problems, the opposite is valid.

The statement above shows that the dynamic variation of PSO topology can be a nice research area from a performance tuning perspective, but as remarked it is not in scope for this dissertation.

Instead, the MSAPSO needs a topology structure, which performs on many problems. The recommended PSO topology structure also confirmed in (Olsson A. e., 2011, p. 228) is a “von Neumann Structure”. Therefore, this is our choice for the communication structure for MSAPSO.

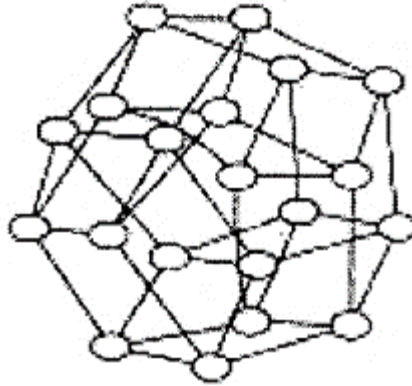


Figure 6: PSO optimal particle’s neighborhood structure

2.2.2.2 Adaptive PSO (APSO)

An early self-adaption variant of PSO named “Adaptive Particle Swarm Optimization (APSO) using information about Global Best” describes the aspect of dynamic tuning of parameters (Teruyoshi Yamaguchi, 2007). Due to the wide ranges of applicability to problems (unimodal, polymodal, in differentiability, etc.) the parameters of course on one hand adapts to these specific problems on the other hand, it would have been useful to automatically tune and adapt them to the optimization problems investigated. In the past, tuning of parameters was mostly based on empiric analytics. Later on, scheduled and planned changes of parameters during the iterations of PSO optimization was another

approach researchers looked into. Autonomous rules of parameter adaption were not really in scope of former PSO research. In 2007, the inventors of Adaptive Particle Swarm Optimization worked out an algorithm, which uses information from “Gbest”, in order to tune parameters during the search process dynamically. The authors first give a definition of adaptability of optimization algorithms in general and describe this as the “self-adjustment of the internal structure of the algorithm in accordance to rules, so it can perform better than before”.

The parameter tuning options they outlined were:

- Fixed parameter setting: Parameter is set before the algorithm execution
- Scheduled tuning rule: A predefined rule is used for the parameter tuning
- Iterative tuning rule: Parameter tuning is performed on one or more previously matched conditions
- Adaptive tuning rule: Parameter tuning is based on a “static search policy” obtained from information which is outside of the internal algorithm
- Autonomous tuning rule: Parameter are optimized based on a “dynamic search policy” with the intent to improve the actual search policy in use, based on obtained information which is available inside the algorithm

APSO algorithm describes the concept of the “tuning capacity of the algorithm”. It examines how the adaption of these parameters relates to the success of the search. In the existent case, the parameters that are tuned on a particle level are c_1 and c_2 (cognition and social weighting factors) and APSO evaluates their correlation to the update frequency

of Gbest. It is assumed that the more often Gbest is updated the more successful the search will be.

The measurement of the success of the algorithm with regard to the parameter tuning is based on the following criteria's due to the authors:

- Gbest update probability
- Gbest update frequency
- Gbest update frequency & Gbest improvements

Where GBest improvements is the difference between the best-determined functional value of Gbest at time t (iteration) minus the functional value of Gbest at time $t = 0$.

The prognosis of APSO is, if there is a relationship between parameter tuning ($c1$, $c2$) and GBest "success" that this is useful for adaptively controlling the PSO algorithm in order to perform a stable and fast convergence.

Within some numerical experiment, they analyzed the following:

- $c1$ relation to the update frequency of Gbest
- $c2$ relation to the update frequency of Gbest
- Gbest update frequency relation to improvements of Gbest

As a result, and based on different benchmark functions applied, there is a certain value range of c_1 , c_2 respectively where there is a higher update frequency of GBest. Moreover, based on the nature of the tested benchmark problems there was no significant value difference of c_1 , c_2 . The authors furthermore describe also the significant correlation of the update frequency of Gbest and the improvements of Gbest, where this seems to be an obvious relation.

All the above indicate that an adaptive search algorithm makes sense that uses the principle of parameter tuning and update frequency of Gbest. The self-improvement of the APSO algorithm implies the following principle:

“A particle that holds a c_1 , c_2 parameter with a lower update frequency with regard to Gbest, very likely will not contribute to the success of the search”.

The improvement equations and the definitions for the parameters are as following:

- k – iteration step
- α_i , $i = \{1, 2, \dots, m\}$, – the step width of the tunable parameter
- T_{\max} – total number of iterations when calculating c_1 , c_2 parameters
- c_{1_i} , $i = \{1, 2, \dots, m\}$, and c_{2_i} , $i = \{1, 2, \dots, m\}$, – social tuning for each Particle
- $cbest1$ – best cognitive parameter holds by particle i
- $cbest2$ – best social parameter holds by particle i
- improvement equation for c_1 of particle i at iteration k

$$c1_i^{k+1} = c1_i^k + \alpha_i^k (cbest1^k - c1_i^k), \quad i = \{1, 2, \dots, m\}$$

Equation 17: APSO – Cognitive factor adaption

- improvement equation for c2 of particle i at iteration k

$$c2_i^{k+1} = c2_i^k + \alpha_i^k (cbest2^k - c2_i^k), \quad i = \{1, 2, \dots, m\}$$

Equation 18: APSO – Social factor adaption in APSO

- step width when calculating c1, c2

$$\alpha_i^k = \begin{cases} 0, \\ \frac{1}{T_{max}} \end{cases}$$

Equation 19: APSO – Step width when calculating c1, c2

After initialization of the algorithm c1, c2 is continuously determined due to the above equations with regard to the cbest1, cbest2 values. The c1, c2 values are then put into the velocity and position update equations for the next iteration for each particle. Finally, the algorithm checks, whether there is a functional value improvement in iteration k + 1 with the new adapted c1, c2 values compared to the actual personal best functional value at iteration k of particle i. In a last step, the best personal value of all particles is searched and then set to the global best value at iteration k + 1. The algorithm stops when k = T_{max}.

The authors confirmed that the continuous parameter adaption of c_1 , c_2 maintained good search capabilities and therefore have a good robustness in a wide range of problems.

2.2.2.3 Adaptive Learning PSO (ALPSO I)

Researchers from the University of Leicester introduced one of the first entry points into self-adaption in 2009. In their research paper they describe the principles of the so-called ALPSO I algorithm (Chang Li, 2009). The learning strategy they outline separates the algorithm into four essential parts, which gathers information coming from

- a. The own historical best position (Pbest)
- b. The closest neighbor (Pbest from closest neighbor)
- c. A random position around itself
- d. The global best one (Gbest)

The basic learning principle in ALPSO is based upon, that each particle is able to change their individual search behavior and strategy. ALPSO I realize four operators, which do have different characteristics in different search spaces. Based on a selection ratio the operators are in use during the search process. For the learning of the Gbest particle only information from other improved particles is used.

In order to keep diversity in ALPSO I, the cognitive (Lbest) part separates from the social (Gbest) part. Nevertheless, the point where the switch over between learning from Lbest or GBest has to happen is still a hard problem to solve.

According to the four different information sources ALPSO I do have four different update equations or operators, which reflect the different learning strategies available to a particle.

ALPSO describes the learning strategies as follows:

Definitions:

x_i^d – Position of particle i in dimension d

v_i^d – Velocity of particle i in dimension d

r_i^d – Random position of particle i in dimension d (around itself)

$pbest_i^d$ – Personal best position of particle i in dimension d

$pbest_{i-nearest}^d$ – Personal best position of closest neighbor to particle i in dimension d

w – Inertia weight

η – Acceleration constant

v_{avg}^d – Average velocity of all particles in dimension d

$N(0,1)$ – Random number drawn from a normal distribution (mean = 0, variance = 1)

The four Update Operators:

$$v_i^d = wv_i^d + \eta r_i^d (pbest_i^d - x_i^d)$$

Equation 20: ALPSO I – Velocity update of particle i of the own (best) position

$$v_i^d = wv_i^d + \eta r_i^d (pbest_{i-nearest}^d - x_i^d)$$

Equation 21: ALPSO I – Velocity update of particle i related to Pbest closest neighbor

$$x_i^d = x_i^d + v_{avg}^d N(0,1)$$

Equation 22: ALPSO I – Position of particle i

$$v_i^d = wv_i^d + \eta r_i^d (gbest^d - x_i^d)$$

Equation 23: ALPSO I – Velocity update of particle i related to Gbest position

The authors describe that based on the four operator equations, the particles do have a chance to behave in four different ways and therefore increase the probability of a particle i to move to a more promising position. As the particle still do not know how the region around him looks like, the researchers introduce a learning concept on a particle level such that the particle is able to apply the right operator equation.

The learning is dependent on two factors:

- A so-called progress value of the appropriate operator equation at iteration t . This value memorizes how many particles and child particles were produced by which operator equation in the past
- A reward value which is a significance factor per operator equation, which memorizes the relative importance of the operator equation at time t . Reward is the increase in relative progress compared to others

The selection ratio (selection probability) of the operator equation for the next iteration denotes then as following:

“A reward value of the individual operator equation in relation to the overall reward”.

The critics stem from the fact that there is no concept of taking dependency between subsequent iterations into consideration, as there is a certain likelihood and tendency for an operator equation to be used by a particle. As it is designed in ALPSO I, it looks like that they assume simply independence between the iterations. Also, it is still not understood how the convergence characteristics of this PSO variant looks like to better understand and set optimal parameter settings of the different update equation.

2.2.2.4 Adaptive Learning PSO (ALPSO II)

A further improved version of ALPSO I can be found in (Chang & Yang, 2010). The authors mention that they were able to improve the performance of ALPSO I in ALPSO II on multimodal problems. The major improvements are on the following topics:

- Adding particle's status monitoring mechanism
- Controlling the number of particles
- Learn from the global best position Gbest, and the replacement of two of the four learning operators used in ALPSO I

They claim in their tests that ALPSO II outperforms ALPSO I, but there was no study done which compares the improvement to other Non-PSO algorithms, so it cannot be evaluated if it is an improvement without tradeoffs in other areas.

2.2.2.5 Self-adaptive learning based PSO (SLPSO I)

An initial version of the “Self-adaptive & Learning based PSO” (SLPSO I) (Yu Wang, 2010) was described from a research team at University Anhui China. The new approach describes the principles of SLPSO in his first version. It leverages four simultaneous search strategies. An underlying probability model rates the success rate of the applied search strategies such that, it determines the strategy in time, which strategy is likely the best to update a particle velocity in the search space. In an iterative way and based on a learning rate the execution probabilities of the four-update mechanism are updated. The algorithm embeds also the history of previous optimization iterations in the update mechanism. The

basic idea described in the paper, is that the algorithm combines multiple particle update strategies at different stages of the PSO execution.

The characteristics of the different update strategies summarizes as following:

- Update strategy one – CLPSO (Comprehensive Learning PSO): which has very good exploration capabilities especially when handling multimodal problems, but has low success within unimodal problems
- Update strategy two – PSO-CL-Pbest (PSO-Comprehensive Learning-Pbest) which has worse exploration ability than one, but do have faster convergence behavior than update strategy one.
- Update strategy three – DbV (Difference based velocity update strategy), which uses differential information between the particles. The algorithm is helpful in rotated and unimodal problems.
- Update strategy four – EbV (Estimation based velocity update), which shows very good performance and convergence in unimodal problems.

Initially in SLPSO, there is an assignment of an execution probability. Because there are four strategies to select from it is set to $\frac{1}{4}$ at the beginning: The algorithm denotes the following definitions.

- proSTRi – Probability of strategy i to be selected for the particle velocity update, where $i = \{1...4\}$

- proSTR'_j – Temporal execution probability of strategies j which generates the particle
- proSTR_j – Weighted execution probability of strategies j which generates the particle
- G_s – Fixed number of generations
- α – Learning rate of the algorithm, used to control the update proportion
- ps – Population size of particles
- S_i – Accumulator for strategy i
- S_j – Accumulator for particle j
- W_j – Assigned weight to the particle j

During each iteration, particles are “ordered” based on their fitness values and then each particle j is assigned a weight (logarithmic weighted average) with the formula:

$$W_j = \frac{\log(ps-j+1)}{\log(1)+ \dots + \log(ps)}$$

where,

$$j = [1 \dots ps]$$

Equation 24: SLPSO I – Assigned logarithmic weight to particle

As a next step, the weight is added to so-called accumulators S_j . After a number of

generation G_s the SLPSO algorithm does the following updates on the execution probabilities of strategy j , which makes the particle(s) at a certain iteration.

$$\text{proSTR}'_j = (1 - \alpha) \text{proSTR}_j + \alpha \frac{S_j}{G_s}$$

Equation 25: SLPSO I – Temporal execution probability of strategy j

$$\text{proSTR}_j = \frac{\text{proSTR}'_j}{(\text{proSTR}'_1 + \text{proSTR}'_2 + \text{proSTR}'_3 + \text{proSTR}'_4)}$$

Equation 26: SLPSO I – Relative execution probability of strategy j

The goal of the algorithm is to find the strategy j that makes the fitness of all particles. The strategy is selected based on an accumulator value S_j in a certain iteration.

$$S_j = S_j + W_j$$

Equation 27: SLPSO I – Accumulated importance of strategy j over all particles

The higher the accumulator, which expresses the most successful strategy j , the more likely is the relevance of the strategy j .

SLPSO looks from his core idea very similar as ALPSO I in the original version. Although executed in a different way it looks like a related concept to ALPSO I and II. Also in this

concept, the strategy limits at the end of the day to the four proposed update strategies. The behavior of particles restricts to that model and they cannot choose alternate behavioral models. In addition, there is not really a way to guess how the area around the individual particle looks like. Multimodal and unimodal regions can change quickly in a problem so the learning may simply take too long to make good just-in-time decisions for the particle.

2.2.2.6 Self-Learning PSO for Global Optimization (SLPSO II)

In (Changhe Li, 2012) the Self-Learning Swarm Optimizer for Global Optimization (SLPSO II) describes a new and evolved aspect with regard to self-adaption of particle swarms, which is the switch of PSO operators to adapt to the structure of the problem surface. One of the motivations is to deal with very complex multimodal functions, which do have a significant number of local minima's. In the today's and past invented PSO approaches there has been just one single learning behavior of a swarm or a particle, which ignore the fact, that a complex problem to solve may have various shapes and problem surface structures embedded. It seems to be obvious that it should be possible for a particle or a swarm to encourage varying learning strategies.

The SLPSO II described here, is a natural evolution of the Advanced Learning Particle Swarm Optimization (ALPSO I/II). The basic concept of ALPSO and SLPSO is that both can select between four learning strategies, where a particle can use those as a source to

optimize the exploration and exploitation phase. SLPSO introduces five new features⁹ compared to ALPSO, with the purpose to improve convergence performance and stability.

The basic idea behind SLPSO II stems from the fact, that when a particle learns from Gbest and Lbest model at the same time, that this may combine disadvantages from both models (Gbest – early stuck in local minima, Lbest – slower convergence). Remember that Gbest and Lbest differ from each other such that Gbest (social part) interconnects every particle with every other particle in a swarm, while Lbest (cognitive part) just have a few neighborhoods. One idea to avoid the above problem of combining disadvantages is actually to separate the cognitive (individual) from the social (group) learning. By that, the individual particles could focus on the exploitation (Gbest) component or exploration (Lbest) component in a certain iteration of the search or vice versa. The problem, which appears in SLPSO II, is the choice of the right timing to select one or the other search strategy and the right moment for it. Based on the “division of labor” idea the particles can play different roles within the search process:

- Converging to a global best particle
- Exploiting the personal best position
- Exploring new promising areas
- Jumping out of local optima

⁹ cf, Changhe et al. (2012, June). A self-Learning Particle Swarm Optimizer for Global Optimization Problems (SLPSO). IEEE Transaction on Systems, Man and Cybernetics - Part B: Cybernetics, Vol 42, No. 3, page 1

The four strategies are implemented as varying operators, and each particle has then the option to deal independently with different parts of the problem surface.

The work in SLPSO II also describes an Adaptive Learning Mechanism, which makes sure that a particle uses the best operator in time during the search process. The adaption mechanism itself founds on a success rate of each individual operator. The success rate assumes that a “successful” operator may also be successful in the future. The success rate of the operator itself is expressed in a so-called “selection ratio”. The selection ratio is composed of a combination of:

- Current success ratio
- Offspring fitness
- Previous selection ratio

The higher the selection ratio, the more likely is the usage of the operator by the particle. Over time, the particle gradually uses the best operator. Although SLPSO II is an improvement with regard to selection of operator equations, it contains no concept of future selection ratios, because if an operator equation is at a low success level it might make sense to take the appropriate operation out of the selection process. The general doubt that can be raised is that success on an operator level is not a scaling self-adaption method, rather than using a success metric of particle on an individual level. As there are probably infinite ways to create operators, it is a strong assumption that just four update strategies can represent a broad range of search needs for a broad range of problems.

2.2.3 Self-Optimization in alternate bio-inspired algorithm's

PSO is not the only research area with regard to swarm algorithms. In the meantime, many others swarm and bio-inspired approaches evolved such as:

- Artificial Bee Colony Algorithm (ABCA)
- Artificial Ant Colony Algorithms (AACA)
- Memetic Firefly Algorithm FFA (MFFA)
- Glowworm Algorithm (GWA)
- Cuckoo Search Algorithm (CSA)
- Bat Artificial Algorithm (BAA)
- Cultural Algorithm (CA)
- Hunting Search Algorithm (HSA)
- Etc.

In general, these swarm algorithms mimic behavior of social animal groups and colonies in order to apply them to complex optimization problems. The members of these groups communicate based on simple rules either directly or indirectly with each other. This interaction triggers emergent patterns in the system, which makes the colony to better react to the environment and therefore perform better on problems apparent to the group.

Out of these algorithms, two of them as representatives of other “bio-inspired algorithms” are more closely evaluated. The intent is not only to understand their algorithmic approach, but also what kind of approaches they have with regard to self-adaption or self-organization in their model compared to the PSO approach.

The selection of the investigated algorithms is done based on the self-adaption capabilities of these bio-inspired algorithms, which means that they have just a few parameters to define and that these parameters do have potential to be self-adaptive during the algorithmic runtime.

2.2.3.1 BAT Artificial Algorithm (BAA)

BAA was originally created by (Yang, 2010). Bats uses sonar and echolocation to find prey and avoid barriers. They do this by sending a very loud sound pulse at certain frequencies and wavelength and then listen to the corresponding echo.

The BAT strategy can be theorized as follows:

- All bats use echolocation to determine the distance of objects. At the same time, they are able to differentiate between objects (e.g. prey or obstacles)
- Bats randomly fly with velocity v_i to position x_i at a fixed frequency f_{\min} . They are able to adapt the wavelength λ and loudness A^0 of the send pulses.
- Although loudness can vary significantly, it is assumed that loudness decreases from large value A^0 to a minimum value A_{\min} .

The bat colony is set into the search room with an initial position x_i and velocity v_i (similar to the PSO particles). Each bat is equal to a candidate solution $x_i = \{x_1, x_2, \dots, x_n\}$ of the optimization problem.

In addition, m design variables are defined with

$\{x\} = \{x_1, x_2, \dots, x_m\}^T$ and an objective function $f(x)$.

At the same time the pulse rate r_i , the loudness A_i and the frequency f_i at x_i is initialized.

New solutions at time step t are created due to the following update equations:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta$$

Equation 28: BAT Algorithm – Frequency update of bat at position x_i

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)f_i$$

Equation 29: BAT Algorithm – Velocity update of bat at iteration $t+1$

$$x_i^t = x_i^{t-1} + v_i^t$$

Equation 30: BAT Algorithm – Position update of bat at iteration $t+1$

β – is a random number from a uniform distribution $[0,1]$, x^* is the current best bat under n bats. If a generated random number is $>$ than pulse rate r_i , then the algorithm looks for a solution among the best solutions. Furthermore, the algorithm generates a local solution around the best ones. In case a random generated number is $>$ than the bats loudness A_i and the functional value is smaller than the functional value of the actual best bat x^* then

it is time to accept new solutions. Pulse rate i of the bat increases accordingly and the loudness decreases according to a cooling mechanism. In order to find the new best bat, all the best bats need to be ranked to find the new x^* .

With regard to self-adaption, the BAT algorithm categorizes as a self-adaptive parameterization during the algorithmic runtime. The parameters adapted are the loudness and the pulse rate of a bat, which then compares against generated random numbers. They are a representation for exploitation and exploration when we talk in PSO terms. The overall falling loudness when getting to optimal solutions looks similar like the inertia weight factor which also cools down over time, indicating the exploitation phase, whereas higher loudness is comparable with exploration state. The pulse rate instead is comparable to the cognition factor in the PSO, which increases when an interesting place is found by a particle. Overall there are a couple of similarities in the BAT algorithm compared to PSO, the major difference lies in the concept of the multiple G_{best} 's in the bat algorithm, whereas in the standard PSO there is just one.

2.2.3.2 Memetic Firefly Algorithm (MFFA)

The second example is the Memetic Firefly Algorithm (MFFA). In Yang et al. (Swarm Intelligence and Bio-Inspired Applications, 2013) the authors describe the criteria's how the firefly algorithm works. The major principle is that of a flashing light and the appropriate light intensity I_L . The light intensity shrinks when a firefly moves away from source r according to the following relation:

$$I_L \approx \frac{1}{r^2}$$

Equation 31: MFFA – Relation Light Intensity and distance r from source

The light intensity is a synonym of the fitness function of a candidate solution to a problem to be optimized. So, it can be written such that, $I_L \approx f(s)$, where $s = S(x)$ is the candidate solution.

The following characteristics must hold for the firefly algorithm:

- Fireflies are unisex
- Attractiveness β is proportional to the light intensity (relative measure from beholders view)
- Light intensity is determined by the problem landscape (absolute emitted light)

The exact definition of light intensity is as follows:

$$I_L(r) = I_{L_0} e^{-\gamma r^2}$$

Equation 32: MFFA – Light intensity varied by distance r

Where I_{L_0} is the light intensity at the source and λ a fixed light absorption coefficient.

Similar to the light intensity also the attractiveness β depends on distance r according to the following formula:

$$\beta(r) = \beta_0 e^{-\gamma r^k}, \text{ for } k \geq 1$$

Equation 33: MFFA – Attractiveness from beholders view varied by distance r

The distance between any pair of fireflies (i, j) is then calculated according the Euclidian Distance formula with:

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2}$$

Equation 34: MFFA – Euclidian distance between a pair of fireflies

Finally each firefly i will change its position and move to more interesting firefly j based on the following update equation:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha N_i(0,1)$$

Equation 35: MFFA – Final update equation of a firefly i

The first portion of the equation describes the actual position, the second the attractiveness and the third part corresponds to the random movement of the firefly.

It is only similar to PSO with regard to the position update and the random movement of the firefly. Apart from that, it also differs such that MFFA has no conception of social and cognition between fireflies. It just employs the approach of attractiveness at varying strength between fireflies, which is simpler compared to the PSO algorithm.

Because the efficiency of MFFA is strongly dependent on the control parameters α , β and γ , the initial setting of these parameters can turn into a problem during the runtime of the algorithm.

The concept is as following:

The three control parameters α , β and γ are coded in the following form as a vector into the firefly's gene. In other words, it describes the firefly in his actual state including the actual parameter values and the appropriate standard deviations of the parameter values:

$$x_i^{(t)} = \left(x_{i0}^{(t)}, \dots, x_{in}^{(t)}; \alpha^{(t)}; \sigma_0^{(t)}; \beta^{(t)}; \sigma_1^{(t)}; \lambda^{(t)}; \sigma_{12}^{(t)} \right), \text{ for } i = 1 \dots N$$

Equation 36: MFFA – Genotype of a firefly i as a base for self-adaption

MFFA then mutates the self-adaptive control parameters according to specific rules, which also implies randomness from a Gaussian distribution. In every iteration, MFFA then calculates the fitness of the fireflies according to this genotype.

There are no similarities to existing PSO's with regard to self-adaption of control parameters. It is a new concept, which pretty much characterizes the firefly according its success in the problem landscape. This can be an interesting idea for application in the MSAPSO as well, although in the actual version there is no similarity between MFFA and MSAPSO.

2.2.4 Dynamics in Social Systems - Bandura's Learning Theory

In the social cognitive theory, (Bandura, 1986) describes the concept of reciprocal determinism. In this model of reciprocal causation, he states that social behavior and cognition and other environmental factors do have influence on each other in a bidirectional way.

Furthermore, he outlines that reciprocal causation does not necessarily mean that the different sources of influence (factors) needs to be of equal strength. He describes in an example that a person (individual) is influenced by others behavior. On the other side, the individual is also responsive to changes in environmental conditions. Therefore, environmental conditions would trigger changes in behavior in an individual and in groups where the individual is a part of.

This principle of causal reciprocal behavior is one inspiration source for the MSAPSO to vary the cognition and social parameter. In the MSAPSO algorithm, the variation of the social and cognition strength can have an influence on the algorithm such that, in certain cases it might be better to have a stronger emphasis on collaboration (social) rather than cognitive behavior. An example is when the to-be-solved problem gets complicated (multimodal problem surface) then it is worthwhile to better collaborate whereas in simple problem benchmarks the opposite is true.

2.3 Summary of Findings

In section two, various research branches in PSO were reviewed. One of the key takeaways is that improvements in either parameters or superior update equations are often “bought” with introduction of new parameters and other fields of complexity. Although in certain areas there is progress, the aforementioned approaches are somehow limited by nature. In the new research branch, self-adaptive PSO’s researchers try to eliminate these issues. Therefore, they look for new self-adaptive methods, with the intent of automatic PSO parameter or update equation adaption. The main goal is that this “self-adaption” lead to a better representation of the surrounding and actual search space the particle resides in. Related other bio-inspired swarm algorithm does follow similar concepts in the area of adaptive parameters.

A major difference, which can be seen, is that the self-adaptive algorithmic approaches often employ the concept of a gene or genotype, which characterizes the individual parameters iteration by iteration. Based on this, more optimal parameter sets can be determined. In the area of social and cognitive theories new concepts such as bidirectional learning at varying strengths is an interesting direction, which primarily influences the area of dynamic update equations. Albert Bandura and Kurt Levin as social and cognitive researchers can be an interesting innovation source, when defining a new concept of self-adaption. Synergies between different self-adaption variants is a logical consequence for a “to be designed” algorithm. In any case a very good understanding of the convergence parameter room is necessary to get understanding how automatic parameter settings in a PSO can be done.

In the next sections, the different ideas, which were discussed previously are pulled together. In order to come up with a new self-adaptive PSO approach, which works on multiple levels (self-adapting parameters, dynamic escape strategies, dynamic update equations, dynamic use of probability distributions for randomization of the particles), the discussed concepts in this chapter are taken into consideration.

3 CONVERGENCE FRAMEWORK OF THE MSAPSO

Multi Self Adaptive Particle Swarm Optimization (MSAPSO) will combine several new concepts (dynamic inertia weight, dynamic relation of inertia weight with the sum of social and cognition, dynamic use of random probability distributions) into an advanced multi self-adaptive PSO framework.

Furthermore, the new overall concept should employ the capability to adapt to the varying environments in the search space. Also, a new concept to dynamically escape potential local optima in high dimensional problem benchmarks is presented.

The key entry into all this is seen in the deep mathematical and graphical understanding of the convergence characteristics of MSAPSO so an appropriate approach can be designed.

More specifically MSAPSO contains the following elements and contributions:

- The in-depth understanding of the stochastic convergence behavior of MSAPSO as a foundation and the resulting stochastic convergence curves both for normal and uniform distribution.
- The understanding of the relation of the so-called order-1 and order 2 convergence zones, where order-1 describes the convergence behavior that is based on average values of the particle flights and the corresponding parameter settings of MSAPSO.

- In addition, order-2 convergence implies the variance and standard deviation of the particle flight and the associated parameter sets.
- Mathematical proof of the relation of order-1 and order-2 convergence zones and the order-2 collapse into order-1 convergence zone under certain conditions.
- A self-optimization formula which describes the relation of the PSO parameters social, cognition and inertia weight along the set of convergence curves.
- A method to use different probability distributions to control the exploitation and exploration characteristics of the algorithm.
- A generic stability formula which describes the relation between social, cognition and inertia weight independent of the applied probability distribution and the to-be-solved benchmark problem.
- Mathematical proof that the found stability line is a natural property of the set of convergence curves possible.
- A new escape local minima strategy for low and high dimensional problems.

3.1 Self-Adaptive behavior of MSAPSO

MSAPSO is by nature a stochastic PSO variant, which “detects” an optimal inertia weight value based on a success model of the particle and/or swarm and then fits an corresponding sum of social and cognition value to it, by some later explained equation. As the parameter sets are dynamically changed from iteration to iteration, this basically means that there are multiple dynamical systems at the same time, more precisely one dynamical system per iteration. The basic behavior of MSAPSO is that of a stochastic process, which means that the position and the velocity of a particle is stochastically dependent from previous

iterations. As the PSO uses random distributions such as normal, and uniform distribution to vary the position of the particle during the convergence phase by default MSAPSO implies stochasticity in the algorithm. This design of the self-adaptive MSAPSO algorithm requires an in depth understanding of the respective convergence model.

3.2 General convergence analysis PSO and considerations

In Tian (2013), a review of different ways on how to do a convergence analysis can be found. In the context of MSAPSO the following methods are theoretically available:

- Use of differential equations
- Matrix form calculations solving characteristic polynomial
- Difference equations to reduce the PSO system into recursive equations
- Z-Transformation
- Others methods, specific to an individual algorithm

All of the methods have the goal to consider the algorithm in a convergence situation, more precisely when the particles or swarm velocity tends to zero. This is simply the case in the so-called stagnation phase. Formally this is the situation when the following condition is true.

$$x_i = \frac{(\varphi_1 \vec{P}_i + \varphi_2 \vec{P}_g)}{\varphi_1 + \varphi_2}$$

$$v_i = 0$$

Equation 37: Stagnation condition of the MSAPSO

where x_i is the oscillating position of the particle i towards the potential convergence point (equilibrium point) and v_i is the corresponding velocity of the particle i . In the literature, it can be seen that there are many simplifications made, before the appropriate mathematical convergence analysis is performed. For example, the following assumptions are being made:

- Reduction of the dimensionality of the PSO
- Social, cognition and inertia weight are assumed as constants
- Exclusions of the influence of the probability distributions applied

It is then relatively easy to study the convergence of PSO in general under such rigid assumptions, but on the other hand it also has limits when studying detailed behavior of our MSAPSO algorithm. In order to dynamically calculate the relation of social, cognition and inertia weight, the previously made assumption need to be removed such that the real parametric changes and probability distributions of the convergence analysis are taken into consideration.

3.3 Assumptions for convergence analysis

The only assumption which is made is: that the particle swarm can be assumed in a stagnation situation by the end of the algorithmic run. As a consequence of that, when the particle swarm velocity tends to zero and has reached the final stagnation point it can be assumed that statistical independence is given with regard to the previous iteration, as

velocity and position of the particle swarm should be almost the same both in actual and previous situation.

Besides that, randomness is introduced reflected by the used probability distribution applied to the variation of social, cognition and inertia weight into the convergence analysis.

Then an order-1 and order-2 model is executed to find specific convergence lines for MSAPSO for a set of given probability distributions. In the case of initialization of the MSAPSO algorithm a uniform distribution is used as well as when a stuck-in-local-optima situation appears during the algorithmic convergence.

In all other cases (iterations) a normal distribution is used with an average value of $\frac{1}{2}$ and a standard deviation of $\frac{1}{2\sqrt{3}}$.

It is important to mention that the standard deviation parameter in the uniform distribution and the normal distribution do have the same value. The reason for that is explained in the later discussion.

3.4 Order-1 convergence analysis – Difference & Matrix Model

The order-1 convergence embeds the average values of the evaluated MSAPSO parameters (inertia weight, social and cognition) into the convergence analysis and by that the average values of the particle flight. With the convergence analysis, it is possible to understand the MSAPSO system behavior in certain situations such as stagnation and or general convergence.

The base equations for the order-1 convergence analysis are denoted as following:

$$x_i^{k+1} = x_i^k + v_i^{k+1}$$

$$v_i^{k+1} = w^k v_i^k + \varphi_1^k (g^k - x_i^k) + \varphi_2^k (l_i^k - x_i^k)$$

Equation 38: Base MSAPSO equations for convergence analysis

Where $\varphi_1^k = r_1^k c_1^k$ and $\varphi_2^k = r_2^k c_2^k$ are the respective random distributed social (c_2^k) and cognitive values (c_1^k). The actual iteration is named k. The parameter r represents a random number drawn from a chosen probability distribution. Then the second order stochastic difference equation can be defined with a recursive approach of the previous equations such that:

$$x_i^{k+1} = x_i^k + w^k v_i^k + \varphi_1^k (g^k - x_i^k) + \varphi_2^k (l_i^k - x_i^k)$$

$$x_i^{k+1} = x_i^k + w^k v_i^k + \varphi_1^k g^k - \varphi_1^k x_i^k + \varphi_2^k l_i^k - \varphi_2^k x_i^k$$

$$x_i^{k+1} = x_i^k + w^k v_i^k + \varphi_1^k g^k + \varphi_2^k l_i^k - \varphi_1^k x_i^k - \varphi_2^k x_i^k$$

$$x_i^{k+1} = x_i^k + w^k v_i^k + \varphi_1^k g^k + \varphi_2^k l_i^k - (\varphi_1^k x_i^k + \varphi_2^k x_i^k)$$

With $\varphi^k = \varphi_1^k + \varphi_2^k$ it is possible to further reduce the equation to

$$x_i^{k+1} = x_i^k + w^k v_i^k + \varphi_1^k g^k + \varphi_2^k l_i^k - \varphi^k x_i^k$$

Resorting and factorize the term with

$$x_i^{k+1} = x_i^k - \varphi^k x_i^k + w^k v_i^k + \varphi_1^k g^k + \varphi_2^k l_i^k$$

$$x_i^{k+1} = x_i^k (1 - \varphi^k) + w^k v_i^k + \varphi_1^k g^k + \varphi_2^k l_i^k$$

From previous base equation it can be concluded that $v_i^k = x_i^k - x_i^{k-1}$. This equation can then be substituted into the previous one and then via several steps transformed into the final recursive equation:

$$x_i^{k+1} = x_i^k (1 - \varphi^k) + w^k (x_i^k - x_i^{k-1}) + \varphi_1^k g^k + \varphi_2^k l_i^k$$

$$x_i^{k+1} = x_i^k (1 - \varphi^k) + w^k x_i^k - w^k x_i^{k-1} + \varphi_1^k g^k + \varphi_2^k l_i^k$$

$$x_i^{k+1} = x_i^k (1 - \varphi^k - w^k) - w^k x_i^{k-1} + \varphi_1^k g^k + \varphi_2^k l_i^k$$

Equation 39: Base MSAPSO recursive equations

For the reason of simplification, equation parameters are replaced with the following terms:

$$A = 1 - \varphi^k - w^k$$

$$B = -w^k$$

$$C = \varphi_1^k g^k + \varphi_2^k l_i^k$$

Equation 40: Simplified terms of the recursive order-1 equation

The mean trajectories of the particle swarm can be calculated by the introduction of the expectation values and then apply it to both sides of the respective recursive equation, which results into the following term.

$$E(x_i^{k+1}) = E(x_i^k) E(A) + E(x_i^{k-1})E(B) + E(C)$$

Equation 41: Expectation values of MSAPSO particles position

In order to understand the dynamics of this equation this equation is transformed into the following dynamical system form.

$$\begin{pmatrix} E(x_i^{k+1}) \\ E(x_i^k) \end{pmatrix} = A_{\text{order-1}} \begin{pmatrix} E(x_i^k) \\ E(x_i^{k-1}) \end{pmatrix} + C_{\text{order-1}}$$

$$\text{with } A_{\text{order-1}} = \begin{pmatrix} E(A) & E(B) \\ 1 & 0 \end{pmatrix} \text{ and } C_{\text{order-1}} = \begin{pmatrix} E(C) \\ 0 \end{pmatrix}$$

Equation 42: Dynamical Form of the MSAPSO order-1 convergence

where $\begin{pmatrix} E(x_i^{k+1}) \\ E(x_i^k) \end{pmatrix}$ equals the condition vector of the particle in the next iteration, $\begin{pmatrix} E(x_i^k) \\ E(x_i^{k-1}) \end{pmatrix}$ corresponds with the condition vector in the previous iteration, $A_{\text{order}-1}$ describes the matrix of the dynamical system and $C_{\text{order}-1}$ is the outer influence matrix of the dynamical system.

The constants in the system are the following terms $E(A)$, $E(B)$ and $E(C)$. Similar to Equation 40: Simplified terms of the recursive order-1 equation it is possible to write:

$$E(A) = 1 - E(\varphi^k) - E(w^k)$$

$$E(B) = - E(w^k)$$

$$E(C) = E(\varphi_1^k)E(g^k) + E(\varphi_2^k)E(l_i^k)$$

The subscript k are dropped as all constants are related to iteration k so, a simplified form can be described as:

$$E(A) = 1 - E(\varphi) + E(w)$$

$$E(B) = - E(w)$$

$$E(C) = E(\varphi_1)E(g) + E(\varphi_2)E(l_i)$$

Expectation values are renamed again into the well-known terms for average values with:

$$E(w) = \mu_w$$

$$E(\varphi) = E(\varphi_1 + \varphi_2) = E(\varphi_1) + E(\varphi_2) = \mu_{\varphi_1} + \mu_{\varphi_2} = \mu_\varphi$$

As the particle flights are linear independent by the end of the convergence or in stagnation phase from the previous iteration the E(C) equation can be rewritten into:

$$E(C) = \mu_{\varphi_1} E(g) + \mu_{\varphi_2} E(l_i)$$

E(A), E(B) and E(C) is further renamed into:

$$E(A) = 1 + \mu_w - \mu_{\varphi}$$

$$E(B) = -\mu_w$$

$$E(C) = \mu_{\varphi_1} E(g) + \mu_{\varphi_2} E(l_i)$$

Equation 43: Expectation values μ_w, μ_{φ} of the dynamical order-1 convergence

Subsequently Equation 42: Dynamical Form of the MSAPSO order-1 convergence will turn into the following μ_w, μ_{φ} based form:

$$\begin{pmatrix} E(x_i^{k+1}) \\ E(x_i^k) \end{pmatrix} = \begin{pmatrix} 1+\mu_w - \mu_{\varphi} & -\mu_w \\ 1 & 0 \end{pmatrix} \begin{pmatrix} E(x_i^k) \\ E(x_i^{k-1}) \end{pmatrix} + \begin{pmatrix} \mu_{\varphi_1} E(g) + \mu_{\varphi_2} E(l_i) \\ 0 \end{pmatrix}$$

Equation 44: μ_w, μ_{φ} based form of the dynamical order-1 convergence

From the theory of dynamical systems, it is well known that a dynamical system exactly then converges if the absolute values of the eigenvalues of the matrix A are smaller than one. The eigenvalues of A can be calculated by the zeroing's of the characteristic polynomial. The characteristic polynomial X_A is defined as:

$$X_A = \det(A - \lambda E)$$

Equation 45: Characteristic Polynomial of the MSAPSO dynamical system

where A is the iteration matrix of the dynamical system, λ is the Eigenvalue and E is the identity matrix of the dynamical MSAPSO algorithm. With the determinant of this linear system it is possible to detect if there is a unique solution.

The eigenvalues of A are exactly the values, which define, whether a system converges or not. The zeroing's of the characteristic polynomial or in other words the eigenvalues of A can be determined with the following equation:

$$\det(A - \lambda E) = \det \left[\begin{pmatrix} 1+\mu_w - \mu_\phi & -\mu_w \\ 1 & 0 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right] = 0$$

$$\det \left[\begin{pmatrix} 1+\mu_w - \mu_\phi & -\mu_w \\ 1 & 0 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right] = 0$$

$$\det \left[\begin{pmatrix} ((1+\mu_w - \mu_\phi) - \lambda) & -\mu_w \\ 1 & -\lambda \end{pmatrix} \right] = 0$$

Equation 46: Determinant of the order-1 MSAPSO dynamical system

Due to the rule of Cramer the determinant can be turned into the following quadratic equation:

$$-\lambda(-\lambda + 1 + \mu_w - \mu_\varphi) - (-\mu_w) = 0$$

$$\lambda^2 - \lambda - \lambda\mu_w + \lambda\mu_\varphi + \mu_w = 0$$

$$\lambda^2 - \lambda(1 + \mu_w - \mu_\varphi) + \mu_w = 0$$

This quadratic equation can be solved with the following formula of $\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, where $a = 1$, $b = -(1 + \mu_w - \mu_\varphi)$ and $c = \mu_w$ in this case. Consequently, the eigenvalues of the system in the order-1 convergence case are:

$$\lambda_{1,2} = \frac{-(-(1 + \mu_w - \mu_\varphi)) \pm \sqrt{(1 + \mu_w - \mu_\varphi)^2 - 4\mu_w}}{2} = \frac{(1 + \mu_w - \mu_\varphi) \pm \sqrt{(1 + \mu_w - \mu_\varphi)^2 - 4\mu_w}}{2}$$

The root of the quadratic equation can be replaced with Υ , so finally the following eigenvalues of the order-1 system can be denoted.

$$\lambda_{1,2} = \frac{(1 + \mu_w - \mu_\varphi) \pm \Upsilon}{2}$$

$$\text{with } \Upsilon = \sqrt{(1 + \mu_w - \mu_\varphi)^2 - 4\mu_w}$$

Equation 47: Eigenvalues of the order-1 MSAPSO dynamical system

3.4.1 Final order-1 convergence room of MSAPSO

According to previously mentioned dynamical system theory, a system converges formally when, the $|\lambda| < 1$. Formally this is defined this as the spectral radius ρ of the iteration matrix A with the following order-1 convergence condition

$$\rho(A) = \max |\lambda_i| < 1$$

Where i is the respective eigenvalue calculated based on iteration matrix A .

For the case where $|\lambda| = 1$, the appropriate convergence line is defined with the recall of Equation 46: Determinant of the order-1 MSAPSO dynamical system, by plugging in λ with a value of one.

$$\det \left[\begin{pmatrix} (1 + \mu_w - \mu_\varphi) - \lambda & -\mu_w \\ 1 & -\lambda \end{pmatrix} \right] = 0$$

setting $\lambda = 1$, equals $-\lambda = -1$, which results into

$$\left| \begin{pmatrix} (1 + \mu_w - \mu_\varphi) - 1 & -\mu_w \\ 1 & -1 \end{pmatrix} \right| = 0$$

Again, the rule of Cramer is used, after multiplying out

$$\begin{aligned} & ((1 + \mu_w - \mu_\varphi) - 1)(-1) - (-\mu_w)(1) = (1 + \mu_w - \mu_\varphi) + 1 + \mu_w \\ & = 2 + 2\mu_w - \mu_\varphi = 0 \end{aligned}$$

The equation for order-1 convergence can be found for μ_ϕ with

$$\mu_\phi = 2 + 2\mu_w$$

Equation 48: Order-1 convergence limit MSAPSO dynamical system

The visualization of the order-1 convergence limit, is shown in the following graph:

3.4.2 Visualization of order-1 convergence

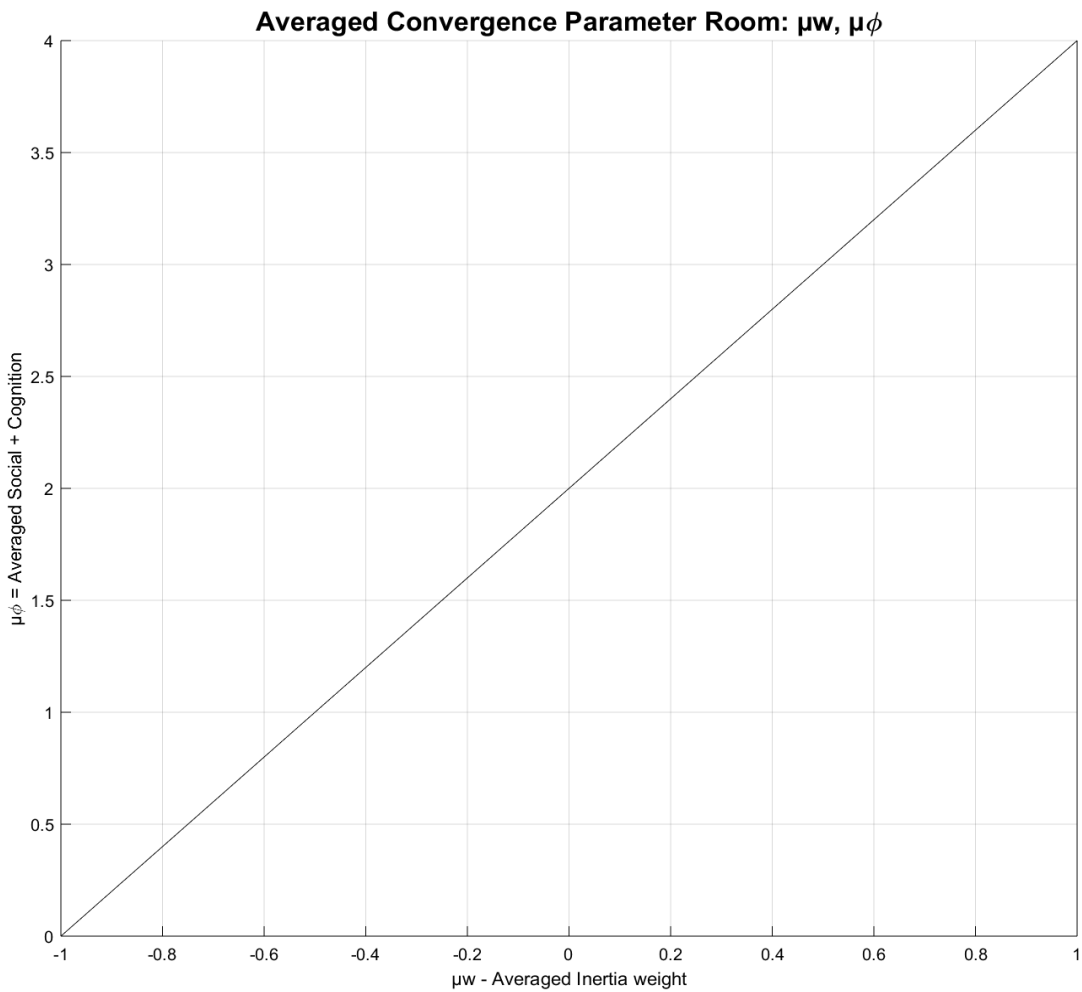


Figure 7: MSAPSO order-1 (μ_w , μ_ϕ) convergence room

3.4.3 Harmonic oscillation curve order-1 convergence

In order to determine the nature of the oscillation behavior of the MSAPSO system the discriminant of Equation 47: Eigenvalues of the order-1 MSAPSO dynamical system can be analyzed.

$$Y = \sqrt{(1 + \mu_w - \mu_\varphi)^2 - 4 \mu_w}$$

The discriminant is defined as:

$$Y = D = (1 + \mu_w - \mu_\varphi)^2 - 4 \mu_w$$

Equation 49: Discriminant of the order-1 MSAPSO dynamical system

when $D < 0$ the eigenvalues are getting complex and there are no real solutions in this case.

In this case $\lambda \in \mathbb{C}$. The so called harmonic oscillation curve is where the particles swing around the so-called equilibrium point which is defined with:

$$x_i = \frac{(\varphi_1 \vec{P}_i + \varphi_2 \vec{P}_g)}{\varphi_1 + \varphi_2}$$

Equation 50: Equilibrium point of particles

From Equation 37: Stagnation condition of the MSAPSO and by setting the discriminant to zero ($D = (1 + \mu_w - \mu_\varphi)^2 - 4 \mu_w = 0$) the harmonic curve equation can be derived in the following way:

$$(1 + \mu_w - \mu_\varphi)^2 - 4 \mu_w < 0$$

$$(1 + \mu_w - \mu_\varphi) (1 + \mu_w - \mu_\varphi) - 4 \mu_w < 0$$

$$(1 + \mu_w - \mu_\varphi + \mu_w + \mu_w^2 - \mu_w \mu_\varphi - \mu_\varphi - \mu_w \mu_\varphi + \mu_\varphi^2) - 4 \mu_w < 0$$

$$(1 + 2\mu_w - 2\mu_\varphi + \mu_w^2 - 2\mu_w \mu_\varphi + \mu_\varphi^2) - 4 \mu_w < 0$$

$$(1 - 2\mu_w - 2\mu_\varphi - 2\mu_w \mu_\varphi + \mu_w^2 + \mu_\varphi^2) < 0$$

$$\mu_w^2 + \mu_\varphi^2 - 2\mu_w \mu_\varphi - 2\mu_w - 2\mu_\varphi + 1 < 0$$

Equation 51: Harmonic oscillation curve around equilibrium point

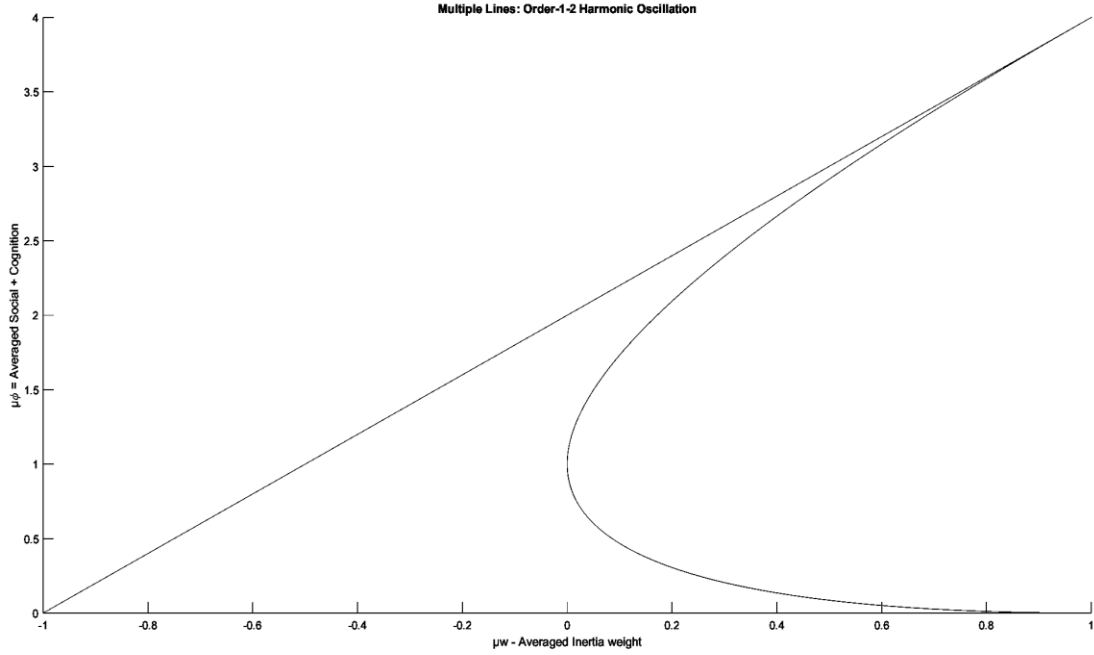


Figure 8: Harmonic oscillation area around equilibrium point

In the graph all value right to the harmonic oscillation curve are complex eigenvalues. The analysis of the 1-order stability proved that convergence of MSAPSO depends on the parameters on μ_w , μ_φ , such that the particles oscillate around the equilibrium point

$$x_i = \frac{(\varphi_1 \vec{P}_i + \varphi_2 \vec{P}_g)}{\varphi_1 + \varphi_2} , \text{ where the averaged convergence limit is defined as } \mu_\varphi = 2 + 2\mu_w .$$

In the next step, the higher statistical moments such as the variance are included into the further convergence analysis, in order to analyze what the variance of the trajectories of the particles means to the overall convergence of the MSAPSO algorithm.

The resulting order-2 convergence analysis is described in the next chapter.

3.5 Order-2 convergence Analysis based on Martinez approach

The foundation of the order-2 convergence analysis is described in two papers from (Esperanza García-Gonzalo, 2014) and (Poli, Riccardo, 2009). Based on their convergence methods and approaches the order-2 convergence behavior of MSAPSO can be analyzed. From there on certain convergence conditions and formulas for the specifics of the MSAPSO algorithm can be determined.

The MSAPSO uses two different probability distributions to randomize particle's positions. A uniform distribution for the initial distribution of the particle's positions in the search space and secondly for the case when the algorithm gets stuck into some local optima. A normal distribution with $N(\mu, \sigma)$ is used during the normal convergence phase of the algorithm.

The first step is the recall of the expressions and equations from the order-1 analysis. The variables $x_i^k, x_i^{k+1}, v_i^k, v_i^{k+1}$ describe the stochastic process of MSAPSO, where by end of the convergence of the MSAPSO algorithm or in a stagnation situation the particles are getting almost linear independent from each other.

This means that the particle position is de facto not stochastically influenced by the previous iterations anymore. The base formulas as in the order-1 convergence case are the following:

$$x_i^{k+1} = x_i^k + v_i^{k+1}$$

$$v_i^{k+1} = w^k v_i^k + \varphi_1^k (g^k - x_i^k) + \varphi_2^k (l_i^k - x_i^k)$$

Via the same method as in order-1 convergence analysis the recursive difference equation can be found (compare for: Equation 39: Base MSAPSO recursive equations)

$$x_i^{k+1} = x_i^k (1 - \varphi^k - w^k) - w^k x_i^{k-1} + \varphi_1^k g^k + \varphi_2^k x_i^k$$

As before the expectation values are introduced. The terms are simplified for better handling of the equations and as in order-1 convergence the following term as previously described in Equation 41: Expectation values of MSAPSO particles position can be derived.

$$E(x_i^{k+1}) = E(x_i^k) E(A) + E(x_i^{k-1})E(B) + E(C)$$

Equation 52: Recalled difference equation MSAPSO for order-1/order-2 convergence

The goal for the order-2 convergence analysis is to introduce the variance of the particle's position into the order-2 dynamical system of MSAPSO. From statistics theory it is known that:

$$E((x_i^k)^2) = \text{Var}(x_i^k) + E((x_i^k))^2 \text{ or } \text{Var}(x_i^k) = E((x_i^k)^2) - E((x_i^k))^2$$

Therefore, the exact terms for $E((x_i^k)^2)$ and $E((x_i^k))^2$ in a difference equation context for the MSAPSO order-2 dynamical system need to be determined. With the method showed in (Poli, Riccardo, 2009, p. 714), which is the multiplying of $E(x_i^k)$ to both sides of the order-2 difference Equation 52: Recalled difference equation MSAPSO for order-1/order-2, the first step into this direction can be made.

$$E(x_i^k x_i^{k+1}) = E((x_i^k)^2) E(A) + E(x_i^k x_i^{k-1})E(B) + E(C)E(x_i^k)$$

In a second step $E((x_i^k)^2)$ or $E((x_i^{k+1}))^2$ can be calculated from Equation 52: Recalled difference equation MSAPSO for order-1/order-2 according to the multinomial formula of $(a + b + c)^2 = a^2 + b^2 + c^2 + 2ab + 2ac + 2bc$.

As a next step the term for $E((x_i^{k+1}))^2$ is calculated based on:

$E(x_i^{k+1}) = E(x_i^k) E(A) + E(x_i^{k-1})E(B) + E(C)$, which turns into following equation along with the multinomial rule with:

$$\begin{aligned}
 E((x_i^{k+1}))^2 &= E(A^2)E((x_i^k)^2) \\
 &+ 2(E(AB)) E(x_i^k) E(x_i^{k-1}) \\
 &+ E(B^2)E(x_i^{k-1})^2 \\
 &+ E(C^2) \\
 &+ 2(E(AC)) E(x_i^k) \\
 &+ 2(E(BC)) E(x_i^{k-1})
 \end{aligned}$$

Now the terms for $E(x_i^k x_i^{k+1})$, $E((x_i^{k+1}))^2$ are known, so the formulation of the order-2 dynamical system with mean trajectories and variance of the trajectories of MSAPSO can be done. The dynamical system denotes as following:

$$\begin{pmatrix} E((x_i^{k+1}))^2 \\ E(x_i^{k+1} x_i^k) \\ E((x_i^k)^2) \end{pmatrix} = A_{\text{order-2}} \begin{pmatrix} E(x_i^k)^2 \\ E(x_i^k x_i^{k-1}) \\ E(x_i^{k-1})^2 \end{pmatrix} + C_{\text{order-2}}$$

With

$$A_{\text{order-2}} = \begin{pmatrix} E(A^2) & 2(E(AB)) & E(B^2) \\ E(A) & E(B) & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$C_{\text{order-2}} = \begin{pmatrix} E(C^2 + 2ACx_i^k + 2BCx_i^{k-1}) \\ E(Cx_i^k) \\ 0 \end{pmatrix}$$

Equation 53: Dynamical Form of the MSAPSO order-2 convergence

Again, it is known that the definitions of $E(A)$, $E(B)$, $E(BC)$ are

$$E(A) = 1 + \mu_w - \mu_\varphi$$

$$E(B) = -\mu_w$$

$$E(C) = \mu_{\varphi_1} E(g) + \mu_{\varphi_2} E(l_i)$$

In order to derive the calculations of $E(A^2)$, $2(E(AB))$ and $E(B^2)$ respectively, the following calculation steps need to be done:

$$E(A^2) = (1 + \mu_w - \mu_\varphi)^2 = 1 + 2(\mu_w) - 2(\mu_\varphi) - 2(\mu_w)(\mu_\varphi) + (\mu_w^2) + (-\mu_\varphi^2)$$

$$E(AB) = -(\mu_w)(1 + (\mu_w) - (\mu_\varphi)) = (\mu_w)(\mu_\varphi) - (\mu_w^2) - (\mu_w)$$

$$E(B^2) = E(B(B)) = (-\mu_w)(-\mu_w) = (\mu_w^2)$$

As the relation of the variance from statistical theory is defined as:

$$\sigma_w^2 = (\mu_w^2) - (\mu_w)^2 \text{ and } \sigma_\varphi^2 = (\mu_\varphi^2) - (\mu_\varphi)^2, \text{ the substitution of the } (\mu_w^2), (\mu_\varphi^2)$$

terms in the previous equation can be done such that:

$$E(A^2) = (1 + \mu_w - \mu_\varphi)^2 = 1 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \sigma_w^2 + \mu_w^2 + \sigma_\varphi^2 + \mu_\varphi^2$$

$$E(AB) = \mu_w\mu_\varphi - \sigma_w^2 - \mu_w^2 - \mu_w$$

$$E(B^2) = E(B(B)) = \sigma_w^2 + \mu_w^2$$

Equation 54: Expectation Values of the order-2 iteration matrix

These values are later back substituted into the original $A_{\text{order-2}}$ matrix to determine the order-2 convergence curves. As the $A_{\text{order-2}}$ matrix is of the form 3x3 matrix the resulting calculation of the eigenvalues defines a cubic equation, which is not that easy to solve.

As a starting point, as with the order-1 convergence analysis the zeroing's of the characteristic polynomial of Equation 53: Dynamical Form of the MSAPSO order-2 convergence are determined. This equals the eigenvalues of the order-2 dynamical system of MSAPSO. As before the first step is to calculate the determinant of the order-2 dynamical system of MSAPSO such that:

$$X_A = \det(A - \lambda E) = 0$$

Then the rule of "Sarrus" is applied to execute the calculations in the following way:

$$\det A_{\text{order-2}} = \begin{pmatrix} E(A^2) - \lambda & 2(E(AB)) & E(B^2) \\ E(A) & E(B) - \lambda & 0 \\ 1 & 0 & 0 - \lambda \end{pmatrix} = 0$$

$$= \begin{vmatrix} E(A^2) - \lambda & 2(E(AB)) & E(B^2) \\ E(A) & E(B) - \lambda & 0 \\ 1 & 0 & 0 - \lambda \end{vmatrix} \begin{vmatrix} E(A^2) - \lambda & 2(E(AB)) \\ E(A) & E(B) - \lambda \\ 1 & 0 \end{vmatrix} = 0$$

Finally, this leads to the following terms:

$$\begin{aligned} &= (E(A^2) - \lambda)(E(B) - \lambda)(-\lambda) - (E(B) - \lambda)(E(B^2)) - (-\lambda)(E(A))(2(E(AB))) \\ &= (E(A^2) - \lambda)(-\lambda E(B)) + \lambda^2 - (E(B) - \lambda)(E(B^2)) + (\lambda)(E(A))(2(E(AB))) = \\ &-\lambda^3 + \lambda^2 E(A^2) + \lambda^2 E(B) - \lambda E(A^2) E(B) - E(B^3) - \lambda E(B^2) + \lambda(E(A))(2(E(AB))) \end{aligned}$$

As $\det(A - \lambda E) \equiv \det(\lambda E - A)$ both sides are multiplied by -1 and to finally get the zeroing's of the characteristic polynomial (eigenvalues) of the order-2 dynamical MSAPSO system with:

$$\lambda^3 - \lambda^2 E(A^2) - \lambda^2 E(B) + \lambda E(A^2) E(B) + E(B^3) + \lambda E(B^2) - \lambda(E(A))(2(E(AB)))$$

By resorting the terms this leads into the following simplified equation

$$\lambda^3 - \lambda^2 (E(A^2) - E(B)) + \lambda (E(A^2) E(B) + E(B^2) - 2E(A)E(AB)) + E(B^3)$$

As this is manually very complicated to calculate, MATLAB is used to solve for the respective eigenvalues $\lambda_{1,2,3}$ such that:

$$\lambda_1 = -E(B) = \mu_w$$

$$\lambda_{2,3} = E(B) + \frac{E(A^2)}{2} \pm \frac{E(A)\sqrt{E(A^2) - 4E(B)}}{2}$$

Equation 55: Cubic Eigenvalues equations of the order-2 MSAPSO dynamical system

With the according back substitutions from Equation 54: Expectation Values of the order-2 iteration matrix the eigenvalues of the order-2 convergence system can be found with:

$$\lambda_1 = \mu_w$$

$$\lambda_{2,3} = -\mu_w + \frac{1 + 2 E(\mu_w) - 2 E(\mu_\varphi) - 2 E(\mu_w) E(\mu_\varphi) + E(\mu_w^2) + E(\mu_\varphi^2)}{2} \pm \frac{(1 + \mu_w - \mu_\varphi) \sqrt{1 + 2 E(\mu_w) - 2 E(\mu_\varphi) - 2 E(\mu_w) E(\mu_\varphi) + E(\mu_w^2) + E(\mu_\varphi^2)} - 4 \mu_w}{2}$$

Equation 56: Eigenvalues of the order-2 MSAPSO dynamical system

Obviously, it is very difficult to find a convergence line from this equation, so the same approach is used as previously described in the order-1 convergence study. The convergence border is known from previous convergence studies such that the spectral radius of a dynamical system is defined as:

$$\rho(A) = \max |\lambda_i| < 1$$

and that a system converges formally when, the $|\lambda_i| < 1$. The convergence border of any dynamical system is given when $|\lambda_i| = 1$. This value is plugged into the iteration matrix of

the 2-order convergence system. Accordingly, the convergence border of the order-2 system can be calculated by the use of the rule of “Sarrus”.

$$\begin{vmatrix} E(A^2) - 1 & 2(E(AB)) & E(B^2) \\ E(A) & E(B) - 1 & 0 \\ 1 & 0 & 0 - 1 \end{vmatrix} \begin{vmatrix} E(A^2) - 1 & 2(E(AB)) \\ E(A) & E(B) - 1 \\ 1 & 0 \end{vmatrix} = 0$$

Consequently, the calculation steps are:

$$(E(A^2) - 1)(E(B) - 1)(-1) - (1)(E(B) - 1)(E(B^2)) - (-1)(E(A))(2(E(AB))) = 0$$

$$(E(A^2) - 1)(-E(B) + 1) - (E(B) - 1)(E(B^2)) - (-E(A))(2(E(AB))) = 0$$

$$(E(A^2) - 1)(1 - E(B)) + (1 - E(B))(E(B^2)) + (E(A))(2(E(AB))) = 0$$

$$(1 - E(B))((E(A^2) - 1) + E(B^2)) + 2(E(A))(E(AB)) = 0$$

$$(1 - E(B))((E(A^2) + E(B^2)) - 1) = -2(E(A))(E(AB))$$

$$(1 - E(B))(E(B^2) + E(A^2) - 1) = -2(E(A)E(AB))$$

As a next step $E(A)$, $E(B)$, $E(AB)$, $E(A^2)$, $E(B^2)$ needs to be back substituted

With the following equations in the previous equation:

$$E(A) = 1 + \mu_w - \mu_\phi$$

$$E(B) = -\mu_w$$

$$E(A^2) = (1 + \mu_w - \mu_\varphi)^2 = 1 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + (\mu_w^2) + (\mu_\varphi^2)$$

$(\mu_w^2), (\mu_\varphi^2)$ is equal to $E(w^2), E(\varphi^2)$ respectively and

Because for any stochastic variable the following relation holds

$$E(X^2) = \text{VAR}(X) + E(X)^2$$

It can be written that:

$$E(w^2) = \text{VAR}(w) + E(w)^2 = \sigma_w^2 + \mu_w^2$$

$$E(\varphi^2) = \text{VAR}(\varphi) + E(\varphi)^2 = \sigma_\varphi^2 + \mu_\varphi^2$$

Finally, $E(A^2), E(AB), E(B^2)$ can be calculated as following

$$E(A^2) = 1 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \sigma_w^2 + \mu_w^2 + \sigma_\varphi^2 + \mu_\varphi^2$$

$$E(AB) = \mu_w\mu_\varphi - \sigma_w^2 - \mu_w^2 - \mu_w$$

$$E(B^2) = \sigma_w^2 + \mu_w^2$$

From the previous equation of $(1 - E(B))(E(B^2) + E(A^2) - 1) = -2(E(A)E(AB))$

And by the use of $E(A), E(B), E(A^2), E(AB), E(B^2)$ the resulting calculation is as

following:

$$(1 + \mu_w)(\sigma_w^2 + \mu_w^2 + 1 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \sigma_w^2 + \mu_w^2 + \sigma_\varphi^2 + \mu_\varphi^2 - 1) =$$

$$-2(1 + \mu_w - \mu_\varphi)(\mu_w\mu_\varphi - \sigma_w^2 - \mu_w^2 - \mu_w)$$

$$(1 + \mu_w)(\sigma_w^2 + \mu_w^2 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \sigma_w^2 + \mu_w^2 + \sigma_\varphi^2 + \mu_\varphi^2) =$$

$$-2(1 + \mu_w - \mu_\varphi)(\mu_w\mu_\varphi - \sigma_w^2 - \mu_w^2 - \mu_w)$$

Then the term $2\sigma_w^2$ is canceled from both sides and the split of the first bracket right hand side is done.

$$(1 + \mu_w)(\mu_w^2 - 2\mu_\varphi + 2\mu_w - 2\mu_w\mu_\varphi + \sigma_\varphi^2 + \mu_\varphi^2 + \mu_w^2) =$$

$$-2(1 + \mu_w)(\mu_w\mu_\varphi - \mu_w^2 - \mu_w) + 2\mu_\varphi(\mu_w\mu_\varphi - \mu_w^2 - \sigma_w^2 - \mu_w)$$

First collect terms on left side

$$(1 + \mu_w)(\sigma_\varphi^2 + \mu_\varphi^2 + 2\mu_w^2 - 2\mu_w\mu_\varphi - 2\mu_\varphi + 2\mu_w) =$$

$$-2(1 + \mu_w)(\mu_w\mu_\varphi - \mu_w^2 - \mu_w) + 2\mu_\varphi(\mu_w\mu_\varphi - \mu_w^2 - \sigma_w^2 - \mu_w)$$

Then reorganize terms on left side

$$(1 + \mu_w)(\sigma_\varphi^2 + \mu_\varphi^2 - 2\mu_\varphi) - 2(1 + \mu_w)(-\mu_w^2 + \mu_w\mu_\varphi - \mu_w) =$$

$$-2(1 + \mu_w)(\mu_w\mu_\varphi - \mu_w^2 - \mu_w) + 2\mu_\varphi(\mu_w\mu_\varphi - \mu_w^2 - \sigma_w^2 - \mu_w)$$

Then the term of $-2(1 + \mu_w) (\mu_w \mu_\phi - \mu_w^2 - \mu_w)$ is cut on both sides

$$(1 + \mu_w)(\sigma_\phi^2 + \mu_\phi^2 - 2\mu_\phi) = 2\mu_\phi(\mu_w \mu_\phi - \mu_w^2 - \sigma_w^2 - \mu_w)$$

Then opening brackets on both sides

$$(1 + \mu_w) \sigma_\phi^2 + (1 + \mu_w) \mu_\phi^2 - (1 + \mu_w) 2\mu_\phi =$$

$$2\mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2 - 2\mu_\phi \mu_w$$

$$\sigma_\phi^2 + \mu_w \sigma_\phi^2 + \mu_\phi^2 + \mu_w \mu_\phi^2 - 2\mu_\phi - 2\mu_w \mu_\phi =$$

$$2\mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2 - 2\mu_\phi \mu_w$$

Cutting $-2\mu_w \mu_\phi$ and one time $\mu_\phi^2 \mu_w$

$$\sigma_\phi^2 + \mu_w \sigma_\phi^2 + \mu_\phi^2 + \mu_w \mu_\phi^2 - 2\mu_\phi - 2\mu_w \mu_\phi =$$

$$2\mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2 - 2\mu_\phi \mu_w$$

$$\sigma_\phi^2 + \mu_w \sigma_\phi^2 + \mu_\phi^2 - 2\mu_\phi = \mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2$$

$$\sigma_\phi^2(1 + \mu_w) + \mu_\phi^2 - 2\mu_\phi = \mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2$$

Shift left terms $\mu_\phi^2 - 2\mu_\phi$ to the right

$$\sigma_\phi^2(1 + \mu_w) = \mu_\phi^2 \mu_w - 2\mu_\phi \mu_w^2 - 2\mu_\phi \sigma_w^2 - \mu_\phi^2 + 2\mu_\phi$$

By collecting right terms, the following equation can be found. This equation is also reported in (Esperanza García-Gonzalo, 2014, p. 289):

$$\sigma_{\varphi}^2(1 + \mu_w) = \mu_{\varphi}^2(\mu_w - 1) + 2\mu_{\varphi}(1 - \mu_w^2 - \sigma_w^2)$$

From here a formula is searched, which is dependent from μ_{φ} . Also, it is important to understand how the order-2 convergence is dependent of the variance or standard deviation. The first step is the norming of the variance in the previous equation such that the appropriate variation coefficient or the relative variance can be found with regard to the above formula. In order to do so, both sides are divided through by μ_{φ}^2 , then by μ_w^2 .

$$\frac{\sigma_{\varphi}^2(1 + \mu_w)}{\mu_{\varphi}^2} = \frac{\mu_{\varphi}^2(\mu_w - 1)}{\mu_{\varphi}^2} + \frac{2\mu_{\varphi}(1 - \mu_w^2 - \sigma_w^2)}{\mu_{\varphi}^2}$$

$\frac{\sigma_{\varphi}^2}{\mu_{\varphi}^2}$ is replaced with cv_{φ}^2

$$cv_{\varphi}^2(1 + \mu_w) = (\mu_w - 1) + \frac{2\mu_{\varphi}(1 - \mu_w^2 - \sigma_w^2)}{\mu_{\varphi}^2}$$

$$cv_{\varphi}^2(1 + \mu_w) = \frac{\mu_{\varphi}^2(\mu_w - 1) + 2\mu_{\varphi}(1 - \mu_w^2 - \sigma_w^2)}{\mu_{\varphi}^2}$$

Now divide both sides by μ_w^2

$$\frac{cv_{\varphi}^2(1 + \mu_w)}{\mu_w^2} = \frac{\mu_{\varphi}^2(\mu_w - 1) + 2\mu_{\varphi}(1 - \mu_w^2 - \sigma_w^2)}{\mu_{\varphi}^2\mu_w^2}$$

As a next step, the terms right hand side are split into

$$\frac{cv_{\varphi}^2(1+\mu_w)}{\mu_w^2} = \frac{\mu_{\varphi}^2(\mu_w - 1)}{\mu_{\varphi}^2\mu_w^2} + \frac{2\mu_{\varphi}(1-\mu_w^2)}{\mu_{\varphi}^2\mu_w^2} - \frac{2\sigma_w^2}{\mu_{\varphi}\mu_w^2}$$

Also, the most right-hand side term with $\frac{\sigma_w^2}{\mu_w^2}$ is now replaced with cv_w^2

$$\frac{cv_{\varphi}^2(1+\mu_w)}{\mu_w^2} = \frac{\mu_{\varphi}^2(\mu_w - 1)}{\mu_{\varphi}^2\mu_w^2} + \frac{2\mu_{\varphi}(1-\mu_w^2)}{\mu_{\varphi}^2\mu_w^2} - \frac{2cv_w^2}{\mu_{\varphi}}$$

Now multiply μ_w^2 on both sides and cleanup terms on both sides

$$cv_{\varphi}^2(\mu_w + 1) = (\mu_w - 1) + \frac{2(1-\mu_w^2)}{\mu_{\varphi}} - \frac{2cv_w^2\mu_w^2}{\mu_{\varphi}}$$

Shift $(\mu_w - 1)$ to the left

$$cv_{\varphi}^2(\mu_w + 1) - (\mu_w - 1) = \frac{2(1-\mu_w^2) - 2cv_w^2\mu_w^2}{\mu_{\varphi}}$$

Divide both sides with $(\mu_w + 1) - (\mu_w - 1)$

$$cv_{\varphi}^2 = \frac{2(1-\mu_w^2) - 2cv_w^2\mu_w^2}{\mu_{\varphi}(\mu_w + 1) - (\mu_w - 1)}$$

Then exchange cv_{φ}^2 to the left side and μ_{φ} to the right side

$$\mu_{\varphi} = \frac{2(1-\mu_w^2) - 2cv_w^2\mu_w^2}{cv_{\varphi}^2(\mu_w + 1) - (\mu_w - 1)}$$

Multiply out denominator, resort terms

$$\mu_\varphi = \frac{2(1-\mu_w^2) - 2cv_w^2\mu_w^2}{cv_\varphi^2\mu_w + cv_\varphi^2 - \mu_w + 1}$$

$$\mu_\varphi = \frac{2(1-\mu_w^2) - 2cv_w^2\mu_w^2}{\mu_w (cv_\varphi^2-1) + (cv_\varphi^2+1)}$$

$$\mu_\varphi = \frac{2-2\mu_w^2 - 2cv_w^2\mu_w^2}{\mu_w (cv_\varphi^2-1) + (cv_\varphi^2+1)}$$

$$\mu_\varphi = \frac{2(1-\mu_w^2 - cv_w^2\mu_w^2)}{\mu_w (cv_\varphi^2-1) + (cv_\varphi^2+1)}$$

$$\mu_\varphi = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_\varphi^2-1) + (cv_\varphi^2+1)}$$

As a final step, the zeroing's of the order-2 convergence curve are calculated.

By setting $\mu_\varphi = 0$ and solving the previous equation, the zeroing's can be determined

with

$$0 = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_\varphi^2-1) + (cv_\varphi^2+1)}$$

Multiply both sides with $\mu_w (cv_\varphi^2 - 1) + (cv_\varphi^2 + 1)$

$$0 = 2(1 - (1 + cv_w^2)\mu_w^2)$$

Divide both sides by 2 and shift $(1 + cv_w^2)\mu_w^2$ to the left side

$$(1 + cv_w^2)\mu_w^2 = 1$$

Divide both sides with $(1 + cv_w^2)$ then we get

$$\mu_w^2 = \frac{1}{1 + cv_w^2}$$

By applying the root to the previous equation, we end in:

$$\mu_{w1,2} = w_{1,2} = \sqrt{\frac{1}{1 + cv_w^2}}$$

The final order-2 convergence system of MSAPSO is then written as, which is also reported in (Esperanza García-Gonzalo, 2014, p. 290)

$$\mu_\varphi = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_\varphi^2 - 1) + (cv_\varphi^2 + 1)}$$

And with the zeroing's of the convergence order-2 equation with

$$\mu_{w1,2} = w_{1,2} = \sqrt{\frac{1}{1 + cv_w^2}}$$

Equation 57: Order-2 convergence system for generic probability distributions

3.5.1 Visualization of the order-2 convergence system

With this formula, the next step can be made in the MSAPSO convergence analysis. In order to get an idea about the impact of the formula itself, the first step is to visualize the formula within the ranges of the parameters. As μ_φ , μ_w , cv_φ , cv_w are stochastic variables someone can anticipate that the set of convergence curves are fluctuating and vibrating. This is dependent primarily on the used probability distribution but also on the underlying problem space which is solved.

Based on a normal distribution with $N\left(\frac{1}{2}, \sigma_w\right)$, where, $\sigma_w \in \{0.0 \dots 0.7\}$ the following set of convergence curves can be drawn and visualized.

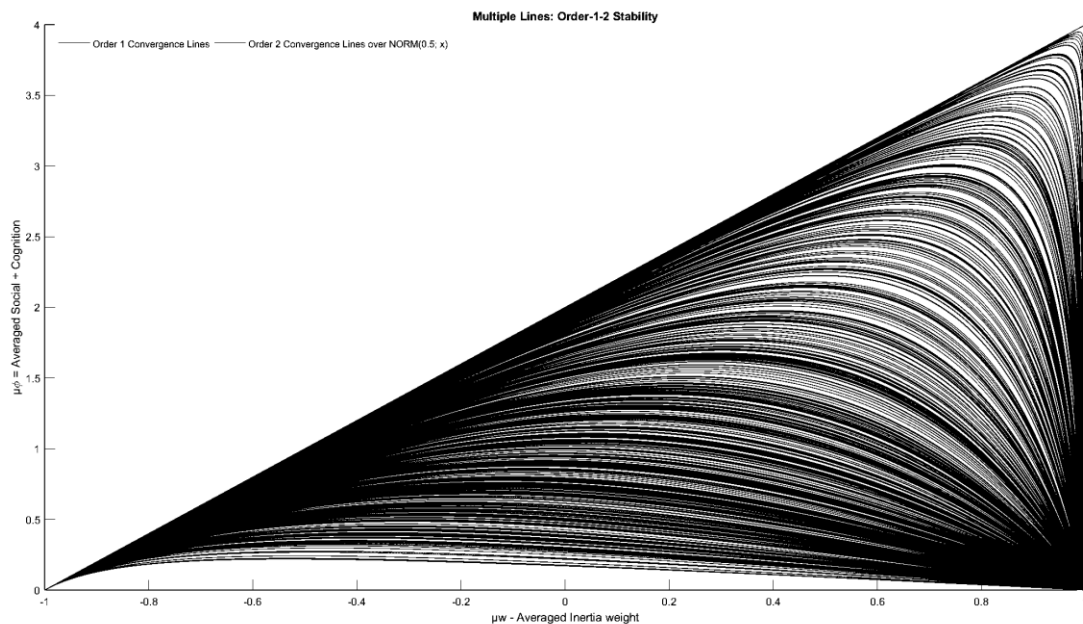


Figure 9: Set of order-2 convergence curves

It can be easily derived that the order-2 convergence lines heavily depend on the variance, in this case of the normal distribution. When the variance is almost zero, the order-2 convergence room (set of parabolic convergence curves) collapses into the order-1 convergence system (triangle) and tend to be in the upper right corner, whereas when the variance is increased the convergence curves flattens and approaches the x-axis. The hypothesis is that order-2 convergence is an element of the order-1 system, when taking out the randomness of the probability distribution. If true, this would open up a way of controlling the behavior of MSAPSO dynamically. The question here at this point is what is a good choice of a parameter set of inertia weight, social & cognition such that it is dynamic and adaptive to the underlying benchmark and probability distribution.

3.5.2 Harmonic oscillation curve order-2 convergence

In order to determine the nature of the oscillation behavior in the order-2 MSAPSO system the discriminant of Equation 55: Cubic Eigenvalues equations of the order-2 MSAPSO dynamical system is being analyzed. The first step is to define the discriminant with:

$$Y = D = E(A^2) - 4 E(B)$$

Equation 58: Discriminant of the order-2 MSAPSO dynamical system

It is known that,

$$E(A^2) = 1 + 2 (\mu_w) - 2 (\mu_\phi) - 2 (\mu_w) (\mu_\phi) + (\mu_w^2) + (\mu_\phi^2), -E(B) = \mu_w$$

with that substitutes of $E(A^2)$, $E(B)$ can be plugged in such that,

$$D = 1 + 2(\mu_w) - 2(\mu_\varphi) - 2(\mu_w)(\mu_\varphi) + (\mu_w^2) + (\mu_\varphi^2) - 4\mu_w < 0$$

$$D = 1 + 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \mu_w^2 + \mu_\varphi^2 - 4\mu_w < 0$$

$$D = 1 - 2\mu_w - 2\mu_\varphi - 2\mu_w\mu_\varphi + \mu_w^2 + \mu_\varphi^2 < 0$$

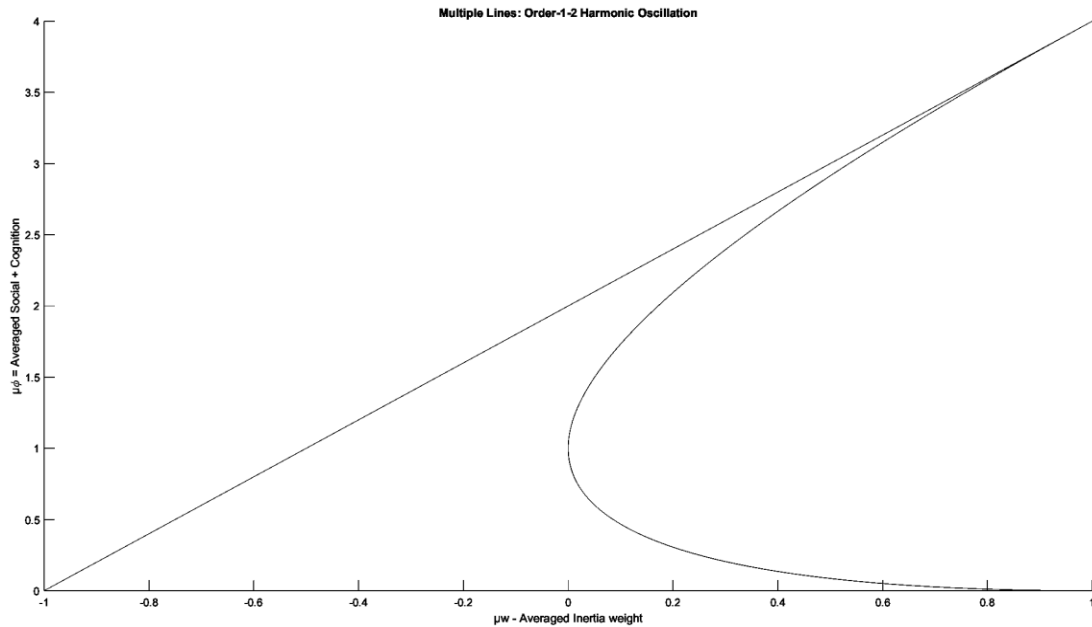


Figure 10: Same Order-1, Order-2 harmonic oscillation curve

In the figure above, it is obvious that order-2 harmonic oscillation curve is exactly the same as the order-1 harmonic oscillation curve found in Figure 8: Harmonic oscillation area around equilibrium point.

3.5.3 Mathematical proof of order-1 order-2 convergence zone collapse

It can be mathematically proven, that the hypothesis is true that the order-1/order-2 convergence room is related. For that Equation 57: Order-2 convergence system for generic probability distributions the following formula can be recalled.

$$\mu_{\varphi} = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_{\varphi}^2 - 1) + (cv_{\varphi}^2 + 1)}$$

If the randomness is reduced in the previous equation by letting run cv_w, cv_{φ} to zero then we get

$$\lim_{cv_w cv_{\varphi} \rightarrow 0} \mu_{\varphi} = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_{\varphi}^2 - 1) + (cv_{\varphi}^2 + 1)}$$

$$\mu_{\varphi} = \frac{2(1 - (1 + 0)\mu_w^2)}{\mu_w (0 - 1) + (0 + 1)}$$

$$\mu_{\varphi} = \frac{2(1 - \mu_w^2)}{-\mu_w + 1}$$

$$\mu_{\varphi} = \frac{2(1 - \mu_w)(1 + \mu_w)}{(1 - \mu_w)}$$

The term $(1 - \mu_w)$, can now be cut out, so the resulting term then finally is equal with the order-1 convergence equation denoted as:

$$= 2(1 + \mu_w) = 2 + 2\mu_w$$

This is exactly consistent with the Equation 48: Order-1 convergence limit MSAPSO dynamical system. By this approach it has been proven that the order-2 convergence can be transformed back into the order-1 convergence formula and that order-2 is element of order-1 convergence area. This knowledge can be used for the design of the MSAPSO algorithm such that varying probability distribution can be triggered for the purpose of controlling exploration and exploitation behavior of the algorithm. In general, if the chosen probability distribution does have a small variance it stimulates the exploration of the algorithm as the initiated convergence curve of the system tends to collapse into the order-1 convergence curve (convergence curve is in the upper area of the convergence room). Whereas when the variance of the used probability distribution increases the appropriate convergence curve is being flattened and then the algorithm is more in favor of exploitation behavior.

3.5.4 Application of convergence study to MSAPSO

For the specifics of the MSAPSO algorithm the base for the analysis is again Equation 57: Order-2 convergence system for generic probability distributions. In the MSAPSO case two probability distributions are used, as they are located in the center of the convergence domain. This will guarantee the balance between exploration and exploitation of the algorithm. This balance is best when both order-2 convergence lines are in the middle of the max value of the sum of social & cognition as well as in the middle of the max value of inertia weight. MSAPSO uses two different probability distributions at the same time:

- A uniform distribution $\text{UNIF}\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$

- A normal distribution with $N\left(\frac{1}{2}, \sigma_w\right)$, where $\sigma_w = \frac{1}{2\sqrt{3}}$, which is also the standard deviation of the uniform distribution

The uniform distribution is being used in the case of initialization of the algorithm and in the situation after the algorithm has run into a local minima or maxima. During regular operation of the algorithm the normal distribution is used to be more optimal in the case of general convergence. The form of the normal distribution makes sure that during regular convergence the algorithm quickly converges while also having some balanced and stable exploration and exploitation behavior. The algorithm can change in every iteration the use of the two probability distributions. Also per one iteration the inertia weight factor is static, but can vary from iteration to iteration.

Therefore, there is a need to evaluate the two extreme cases in the specific case of MSAPSO when studying the convergence characteristics.

- MSAPSO using uniform distribution to detect a first set of convergence lines
- MSAPSO after initialization just using the normal distribution afterwards with another set of convergence lines triggered by the normal case

All other cases and combinations of probability distributions being used from iterations can be found between the two set of convergence curves.

For the first case using a uniform distribution Equation 57: Order-2 convergence system for generic probability distributions is recalled with the following equation.

$$\mu_{\varphi} = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_{\varphi}^2 - 1) + (cv_{\varphi}^2 + 1)}$$

From (Esperanza García-Gonzalo, 2014, p. 291) table two and from the statistics of a uniform distribution it is known that:

- $cv_w^2 = 0$, as $\mu_w = w$, which means inertia weight is assumed static per iteration
- $cv_{\varphi}^2 = \frac{1}{6}$

When using these values, the following μ_{φ} based convergence curve can be derived:

$$\mu_{\varphi} = \frac{2(1 - (1 + 0)\mu_w^2)}{\mu_w (\frac{1}{6} - 1) + (\frac{1}{6} + 1)}$$

$$\mu_{\varphi_msa_unif} = \frac{2(1 - (\mu_w^2))}{-\mu_w (\frac{5}{6}) + (\frac{7}{6})}$$

$$\mu_{\varphi_msa_unif} = \frac{2(1 - \mu_w^2)}{-\mu_w (\frac{5}{6}) + (\frac{7}{6})}$$

Norming the right term with $\frac{6}{6}$ leads into the final μ_{φ} based convergence term of:

$$\mu_{\varphi_msa_unif} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

Equation 59: Order-2 convergence system for MSAPSO with uniform distribution

For the second case, when using a normal distribution with $N\left(\frac{1}{2}, \sigma_w\right)$, where

$\sigma_w = \frac{1}{2\sqrt{3}}$, the following order-2 convergence model can be described:

Again, from (Esperanza García-Gonzalo, 2014, p. 291) in table two and from the statistics of a normal distribution it is known that:

- $cv_w^2 = 0$, as $\mu_w = w$, which means inertia weight is assumed static per iteration
- $cv_{\varphi}^2 = \frac{\sigma_{\varphi 1}^2 + \sigma_{\varphi 2}^2}{(\mu_{\varphi 1} + \mu_{\varphi 2})^2}$
- $\mu_{\varphi 1} = \mu_{\varphi 2} = \frac{1}{2}$
- $\sigma_{\varphi 1}^2 = \sigma_{\varphi 2}^2 = \left(\frac{1}{2\sqrt{3}}\right)^2$

With $\mu_{\varphi_msa_norm} = \frac{2(1 - (1 + cv_w^2)\mu_w^2)}{\mu_w (cv_{\varphi}^2 - 1) + (cv_{\varphi}^2 + 1)}$ and the above definitions of

cv_w^2 , cv_{φ}^2 , $\mu_{\varphi 1}$, $\mu_{\varphi 2}$, $\sigma_{\varphi 1}^2$, $\sigma_{\varphi 2}^2$ it is possible to get into the following equations:

$$\mu_{\varphi_msa_norm} = \frac{2(1 - (1 + 0)\mu_w^2)}{\mu_w \left(\frac{\sigma_{\varphi 1}^2 + \sigma_{\varphi 2}^2}{(\mu_{\varphi 1} + \mu_{\varphi 2})^2} - 1 \right) + \left(\frac{\sigma_{\varphi 1}^2 + \sigma_{\varphi 2}^2}{(\mu_{\varphi 1} + \mu_{\varphi 2})^2} + 1 \right)}$$

$$\mu_{\varphi_msa_norm} = \frac{2(1 - (1 + 0)\mu_w^2)}{\mu_w \left(\frac{\left(\frac{1}{2\sqrt{3}}\right)^2 + \left(\frac{1}{2\sqrt{3}}\right)^2}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} - 1 \right) + \left(\frac{\left(\frac{1}{2\sqrt{3}}\right)^2 + \left(\frac{1}{2\sqrt{3}}\right)^2}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} + 1 \right)}$$

$$\mu_{\varphi_msa_norm} = \frac{2(1 - \mu_w^2)}{\mu_w \left(\frac{\left(\frac{1}{2\sqrt{3}}\right)^2 + \left(\frac{1}{2\sqrt{3}}\right)^2}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} - 1 \right) + \left(\frac{\left(\frac{1}{2\sqrt{3}}\right)^2 + \left(\frac{1}{2\sqrt{3}}\right)^2}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} + 1 \right)}$$

$$\mu_{\varphi_msa_norm} = \frac{2(1 - \mu_w^2)}{\mu_w \left(\frac{\left(\frac{1}{12}\right) + \left(\frac{1}{12}\right)}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} - 1 \right) + \left(\frac{\left(\frac{1}{12}\right) + \left(\frac{1}{12}\right)}{\left(\frac{1}{2} + \frac{1}{2}\right)^2} + 1 \right)}$$

$$\mu_{\varphi_msa_norm} = \frac{2(1 - \mu_w^2)}{\mu_w \left(\frac{\left(\frac{1}{6}\right)}{\left(\frac{1}{1}\right)} - 1 \right) + \left(\frac{\left(\frac{1}{6}\right)}{\left(\frac{1}{1}\right)} + 1 \right)}$$

$$\mu_{\varphi_msa_norm} = \frac{2(1 - \mu_w^2)}{\mu_w \left(-\frac{5}{6} \right) + \left(\frac{7}{6} \right)}$$

Norming the right term with $\frac{6}{6}$ again leads into the final μ_{φ} based convergence term

$$\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

Equation 60: Order-2 convergence system for MSAPSO with normal distribution

It is “surprising” that both convergence systems for MSAPSO with the uniform and the normal distribution lead to the same convergence formula. This would mean, when the variance/standard deviation of the probability distribution(s) (uniform or normal) for cognition (μ_{φ_1}) and social (μ_{φ_2}) are the same, then the appropriate convergence lines are only dependent on the coefficient of variance.

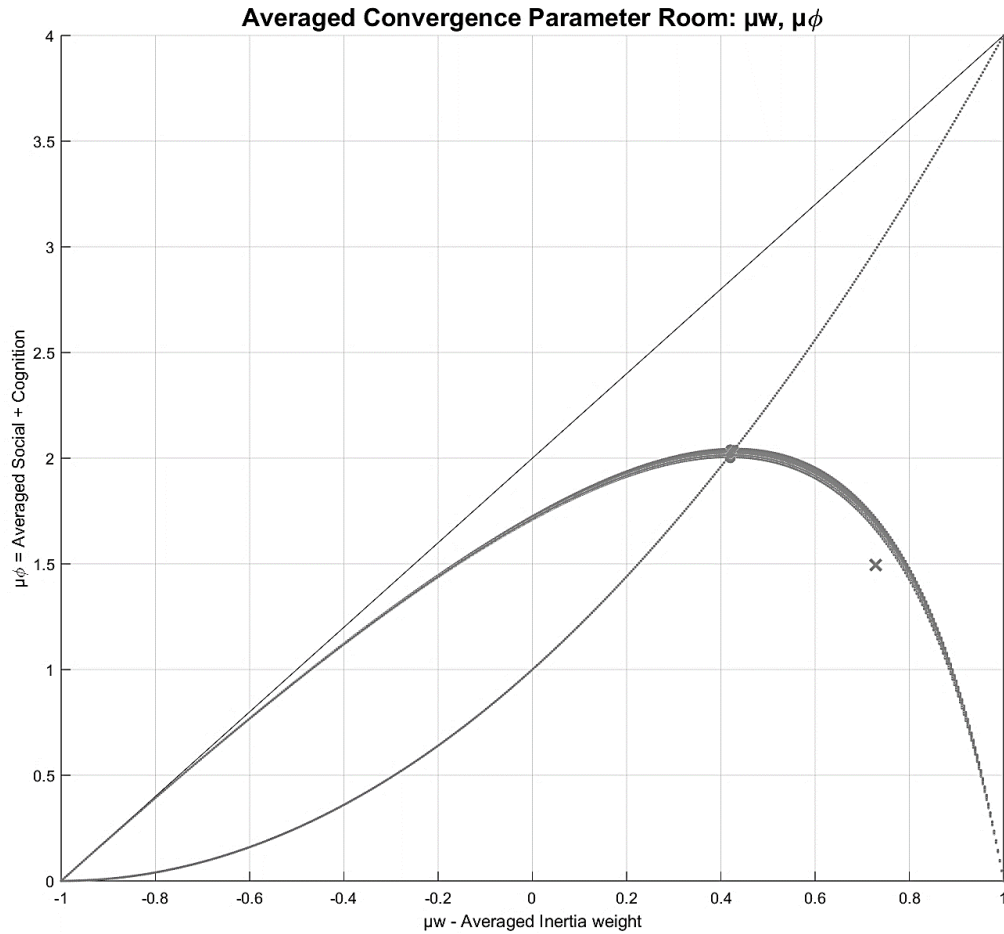


Figure 11: 2D view - order-2 convergence limits and μ_ϕ , μ_w of real algorithm runs

In the above figure the real environment is shown, where the MSAPSO algorithm takes a set of benchmarks over five dimensions and then draw the respective convergence curves of each benchmark, both with uniform as well as with normal distribution. The parabola like curves are the uniform and normal distribution based order-2 convergence limits, triggered by the respective benchmarks. The respective convergence curves are fluctuating per benchmark (Thicker area in the set of convergence curves)

The grey dots on top of the convergence curves represents the real averaged μ_ϕ with the appropriate μ_w of each specific benchmark during the real algorithmic runs.

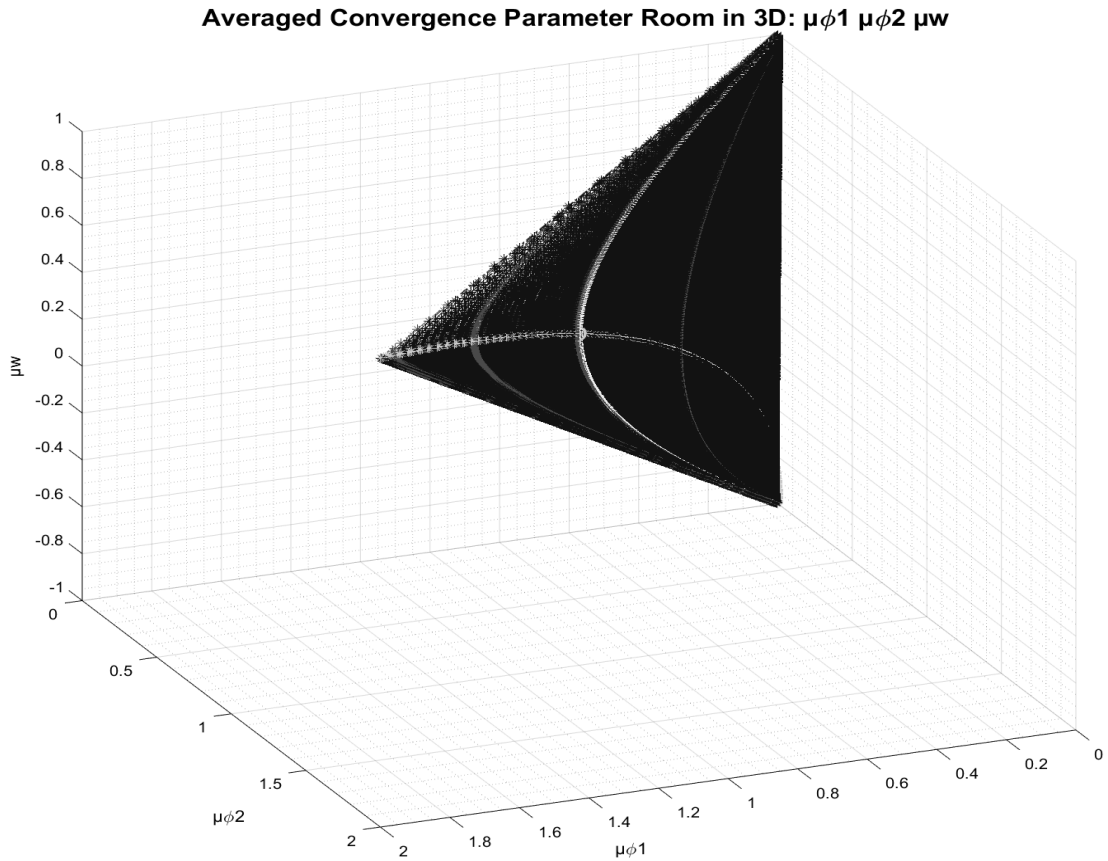


Figure 12: 3D view - order-2 convergence curves and MSAPSO stability curve

In this view $\mu_{\phi_1}, \mu_{\phi_2}$ is split on the x-, z-axis and μ_w is on the y-axis. When visualized in 3D format it can be seen that a very interesting property of the set of convergence curves do exist, when plotting convergence limits with different variance values in a normal distribution case.

The highpoints of the set of convergence curves constitutes a new line, which is named MSAPSO Stability Line (MSL) from now on. (lower left corner to lower right corner in the triangle). The highpoints of the appropriate set of convergence curves are the optimal points for balanced exploration and exploitation behavior. Mathematically this is the location curve with the property of the highpoints for all possible convergence curves. This knowledge can be used for the design of the MSAPSO algorithm later on, as a method to get independent from the variance/standard deviation of the used probability distribution

3.5.5 Final order-2 convergence room of MSAPSO

The final convergence room we can define for MSAPSO as the following convergence curve: Both for uniform and normal distribution with $N\left(\frac{1}{2}, \sigma_w\right)$.

$$\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w} \quad \mu_{\varphi_msa_unif} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

Equation 61: Final convergence room MSAPSO with uniform, normal distribution

3.6 MSAPSO Stability Line for a universal stable convergence

Based on the previously discussed convergence curves of MSAPSO, we want to better understand if there is a property that can be derived from the discussion in order to motivate some general criteria for a parameter independent model for the MSAPSO algorithm. In a first step, it is important to visualize how the set of convergence curves looks like when the parameters of the used probability distributions are varied. For the purpose of a first

visualization of the set of convergence curves the standard deviation of a normal distribution is stepwise decreased from [0.7 ... 0.0].

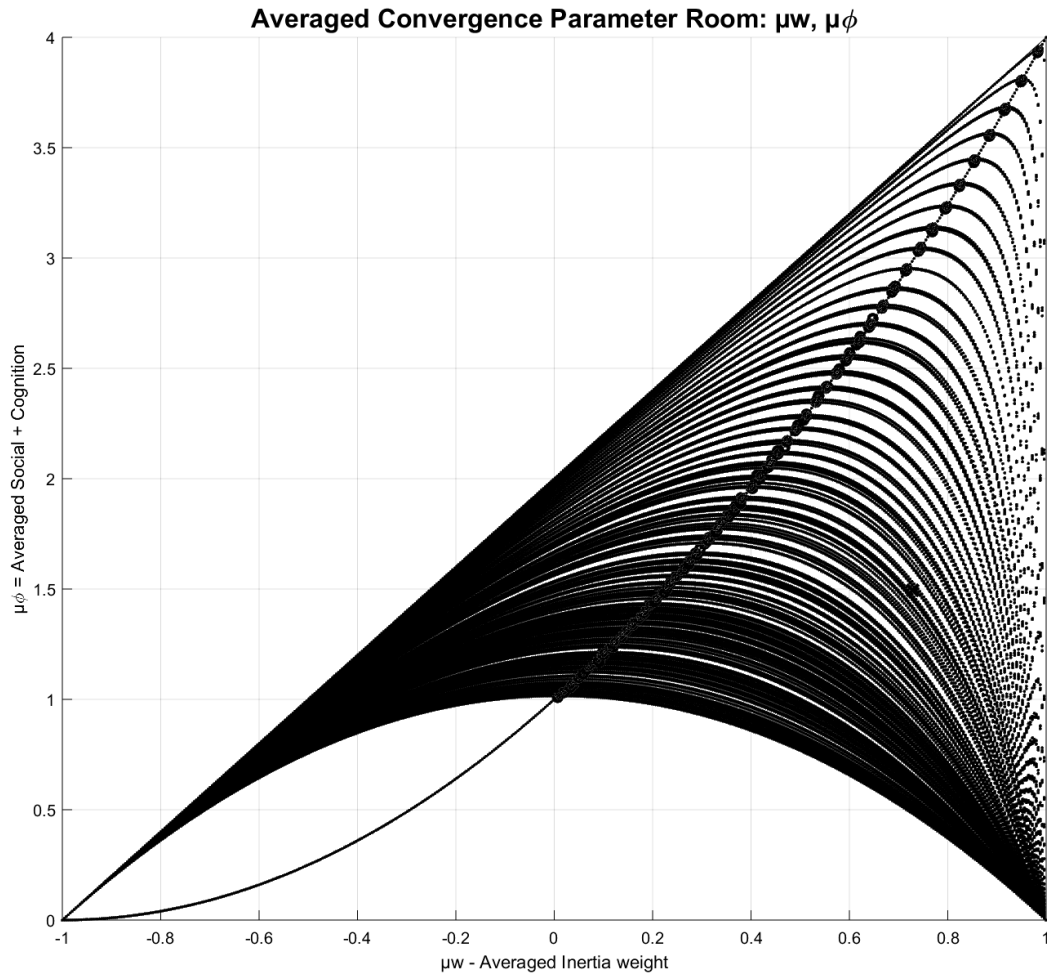


Figure 13: Visualization of set of convergence curves – Stability property MSAPSO

The set of convergence curves of MSAPSO finally tends to the upper right corner when the standard deviation of the normal distribution is reduced to zero. It looks like that the order-2 convergence curves collapses into order-1 convergence room.

The resulting curve equals all the high points of all convergence curves possible (sum of the bold points). This can be seen as a new property of these set of convergence curves. This new curve can be used as a Stability Line which makes the MSAPSO parameter less for every evaluated problem type. Visually it can be concluded, that this new curve is similar to a parable. The derived hypothesis is that these curve is equal to the following formula:

$$\mu_{\varphi_stable} = (\mu_w + 1)^2$$

Equation 62: MSAPSO Stability Line (MSL) hypothesis

The idea for a parameter less MSAPSO is that the high point of a specific convergence curve describes the optimal balance between exploration and exploitation of the particles of a swarm. The convergence curve changes from iteration to iteration slightly as random numbers created by statistics will vary by the used probability distribution. Also, there is an influence by the number of dimensions in a problem as random numbers are generated by dimension and finally are accumulated into an overall averaged number. And finally, the benchmark itself with the specific n- dimensional problem surface will have an influence on the averaged convergence curve.

Therefore, there is a need to follow the stable point(s) of the appropriate convergence curve(s) in order to have a balanced and performance-capable convergence of the algorithm. The MSL is a way to describe the specific and individual parameter sets for inertia weight, social & cognition parameter at every given iteration of the algorithmic convergence. In

order to validate our hypothesis, a two-step process is needed. First, a regression analysis overall high points of all set of convergence curve is performed to see if the resulting regression line equals with the formula in Equation 62: MSAPSO Stability Line (MSL) hypothesis and secondly, the y-value of the first deviation of Equation 60: Order-2 convergence system for MSAPSO with normal distribution is checked whether it equals the y-value of the stability line property of MSAPSO. If so, it is proved that our stability line hypothesis is correct.

3.6.1 Regression analysis of the stability property of MSAPSO

As previously described the first step is a search via a regression line whether the previously described hypothesis is true. The method is described in the following steps:

- Ten subsequent runs are used to detect the appropriate regression lines.
- In every individual run twenty-five benchmarks over five dimensions are used to detect the real averaged μ_{φ} at dynamic μ_w .
- Also, a normal distribution with $N\left(\frac{1}{2}, \sigma_w\right)$ is used to randomize particles positions where $\sigma_w = \{0.7..0.0\}$ and is varied in steps of 0.025 down from 0.7 to 0.0.
- Then the averaged regression line out of the ten subsequent runs is calculated.

RUN	Probability Distribution	Standard Deviation	Regression Line
1	Normal with uw = 0.5	0.70 .. 0.00	1.0178 uw ² + 1.9816 uw + 1.0076
2	Normal with uw = 0.5	0.70 .. 0.00	1.0058 uw ² + 1.9984 uw + 0.9977
3	Normal with uw = 0.5	0.70 .. 0.00	1.0030 uw ² + 1.9960 uw + 1.0020
4	Normal with uw = 0.5	0.70 .. 0.00	1.0042 uw ² + 1.9967 uw + 1.0013
5	Normal with uw = 0.5	0.70 .. 0.00	1.0041 uw ² + 2.0009 uw + 0.9984
6	Normal with uw = 0.5	0.70 .. 0.00	1.0102 uw ² + 1.9887 uw + 1.0034
7	Normal with uw = 0.5	0.70 .. 0.00	0.9974 uw ² + 2.0045 uw + 0.9991
8	Normal with uw = 0.5	0.70 .. 0.00	1.0176 uw ² + 1.9825 uw + 1.0037
9	Normal with uw = 0.5	0.70 .. 0.00	1.0043 uw ² + 1.9981 uw + 0.9992
10	Normal with uw = 0.5	0.70 .. 0.00	1.0077 uw ² + 1.9937 uw + 1.0006
Average			1.0071 uw ² + 1.9941 uw + 1.0013

Figure 14: Regression curves to proof MSAPSO stability property hypothesis

In the regression analysis, the results in the above figure do show that the real regression line is very close to the anticipated Equation 62: MSAPSO Stability Line (MSL) hypothesis, which is a first good indication that our stability line hypothesis is correct.

3.6.2 Mathematical proof of the MSAPSO stability curve

Now in the second step of the mathematical proof, it is checked if the y-value of the first deviation of the MSAPSO convergence curve is equal to the y-value of the MSL. In a first step the first derivation of the MSAPSO convergence curve is detected. For that the following formula is recalled:

$$\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

With MATLAB, the first derivation is calculated with

$$\mu_{\varphi_msa_norm_prime} = \frac{12(5\mu_w^2 - 14\mu_w + 5)}{(5\mu_w - 7)^2}$$

If $\mu_{\varphi_msa_norm_prime} = 0$, then the zeroing's of the first derivation is calculated with:

$$\mu_{w1} = \frac{7-2\sqrt{6}}{5} \text{ and } \mu_{w2} = \frac{2\sqrt{6}+7}{5}$$

As μ_{w1} is the only zeroing which is located in the order-1 order2 convergence area of MSAPSO, this value is used to proof whether the y-values of the convergence curve and the stability property of MSAPSO are the same.

First the μ_{w1} is plugged into $\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$ with,

$$\mu_{\varphi_msa_norm}(\mu_{w1}) = \frac{12\left(1 - \left(\frac{7-2\sqrt{6}}{5}\right)^2\right)}{7 - \frac{5(7-2\sqrt{6})}{5}} = \frac{12\left(1 - \left(\frac{7-2\sqrt{6}}{5}\right)^2\right)}{7 - \frac{(7-2\sqrt{6})}{1}} = \frac{12(1 - 0.17657)}{(7 - 2.10102)}$$

$$= \frac{12(1 - 0.17657)}{(7 - 2.10102)} = \frac{9.88116}{4.89898} = 2.01698$$

Then μ_{w1} is plugged into MSL in the same way with,

$$\mu_{\varphi_stable} = (\mu_{w1} + 1)^2 = \left(\frac{7-2\sqrt{6}}{5} + 1\right)^2 = 2.01698$$

As the y-values are the same, this would mean that we have a very good confidence, that

$(\mu_{w1} + 1)^2$ is the stability property of the set of convergence curves described.

4 DESIGN OF MSAPSO ALGORITHM

With the results of the order-1 and order-2 convergence analysis there is the foundation to design our self-adaptive MSAPSO algorithm. First the start values for social, cognition and the inertia weight parameters need to be defined. Secondly, the different flavors of self-adaptiveness methods based on the knowledge of the previous chapter have to be introduced. Also, different levels of probability distributions such as normal and uniform are evaluated to control exploration and exploitation behavior of the MSAPSO algorithm. With the knowledge of the MSL described in the previous chapter, it is possible to use any inertia based method (also new one's) to find matching and self-adaptive parameter sets in any iteration of the convergence of the algorithm. In general, former key findings from chapter three can be used to implement the described design of MSAPSO accordingly.

The relation between inertia weight, social and cognition, can be described with the found MSL. The saddle points of the set of convergence curves (equals the MSL) is an optimal balance method between exploration and exploitation for the MSAPSO in any case. As per benchmark problem the individual convergence curve can and will vary we can so to say predict in real-time the optimal saddle point per iteration.

During the convergence of the algorithm it can also happen that algorithm overcoats different regions (unimodal, multimodal) of the benchmark's problem surface. When this happens the probability distribution used, also have an influence with regard to the level of

the convergence curve in time. When applying e.g. a normal distribution instead of a uniform distribution for the position and velocity vector randomization of the particles the corresponding convergence curves do look slightly differently with regard to unimodal versus multimodal areas of the benchmark surface. In this chapter, the aforementioned aspects of the self-adaptiveness of MSAPSO will be illustrated and discussed in more detail.

4.1 Self-Adaptiveness of MSAPSO

The MSAPSO algorithm do claim that it's behavior is totally self-adaptive with regard to optimal parameter settings of social, cognition and inertia weight. This can be achieved in two ways:

- By using different types of probability distribution such as uniform, normal distributions and others who do have an average value and a standard deviation.
- By creating a method of varying inertia weight dynamically with some kind of an algorithm, then the corresponding social & cognition values can be found via the MSL.

The nature of self-adaptiveness is that of a “parameter-less” algorithm, where for every iteration in the algorithmic run as well as for different underlying benchmark problems self-optimizing parameters combinations of inertia weight, social & cognition can be found based on the MSL criteria. It is interesting to mention here, that parameter sets can exceed the appropriate convergence curve per iteration, but overall iterations the average of the

parameter values just create optimal behavior of the algorithm (optimum of convergence speed and stability) when it is slightly below the saddle point(s).

4.2 MSAPSO Stability Line Formula

One of the research contributions made is the new-found stability curve Equation 62: MSAPSO Stability Line (MSL) hypothesis over the set of convergence curves which is defined via Equation 57: Order-2 convergence system for generic probability distributions. It is named from now on MSAPSO-Stability-Formula (MSF). This curve describes the general relation between inertia weight, social & cognition in the case of the optimal balance point between exploration and exploitation of the MSAPSO algorithm. In a dynamic algorithm either inertia weight is varied or the standard deviation of the probability distribution. Also in this case the stability curve still describes the parameter relations correctly. It is important to mention here that the stability equation is stochastic in nature and is not an 100% exact prediction of the searched parameter set. The prerequisite of the stability curve/equation is, that it requires a probability distribution which do have an average and a standard deviation. There are probability distributions such as Cauchy distribution, where the stability curve then is not a valid approach.

The proof that the stability curve is a property of the set of the convergence curves was made in the previous chapter. Also, the hypothesis was proved that order-2 convergence room collapses into the order-1 convergence room under certain conditions. This is exactly the case when randomness was taken out in the order-2 convergence analysis. This was done by letting run the limes of the standard deviation of the probability distribution to

zero. In the next chapter it is analyzed how the variation options of the MSAPSO algorithm can look like.

4.3 Variation options of MSAPSO Stability Line

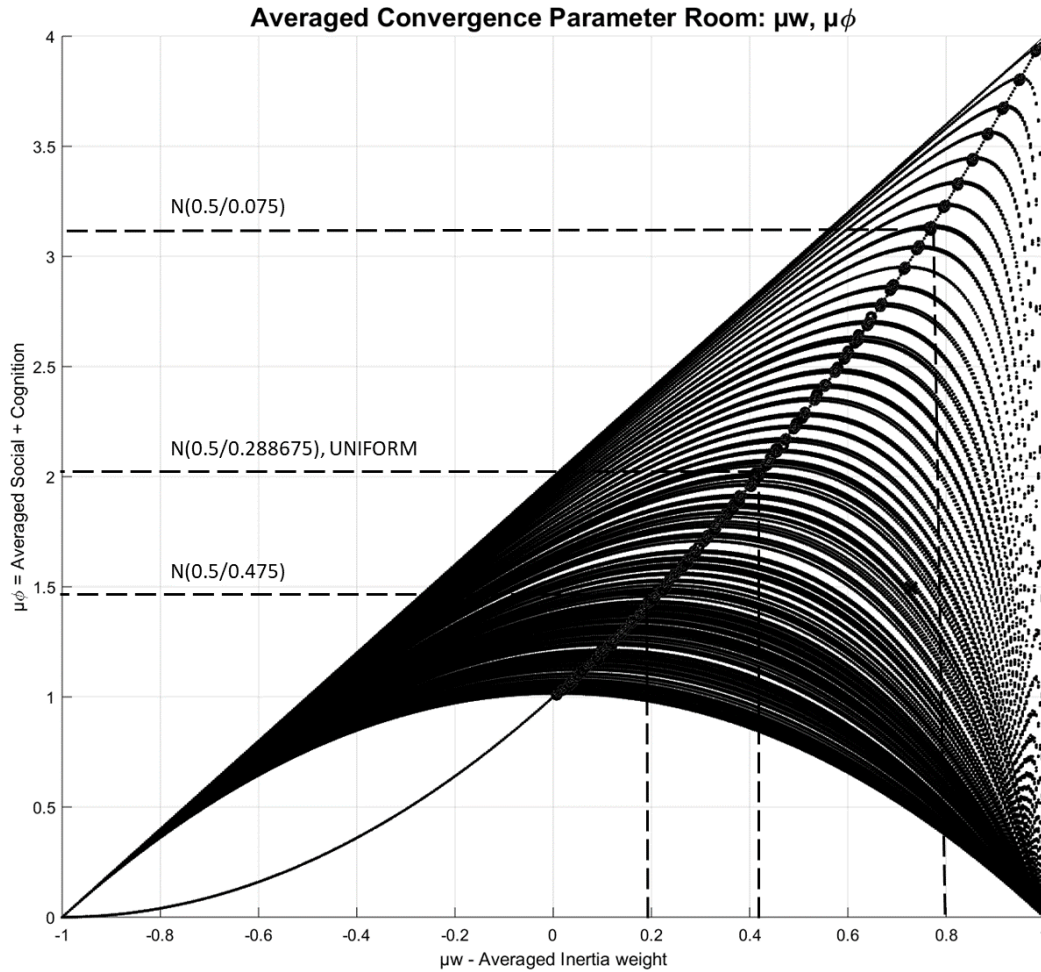


Figure 15: Variation options of MSAPSO Stability Line

As mentioned before the set of convergence curves and the resulting stability curve can be influenced via the probability distribution parameters such as average value and variance/standard deviation and/or by the use of inertia weight. In the above picture, the

options on how to influence the behavior of the MSAPSO algorithm are outlined. The first thing to mention here is, that the form of the order-2 convergence curve is fully dependent on the average value and the appropriate standard deviation of the probability distribution. In the graph, the convergence curve triggered by a uniform distribution lies in the “middle” of all convergence curves possible. Secondly when a normal distribution with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$ is used, the convergence curve form equals exactly that of the convergence curves caused by a uniform distribution.

The average value of a uniform distribution in the range of $[0,1]$ is $\frac{1}{2}$ and the standard deviation of the uniform distribution equals to $\frac{1}{2\sqrt{3}}$. The hypothesis is that when the different flavors of probability distributions do own the same average value and standard deviations that the form and the equations of the corresponding convergence lines are equal. In Figure 15: Variation options of MSAPSO Stability Line it is visualized that it is an almost exact overlay of the convergence curves triggered by uniform as well as the normal distribution with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$. If Equation 59: Order-2 convergence system for MSAPSO with uniform distribution and Equation 60: Order-2 convergence system for MSAPSO with normal distribution is recalled, it is obvious that both probability distributions do lead to the same mathematical specific convergence equation.

As the two specific probability distributions triggers the “same” order-2 convergence curve for MSAPSO and also lie in the middle of all possible convergence curves it can be concluded that these levels of convergence curve are ideal over all possible benchmarks with regard to an optimal initial point of inertia weight, social & cognition parameter.

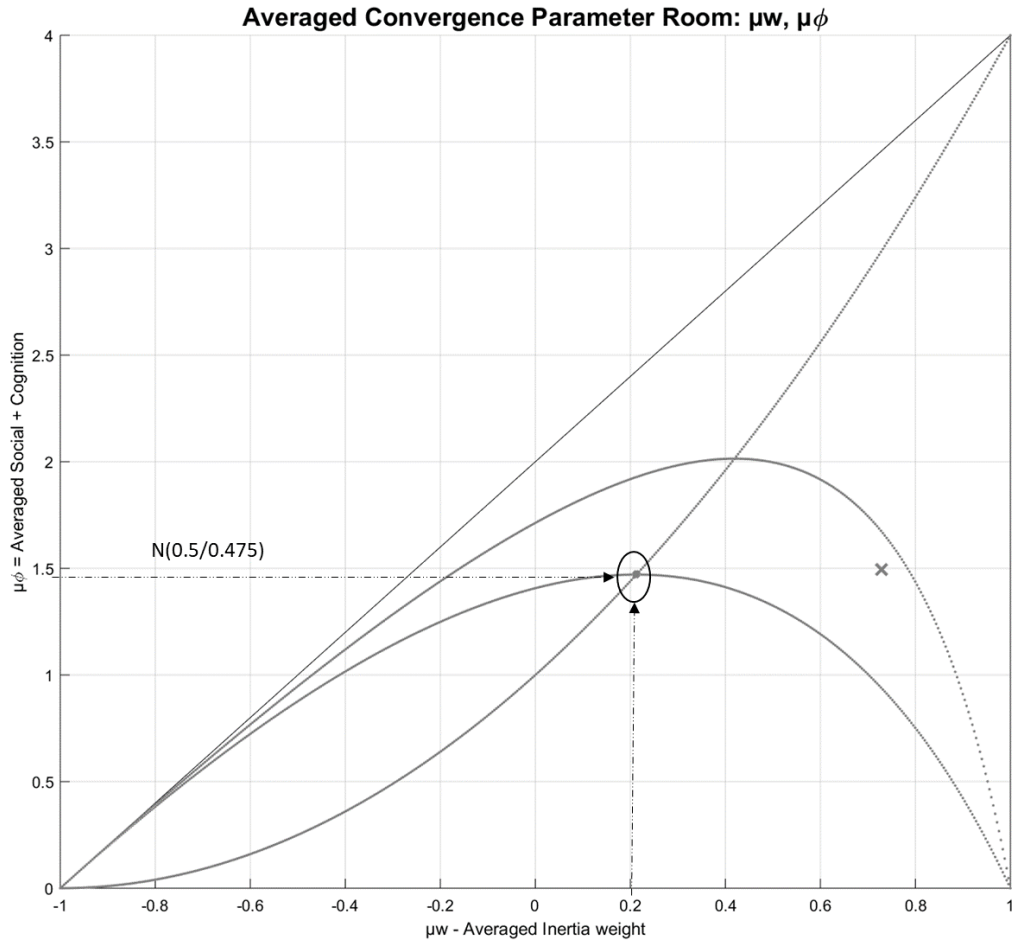


Figure 16: Lowered convergence and stability curve with $N(0.5,0.475)$

In more detail, the above figure shows how to manipulate the behavior of MSAPSO with different levels of normal distributions. In this case a normal distribution with $N(0.5,0.475)$ was used to influence the general behavior of the MSAPSO algorithm.

First of all the broader standard deviation of the normal distribution does have an influence on the height of the triggered convergence curve. Secondly the saddle point has shifted to the left in this case. This new location of the saddle point also has a corresponding inertia

weight, social and cognition value then. In general, if the inertia weight value is lowered and left-shifted the algorithms converge “faster”. The same can be achieved when the sum of social and cognition is lowered. In this case both is true, so having a low sum of social and cognition as well as low inertia weight makes the algorithm to accelerate to find solutions more quickly.

This knowledge can be used to design algorithms in order to perform faster in unimodal problems, where benchmarks do have simple problem surfaces. Whereas in multimodal benchmark problems to fast convergence speed can be a problem as there is a higher risk to be stuck into local optima and then afterwards losing the capability to find better local optima. In general, the described behavior is useful to put the algorithm into exploitation mode.

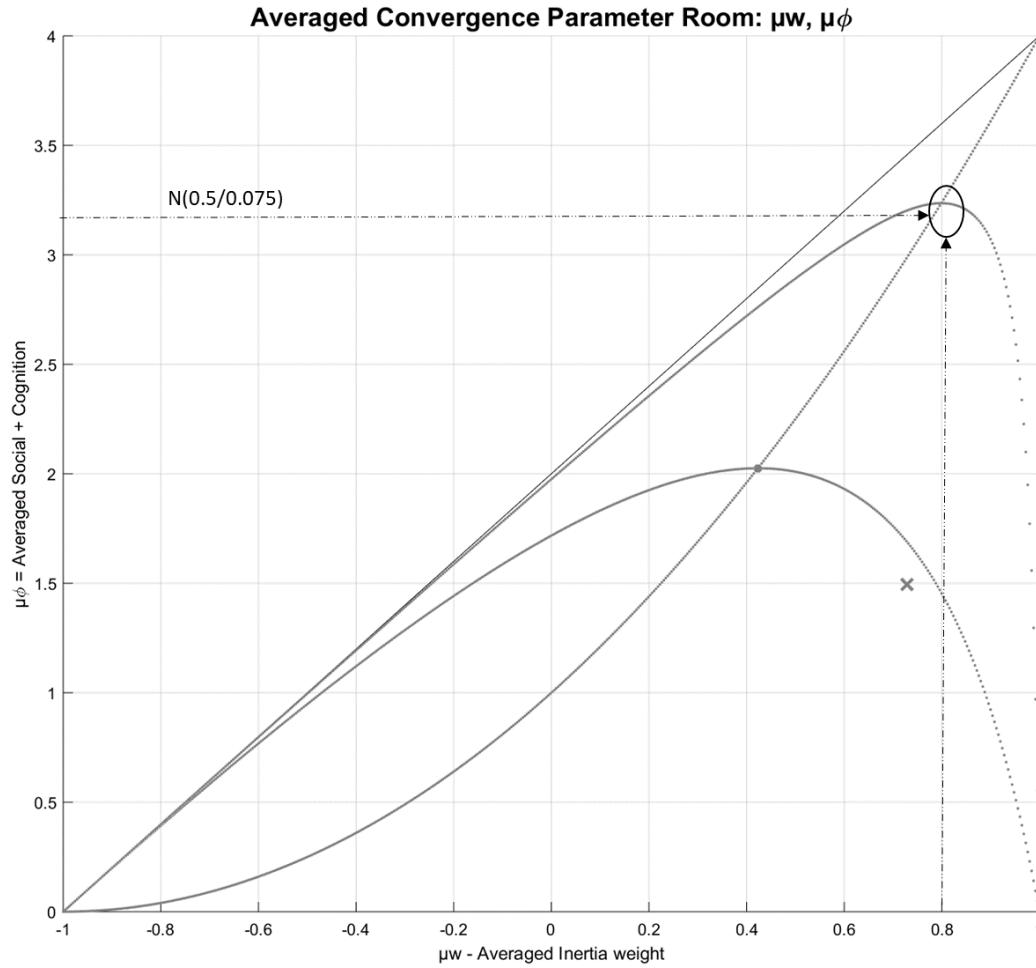


Figure 17: Raised convergence and stability curve with $N(0.5,0.075)$

In the other case, which means that the standard deviation of the normal distribution with $N(0.5,0.075)$ is decreased, then the corresponding convergence curve moves up to the upper right corner. In fact, this means that the exploration behavior of the algorithm is improved, which is useful in multimodal functions but not so useful in unimodal functions.

4.3.1 Variation with success-based inertia-weight strategy

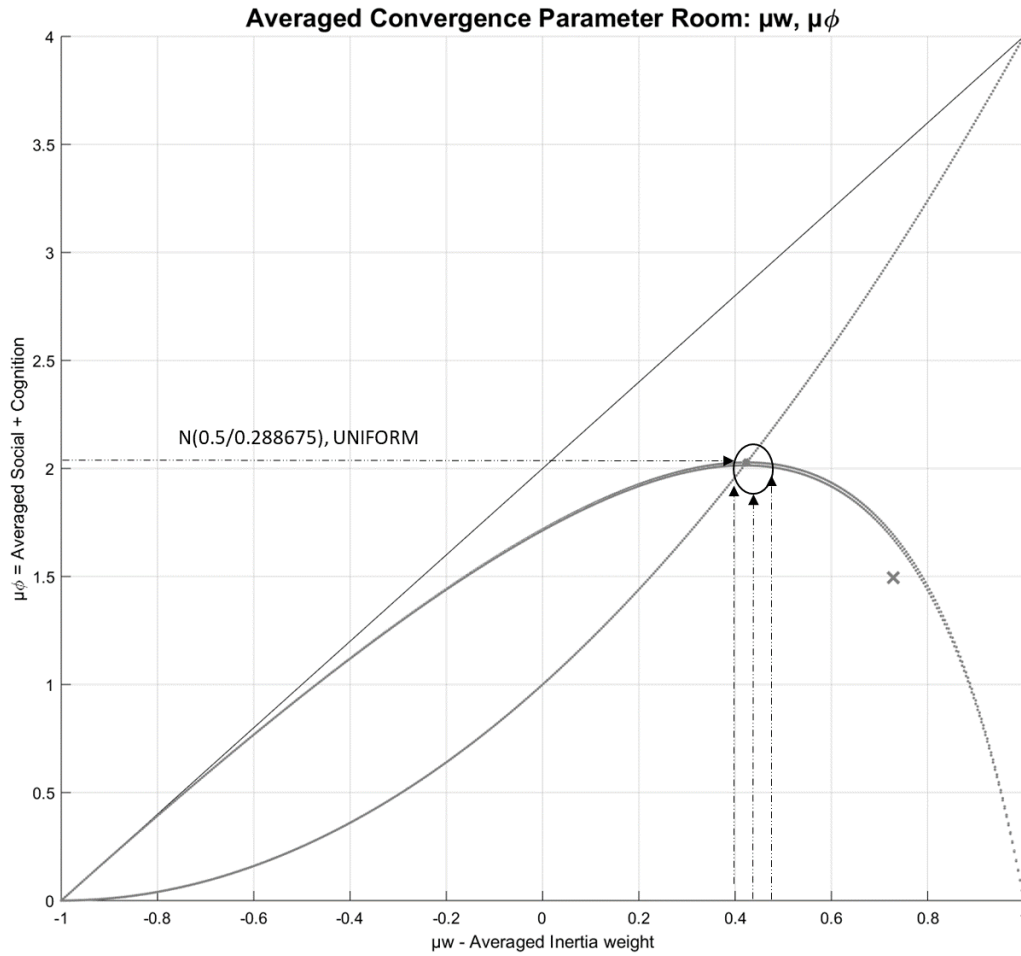


Figure 18: Balanced convergence and stability curve with $N(0.5, 0.288675)$, UNIF

Another way to let the MSAPO algorithm self-adapt is the creation of a dynamic inertia weight strategy around the saddle point location. In this case MSAPSO uses the probability distributions with uniform and normal distribution of $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$. It is important to mention here that also very slight variations of the inertia weight value around the optimal x-value of the saddle point or the appropriate x-value of the stability curve can have big impact

with regard to performance of the algorithm. This is especially true when the benchmark implies a lot of dimensions. Actually, all the options described before are being used in the real MSAPSO algorithm implementation. The method described along with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$, UNIF probability distributions is the self-adaptive method used along the regular convergence of the algorithm. When the algorithm is stuck into local optima then the self-adaptive method $N\left(\frac{1}{2}, \text{narrow value}\right)$ is used to avoid premature convergence. For the last iterations during the algorithmic run, when global optimum search stabilizes then $N\left(\frac{1}{2}, \text{broad value}\right)$ is applied in order to accelerate convergence by end of the algorithm or when in between a unimodal area is being seen by the algorithm.

4.4 Reasoning of chosen MSAPSO probability distributions

As described in previous discussions it is important to select a well-fitting probability distribution in order to support the optimal balance point of exploration versus exploitation of the MSAPSO algorithm. This point was located both with uniform as well as normal distribution with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$ at the same inertia weight value. For the two types of probability distribution it was found that the average value and the standard deviation is the same.

The following discussion will show, when we deviate from the chosen probability distributions we can either get better results for unimodal benchmarks, but then stability in multimodal functions gets worse or the other way around. It really looks like that the uniform distribution with its characteristics can be used as the “middle way” of stability

and performance and furthermore that the normal distribution with the same values of average and standard deviation like the uniform distribution makes the algorithm again perform better. Why the two types of probability distributions do create the same convergence curve can be answered from Equation 59: Order-2 convergence system for MSAPSO with uniform distribution and Equation 60: Order-2 convergence system for MSAPSO with normal distribution, where we got the same specific convergence curve with:

$$\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w} \quad \mu_{\varphi_msa_unif} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

In both cases the formula finally is just dependent on μ_w . When other levels of normal probability distributions are used other convergence-equations will arise with similar formulas but different numbers in the above $\mu_{\varphi_msa_norm}$ term. So, this means that with the average value of $\frac{1}{2}$ and the standard deviation of $\frac{1}{2\sqrt{3}}$ this is the only case where the convergence curve formulas of the uniform and normal probability distribution are the same and do lie in the “middle” of the set of convergence curves. MSAPSO uses these two levels of probability distributions during the algorithmic run, where the uniform distribution is used in the initialization phase and after the algorithm has been stuck-in-local-optima, whereas the normal distribution is being used during the normal algorithmic run with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$ and also when stuck-in-local-optima should be avoided proactively with $N\left(\frac{1}{2}, \text{narrow_value}\right)$.

4.5 Start values of MSAPSO

4.5.1 Reasoning of MSAPSO start inertia weight

The reasoning for the inertia start value can be derived from a Mathematical proof of the MSAPSO stability curves. There we calculated the zeroing's of the specific convergence curve with

$$\mu_{\varphi_msa_norm} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

$$\mu_{\varphi_msa_unif} = \frac{12(1 - \mu_w^2)}{7 - 5\mu_w}$$

The initial step was to calculate the first derivation of the above term and setting the same to zero such that:

$$\mu_{\varphi_msa_norm_prime} = \frac{12(5\mu_w^2 - 14\mu_w + 5)}{(5\mu_w - 7)^2} = 0$$

Finally, the following zeroing's values of the first deviation can be calculated with,

$$\mu_{w1} = \frac{7 - 2\sqrt{6}}{5} = 0,42020$$

$$\mu_{w2} = \frac{2\sqrt{6} + 7}{5} = 2,37979$$

Equation 63: Most optimal inertia weight values

As μ_{w1} is the only value which falls into the range of a useful inertia weight [0,1] we will choose μ_{w1} as the start inertia weight of the algorithm as it also represents the x-value of the saddle point of $\mu_{\phi_msa_norm}$ equation.

In another convergence analysis from Hua-Ma et al. (2013, p. 7) based on a Simpson distribution a very similar optimal inertia weight value with 0.4222 was reported based on an optimal spectral radius analysis. This study also showed superior performance characteristics for a general PSO compared to other variants.

The small difference in the found optimal inertia weight values can come from the different probability distributions being used in the studies. In general, this gives very good confidence that the found inertia weight start value is an optimal choice with regard to the selected probability distribution(s).

4.5.2 Swarm size in different dimensions

The swarm size defines the number of particles of the MSAPSO algorithm. The values for the MSAPSO itself, for other tested PSO variants and for the Firefly algorithm are all set to the same level.

In 2D: swarm size is set to thirty. In all other dimensions >2D swarm size is also set to thirty. In general, in this PhD study swarm size is not a parameter which is varied.

4.6 Success definition of the MSAPSO

The swarm success is an important criterion to react to certain situations while the particles are converging (e.g. particles are stuck into local minima/maxima, low success rate in previous runs). Therefore, the success is measured in two ways. First the success of the total swarm per iteration. Secondly it also interesting to measure the swarm success over a set of iterations to track the progress of the swarm towards the global minimum or maximum. Both measures are reflected in a percentage value compared to the overall success possible.

4.6.1 Swarm success per iteration

In the MSAPSO algorithm one key element of self-optimization is the idea of the “success” of a particle per iteration. Therefore, it is essential to understand how such a “success” definition translates into a mathematical model.

If we take both the particles and the total swarm’s view into consideration, we can define the following classes of success and fail for a particle in a minimization problem:

$$SUC_i = \begin{cases} 0, \text{ if particle } i \text{ failed} \mid f(x_i^d(t)) > f_{avg\ i}(t) \text{ and } f(x_i^d(t)) > f_{avg\ all}(t) \\ 1, \text{ if particle } i \text{ success} \mid f(x_i^d(t)) > f_{avg\ i}(t) \text{ and } f(x_i^d(t)) < f_{avg\ all}(t) \\ 0, \text{ if particle } i \text{ failed} \mid f(x_i^d(t)) < f_{avg\ i}(t) \text{ and } f(x_i^d(t)) > f_{avg\ all}(t) \\ 1, \text{ if particle } i \text{ success} \mid f(x_i^d(t)) < f_{avg\ i}(t) \text{ and } f(x_i^d(t)) < f_{avg\ all}(t) \end{cases}$$

Where i is the particle, x_i^d is the particle position in dimension d , $f(x_i^d(t))$ is the functional value at the particles position at timestep t , $f_{avg\ i}(t)$ equals to the average functional value of particle i , $f_{avg\ all}(t)$ is the average functional value of all particles in the swarm at timestep t and t itself is the iteration in discrete steps.

In a maximization problem, the opposite holds with:

$$SUC_i = \begin{cases} 0, & \text{if particle failed} \mid f(x_i^d(t)) < f_{avg\ i}(t) \text{ and } f(x_i^d(t)) < f_{avg\ all}(t) \\ 1, & \text{if particle success} \mid f(x_i^d(t)) < f_{avg\ i}(t) \text{ and } f(x_i^d(t)) > f_{avg\ all}(t) \\ 0, & \text{if particle fail} \mid f(x_i^d(t)) > f_{avg\ i}(t) \text{ and } f(x_i^d(t)) < f_{avg\ all}(t) \\ 1, & \text{if particle success} \mid f(x_i^d(t)) > f_{avg\ i}(t) \text{ and } f(x_i^d(t)) > f_{avg\ all}(t) \end{cases}$$

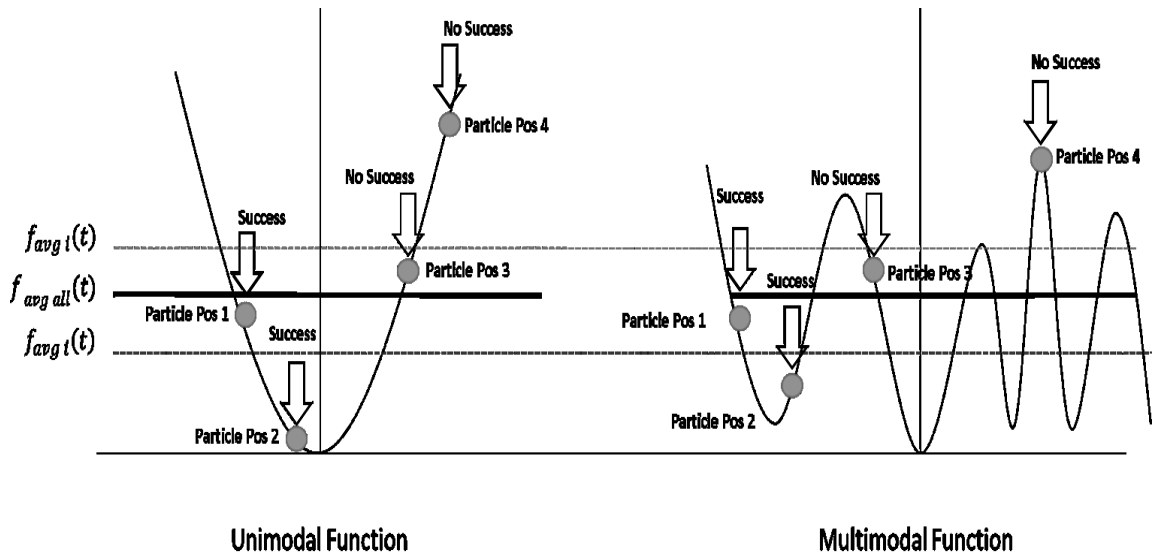


Figure 19: MSAPSO – Graphical Visualization of the “success” of a particle

The picture visualizes the success of a particle in the minima case. In all cases, where the particle state is as follows $f(x_i^d(t)) < f_{avg\ all}(t)$ (Particle Position 1, and 2) then particle i is successful, even when the actual functional value (Particle Position 1) is worse than the particle's individual average $f(x_i^d(t)) > f_{avg\ i}(t)$. In multimodal functions, this definition gets even more important, as it is much harder to decide when a particle is successful, because of the number of local optima's in a region. The proposed model provides an orientation through two measures of averaged functional value (one for the particle/one for the swarm). Based on this a two-level approach the success rate can be calculated:

- an individual particle success rate at iteration t
- a swarm success rate at iteration t

For the particles individual success rate, it is important to know if a particle is below the “contributing line” from the swarm's perspective, because then it really contributes to the success of the algorithm on all aspects. The deciding line will have two perspectives for a minimum problem.

- The particle shall improve (decrease) its own average functional value iteration by iteration to name the particle successful: For example, when the particle was five times below the limit of actual five iterations then the particle i was for sure successful from his own perspective. However, from a particle's point of view it is more important that the particle's functional value is below the functional average of all particles at iteration t .

- Secondly, from the swarm’s perspective the functional value of particle i should be smaller than the average functional value of all particles at iteration t . The swarm would not too much care about if the individual functional value at iteration $t+1$ is above his individual average value at time point t . The swarm would still see the particle successful as he still contributes to the convergence of the algorithm from his individual average functional perspective.

So, to name the particle mathematically as “successful” in an iteration with regard to a minimization problem the following condition must hold:

$$f(X_i^d(t)) < f_{avg\ i}(t) \ \& \ f(X_i^d(t)) < f_{avg\ all}(t)$$

$$f(X_i^d(t)) > f_{avg\ i}(t) \ \& \ f(X_i^d(t)) \ \text{and} \ f(X_i^d(t)) < f_{avg\ all}(t)$$

Equation 64: MSAPSO – 2-View Perspective - Success of a particle i

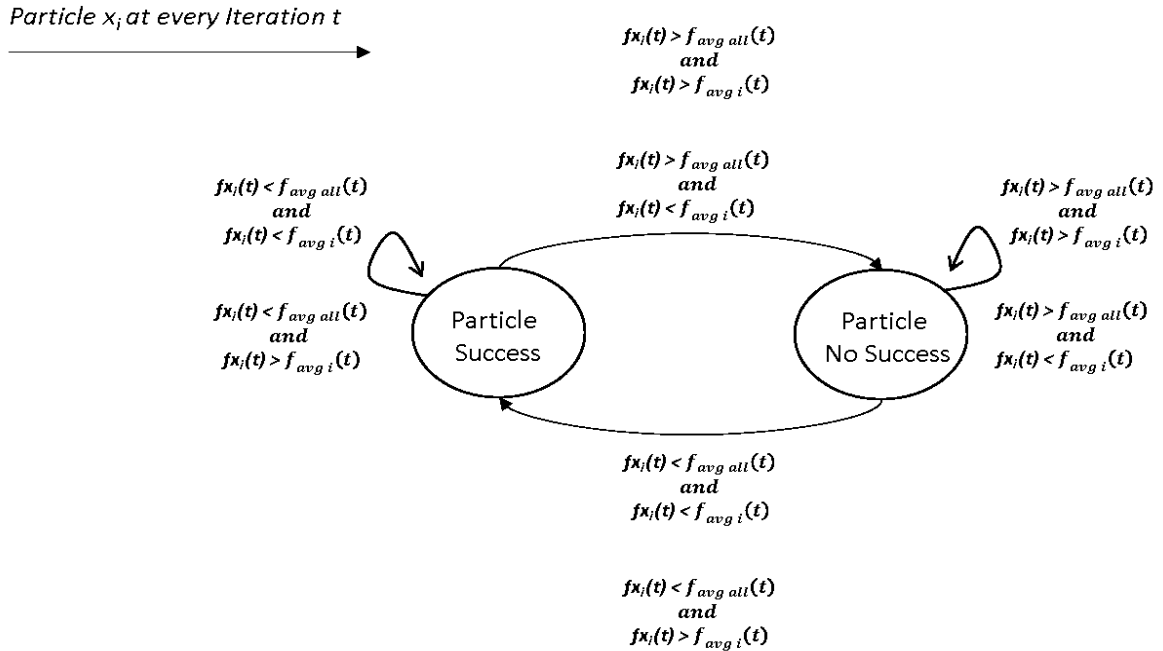


Figure 20: MSAPSO - State transition diagram of particle i – Minimum Problem

The above state transition diagram visualizes the different state changes from success to no success in a minimum problem.

The success rate in an iteration is used to capture situations such as low success rate of the swarm or being stuck-into-local-minima. For example, in a stuck-into-local-minima situation the particle swarm success is used to switch to another level of normal distribution, which then supports the swarm escaping from it by raising the height of the actual convergence curve. We remember here from previous discussions that a small standard deviation in a normal distribution raises the actual convergence line and by that supports the explorative search behavior of the swarm.

4.6.2 Swarm success in consecutive iterations

The second definition of swarm success is, if the swarm has improved itself over five consecutive iterations. It is the same definition as described in 4.6.1 with the difference that it is applied over a time sequence. This criterion is used later on, in the so-called search space characterization.

In summary, the swarm success per iteration is used in combination with the “stuck-in-local-optima” situation to proactively trigger a higher level of the convergence curve with the help of a normal probability distribution with a small standard deviation. The swarm success over consecutive iterations instead is used to characterize the search space of the actual benchmark into unimodal or multimodal like areas. With this characterization, the inertia weight parameter is either slightly adapted towards exploration or exploitation.

For unimodal functions, the parameter is shifted below the optimal inertia weight when success rate of the swarm is high to accelerate convergence of the swarm. In the case of a multimodal benchmark when the success rate is likely to be low then the inertia weight parameter is increased in order to accelerate the particles of the swarm and by that avoid the situation of a “stuck-into-local-optima”.

4.7 Escape Local Minima Strategy

PSO as an algorithm do not have an inbuilt strategy to react to situations like “stuck-into-local-optima”. Therefore, it can happen that also MSAPSO gets into this situation, even if the adaption of the parameter triple (social, cognition, inertia weight) can be dynamically adapted and adjusted during convergence as discussed previously. One of the challenges is therefore the need for a flexible “Escape Local Minima Strategy” especially when the benchmark problem does have a lot of dimensions. Then the search room for the particle also increases by the power of the dimensions. With a fixed number of particles in the swarm it means that the search room of a high dimensional benchmark problem is almost empty in relation to the total number of the particles (swarm size).

Another challenge is the No-free-lunch (NFL) theorem. In order to avoid “stuck-into-local-optima” situations a lot of PSO algorithm variants do collect a lot of information about the individual particles position and trajectory.

In specific, typically the distance from the local and global best position of the swarm is calculated overall dimensions. This is useful in order to have a decision criterion to trigger the swarm escape. On the other hand, this turns down the algorithmic performance when a “stuck-into-local-optima” situation happens in high dimensional benchmark problems as information gathering is then quite compute intensive and exponentially grows with the number of dimensions.

4.7.1 Definition of stuck-into-local-optima

In the MSAPSO algorithm the “stuck-into-local-optima” situation is reached when: after a certain amount of time the Gbest-functional-value of the algorithm do not change anymore. Theoretically when the Gbest-functional-value stays the same more than once then we might have a situation where the algorithm is “stuck-into-local-optima” or simply the algorithm loses performance even if it is able to get out of this situation by itself. On the other end if we are too fast assuming “stuck-into-local-optima” then the previously collected information by particles (expressed by their positions and Pbests) is lost after some escape procedure. It is important to mention here, that when MSAPSO faces such a “stuck-into-local-optima” situation that there is not a change of the actual Gbest-x-value until there is some better position found during the re-randomization strategy of the swarm. MSAPSO do not use swarm-radius measure to detect “stuck-into-local-optima” but for the re-randomization strategy of the swarm. Again, we have differences in the meaning being “stuck-into-local-optima” in unimodal versus multimodal benchmarks. In multimodal functions, there might be better positions closely around the actual best position. In unimodal problems, we might see the situation that at the border of the various dimensions there could theoretically be better minima’s or maxima’s. So, the escape strategy also needs to be adaptive due to the actual nature of the benchmark problem we are facing.

4.7.2 Concept of the Hyper-Middle-Point in the search space

The base concept of the MSAPSO “escape-local-minima” strategy is that of a Hyper-Middle-Point. The Hyper Middle Point (HMP) itself is the average value of the left and right border per dimension. The assumption we make here is that the borders are symmetric

in nature. In relation to the HMP we look for the x-vector orientation of the actual Gbest-value such that if it is “left- or right dimensional” orientated with regard to the HMP.

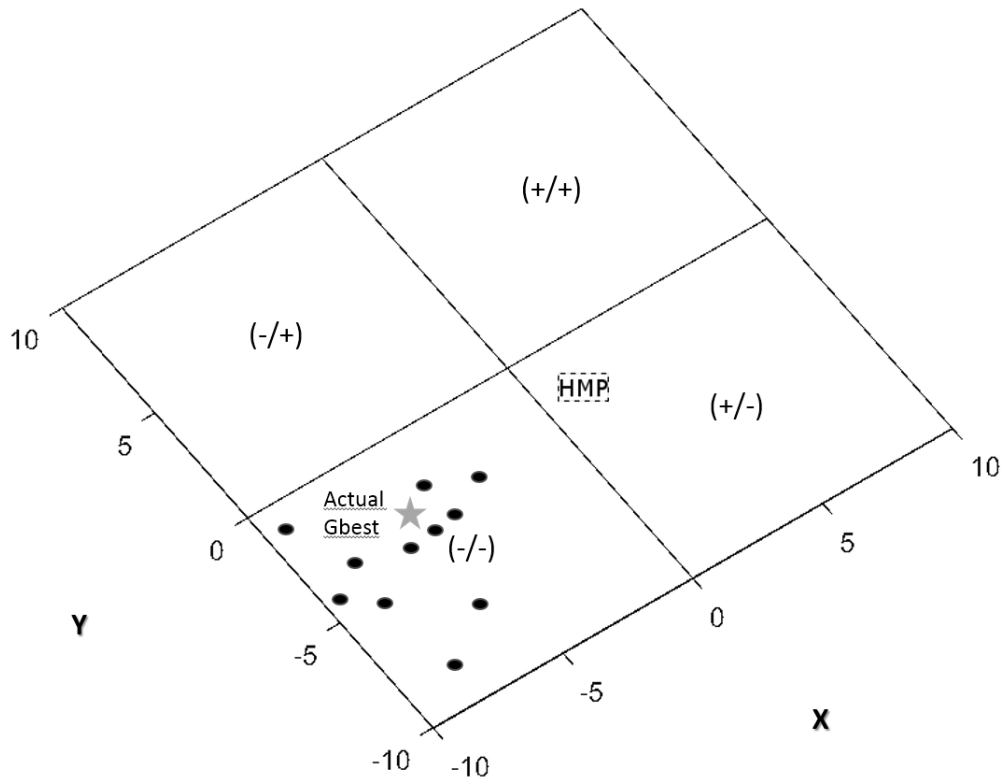


Figure 21: HMP in the 2D search space with particles close to Gbest

The HMP can be seen as a kind of a reference point in the n-dimensional search space. We can use this point to gather some information about the most likely location of the Gbest-value relative to the HMP. In our case the actual Gbest-value is located in the (-/-) quadrant. It is interesting to mention that quadrants with the opposite sign (+/+) touches the actual

Gbest-quadrant with a “corner” whereas all the others touch the Gbest-quadrant with an edge.

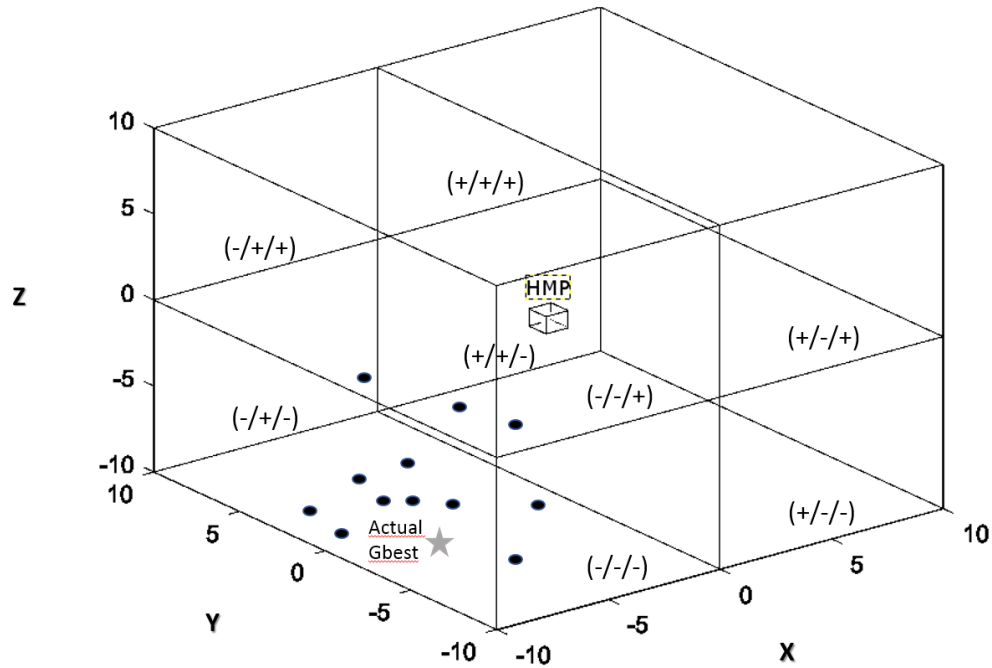


Figure 22: HMP in the 3D search space with particles close to Gbest

In a 3D problem, the HMP has the same property relative to the overall search dimensions. It is the “middle point” overall dimensions. Analogue to this the HMP can be found in the n-dimensional search space. In every dimension, the left and right ranges are summed up and divided by two to find the appropriate x-coordinates of the vector of the HMP.

4.7.3 Gathering information about Gbest-position without vector calculation

As mentioned before we want to avoid compute intensive vector calculations to derive some information about the actual Gbest-value location and orientation. Therefore, we use a method where the mean value of the Gbest-x-value is calculated in order to get an idea how the global best position relate to the HMP-x-vector. In simple words when the Gbest-x-value/Gbest-value lies in a negative orientated quadrant in the 2D case as described in Figure 21: HMP in the 2D search space with particles close to Gbest the following is true:

- $G_{best-x-value} < HMP$, Gbest-x-value is left-neighboring-orientated related to HMP
- $G_{best-x-value} > HMP$, Gbest-x-value is right-neighboring-orientated related to HMP
- $G_{best-x-value} = HMP$ then Gbest-value equals HMP

For the 3D case as shown in Figure 22: HMP in the 3D search space with particles close to Gbest this means that the Gbest-x-value must be more likely in the lower-neighboring 3D area as well. The opposite is true when the mean-Gbest-x-vector is more positive. We mark with a flag whether the Gbest-x-value is with a certain likelihood lower-neighbor-orientated or upper-neighborhood-orientated. It is to mention here that we cannot exactly calculate where the real Gbest-x-vector lies, simply for the reason that we do not want to do it for compute reasons. Based on this approximate information about the Gbest-x-value orientation we can then design a flexible and efficient escape strategy.

4.7.4 Detecting swarm radius to size the local search hyper cube

During the algorithmic run of MSAPSO the calculation of the averaged personal best vector of the swarm takes place. As mentioned earlier we do not use this to detect the stuck-local-optima situation, but for the reason we can size a local search cube in the hyper room around the actual Gbest-x-vector/Gbest-value.

The size of the local search cube is calculated as the absolute value of the difference between the $G_{best} - \text{mean}(P_{bests})$.

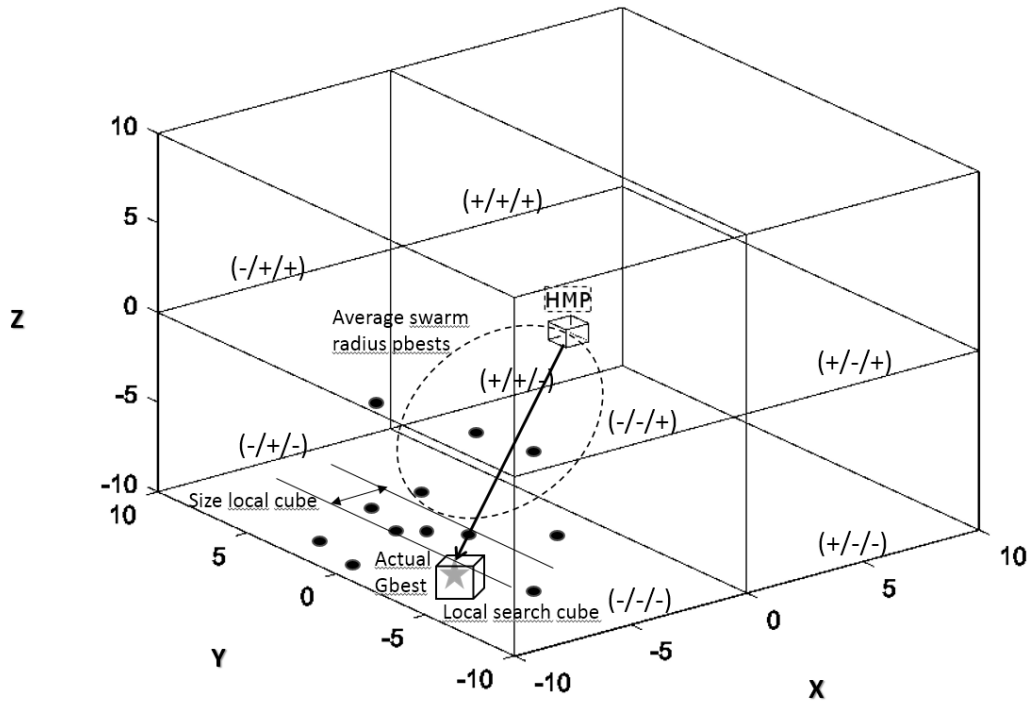


Figure 23: Defining local search hyper cube around actual Gbest

In the graph, the dashed circle represents an averaged vector of the mean Pbest-values of the swarm. For the local search hypercube calculation, this averaged vector is subtracted

from the real Gbest-x-vector to create an absolute value out of it. As a next step this absolute is used to center a local search hypercube around the actual Gbest-x-vector to perform a local search. Afterwards a subset of the particles is randomized in this local cube.

4.7.5 Detecting the global search hyper cubes

When we roughly know the approximate orientation of the Gbest-x-vector from previous discussion we not just search around the local search hypercube but also in so-called global search hyper cubes. Independent from the dimensionality we always flip two x-vector elements (e.g. x- and z-axis) of the complete vector and by that we always generate four global search hyper cubes.

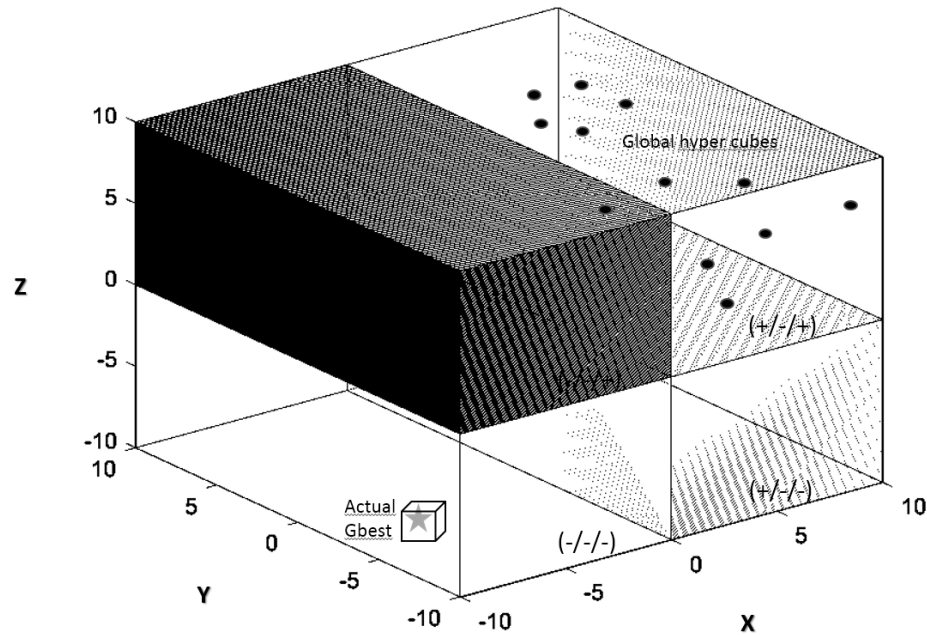


Figure 24: Defining one of the four possible global search hyper cubes

This is also true for higher dimensional benchmark problems. Then, as before part of the particles are randomized in this global search hyper cubes. It is important to mention that the likelihood of finding better minima or maxima is given by the fact that there is a higher density of particles per hyper cube. One by the other the next hypercubes is created by flipping two axes of the complete vector and by intent a hyper cube is used which is outside of the actual Gbest-x-vector. By that the likelihood is increased again that we find a better position.

Also, there are limitations to this approach. When there is a high number of dimensions the density of the particles decreases again as the amount of the four hyper cubes do not change.

What is effective, that the particle always travels back from the part hyper cube back to the actual Gbest-x-vector, which means that we have a good likelihood to find better positions along that path, as the method is repeated a couple of times.

4.7.6 HMP centric search cube

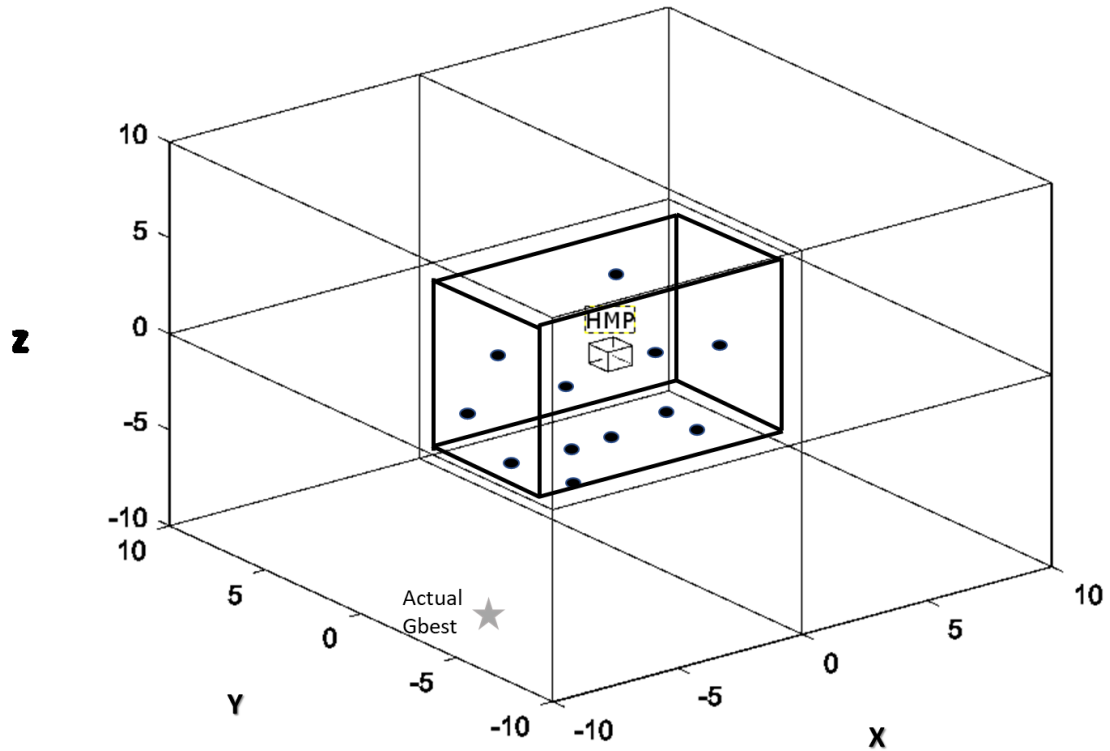


Figure 25: Defining centric hyper cube around HMP

The third element of the hyper cube strategy is using the maximum average swarm radius. The absolute value of it is then used to center a centric hyper cube around the HMP. Another effect of this is that wherever another better local minimum is located (in other regions, close to borders, etc.) part of the swarm is always centered again around the HMP in order to keep the balance between the different options. It is so to say a gravitational hyper cube between the local and global hyper cube approach and also self-adaptive in nature as the swarm radius is variable due to the other self-adaptive methods previously discussed.

4.7.7 Final escape strategy for 2D to N-dimensions

Based on the three previously discussed hyper cube strategies

- Smaller local hyper cube around Gbest-value
- Global part hyper cube based on two flipping x-vector elements
- Gravitational hyper cube to balance between local and global search

all three methods in place are used to setup an escape-local-minima strategy for MSAPSO.

The question how many particles to use for which strategy is again dependent on the type of the benchmark area (unimodal or multimodal like). In the case of a multimodal area a combination of local and global hyper cubes is used, where the number of particles for the local search was slightly higher than for the global hyper cubes.

For unimodal problems, a combination of global and centric hyper cubes is leveraged. In this case the vast majority of the particles were used for the global hyper cubes in order to better capture better minima or maxima in border regions of the search dimensions. The centric hyper cube was useful to avoid biases when performing the global hyper cube model. In the case of mixed unimodal or multimodal areas within a benchmark problem a balanced model of all three hyper cube strategies is executed.

4.8 Dynamic inertia weight strategy

Another aspect of self-adaptiveness in the MSAPSO can be achieved via a dynamic inertia weight strategy. As a foundation to it, we use our mathematically derived optimal inertia

weight value of $\mu_{\text{woptimal}} = \frac{7-2\sqrt{6}}{5}$ as a start value, which was found in the chapter “Reasoning of MSAPSO start inertia weight”. The principle is that an increase of the convergence speed of the swarm can be achieved by reducing the inertia weight value when we face a unimodal area or when there is a multimodal like segment the exploration of the swarm by increasing the inertia weight value should be strengthened.

4.8.1 Search Space characterization unimodal and multimodal

In order to implement the previous concept, a so-called search space characterization of the benchmark surface is performed. This is executed with the model of “Swarm success in consecutive iterations”. With this model, it is evaluated how successful the swarm is over five iterations at every timestep of the convergence of algorithm. The search space characterization is then as following:

Swarm success < 50% in five runs | search space
 = 0, where 0 stands for multimodal area

50% \geq Swarm success < 80% in five runs | search space
 = 1, where 1 stands for mixed area

Swarm success \geq 80% in five runs | search space
 = 2, where 2 stands for unimodal area

Equation 65: Search space characterization - unimodal and multimodal areas

4.8.2 Sliding concept around optimal inertia weight

The concept of the dynamic inertia weight model can be described based on Figure 18: Balanced convergence and stability curve with $N(0.5,0.288675)$, UNIF. In this graph, the optimal inertia weight is slightly varied with small plus or minus increments. The increments itself are dependent on the success of the swarm in consecutive iterations. In a multimode area, the increase of the increments is done by summing it up and with that the speed of the swarm is also slightly increased.

In unimodal situations, the acceleration of the swarm is done by decreasing the sum off increments. The absolute steps of the increments are set at a value of 10^{-5} . Small changes of inertia weight can lead to larger performance differences. Too high values of the incremental steps might lead either to premature convergence or to too much acceleration of the swarm. During the execution of the sliding inertia weight concept a corresponding social and cognition value can be found with Equation 62: MSAPSO Stability Line (MSL) hypothesis formula. In this case it is also mandatory that social and cognition as a sum is equal to the y-value of the previous formula., which would mean that social and cognition can have different values as long as the sum is equal.

5 EVALUATION MODEL & RESULTS

The evaluation model of MSAPSO has the objective to determine whether performance and stability of the MSAPSO algorithm is superior compared to other nature inspired algorithms. It consists of several sub-chapters where we cover the following evaluation aspects:

- Performance influence on MSAPSO when inertia weight, social and cognition is varied
- Performance influence on MSAPSO when normal distribution as adapted
- Performance influence on MSAPSO when MSF is varied
- Minimal error towards optimum of MSAPSO compared to other algorithms
- Minimal runs achieved towards optimum of MSAPSO compared to other algorithms

The tests are based on twenty-five benchmarks. The benchmarks itself do have either unimodal or multimodal characteristic. Also, the tests are split into the class of 2D benchmarks and other benchmarks which vary up to five hundred dimensions dependent on the individual benchmarks solution room. Beside the shortest runs of the algorithms we also measure the minimal error towards the global or local optimum. From an algorithms point of view, MSAPSO compare against other PSO variants as well as other nature

inspired algorithms such as the Memetic Firefly Algorithm (MFFA). The number of particles and/or fireflies is at a fixed number.

Also, the stop criterion is the same across all algorithmic variants tested.

5.1 Performance view of changes in convergence and stability zone

The intention of this chapter is to variate the values of inertia, social and cognition different from the optimal values found and discussed in chapter three. The anticipated results after the variations are as following:

- At a given inertia weight, performance degradation is seen when moving away from the optimal sum of social and cognition which is defined through Equation 62: MSAPSO Stability Line (MSL) hypothesis.
- Changes in stability of the algorithm when the level of the normal distribution is being changed. With level, here the change of the standard deviation parameter is meant, which then triggers another convergence curve.
- Performance changes when inertia weight as well as the sum of social and cognition along the appropriate MSF is variated. In this case we would see performance improvement or degradations for certain levels of the triple (inertia weight, social and cognition). The performance improvements would then be bought with reduced stability of the algorithm

In the following chapters, the variation options are discussed and described in more detail.

This is done by moving away either from the appropriate point of the stability line or by

moving away from the optimal inertia weight found in Equation 63: Most optimal inertia weight values.

5.1.1 Performance influence of inertia weight variation

In order to get an idea of the dynamics of inertia weight and the influence to the performance characteristics of MSAPSO the test is done in the following approach:

- Allow set of increments (± 0.025) to add or subtract from optimal dynamic inertia weight value and see how this influences the performance changes in unimodal or multimodal functions and compare this to the original MSAPSO
- The conditions for the above test are described as following:
 - with thirty particles in the swarm
 - with a preciseness of 10^{-5} for the convergence case
 - hundred fifty runs to accept global convergence for the algorithm
 - Benchmark functions used for the test sets
 - Bird
 - Bohachevsky
 - Booth
 - Camel
 - Dropwave
 - Dimensionality of all benchmark problems was two
 - Escape-lmin-strategy of MSAPSO is turned off to guarantee comparability

Test Type: dynamic inertia weight variation with a set of different increments, cook-formula								Runs:	100
Selective Benchmarks									
	Test Set	Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	iwt	iwt variation	type	
Bird	1	2	-1,06765E+02	187,7	0,0000E+00	0,42019	0,0000E+00	multi	
Bohachevsky	1	2	0,0000E+00	192,1	0,0000E+00	0,42020	0,0000E+00	multi	
Booth	1	2	0,0000E+00	181,1	0,0000E+00	0,42019	0,0000E+00	uni	
Camel	1	2	0,0000E+00	173,8	0,0000E+00	0,42020	0,0000E+00	multi	
Dropwave	1	2	-1,00000E+00	203,4	3,1900E-03	0,42020	0,0000E+00	multi	
				187,62	6,3800E-04				
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	iwt		type	
Bird	2	2	-1,06765E+02	187,9	5,8361E-01	0,44519	2,5000E-02	multi	
Bohachevsky	2	2	0,0000E+00	191,5	0,0000E+00	0,44518	2,5000E-02	multi	
Booth	2	2	0,0000E+00	181,2	0,0000E+00	0,44519	2,5000E-02	uni	
Camel	2	2	0,0000E+00	173,3	0,0000E+00	0,44520	2,5000E-02	multi	
Dropwave	2	2	-1,00000E+00	205,1	6,3800E-03	0,44519	2,5000E-02	multi	
				187,8	1,1800E-01				
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	iwt		type	
Bird	3	2	-1,06765E+02	188,7	1,9454E-01	0,39519	-2,5000E-02	multi	
Bohachevsky	3	2	0,0000E+00	191,9	0,0000E+00	0,39519	-2,5000E-02	multi	
Booth	3	2	0,0000E+00	181,1	0,0000E+00	0,39518	-2,5000E-02	uni	
Camel	3	2	0,0000E+00	173,3	0,0000E+00	0,39520	-2,5000E-02	multi	
Dropwave	3	2	-1,00000E+00	207,1	5,7400E-03	0,39520	-2,5000E-02	multi	
				188,42	4,0056E-02				
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	iwt		type	
Bird	4	2	-1,06765E+02	183,3	1,9454E-01	0,67019	2,5000E-01	multi	
Bohachevsky	4	2	0,0000E+00	186,1	0,0000E+00	0,67020	2,5000E-01	multi	
Booth	4	2	0,0000E+00	177,5	0,0000E+00	0,67020	2,5000E-01	uni	
Camel	4	2	0,0000E+00	171,7	0,0000E+00	0,67018	2,5000E-01	multi	
Dropwave	4	2	-1,00000E+00	200,7	7,6500E-03	0,67019	2,5000E-01	multi	
				183,86	4,0438E-02				
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	iwt		type	
Bird	5	2	-1,06765E+02	183,3	5,8361E-01	0,17019	-2,5000E-01	multi	
Bohachevsky	5	2	0,0000E+00	187,9	0,0000E+00	0,17020	-2,5000E-01	multi	
Booth	5	2	0,0000E+00	178,5	0,0000E+00	0,17019	-2,5000E-01	uni	
Camel	5	2	0,0000E+00	173,2	0,0000E+00	0,17018	-2,5000E-01	multi	
Dropwave	5	2	-1,00000E+00	196,8	7,6500E-03	0,17019	-2,5000E-01	multi	
				183,94	1,1825E-01				

Figure 26: Dynamic inertia weight with increments variation along MSF

- The first test case (grey box) is based on the original MSAPSO algorithm, as it was designed in previous chapters. This test case has the best result with regard to the combination of average runs and average total error:
 - Average runs: 187,62
 - Average error: 6,3800E-04
- In test case two we have slightly increased inertia weight by 0.025 and by that increased the sum of social and cognition along the matching MSAPSO special convergence curve. In general, this would mean that we strengthen exploration capabilities of MSAPSO with the increase of the inertia weight value. This test case has worse results compared to test case one, with regard to the combination of

average runs and average total error. In detail, it has a slightly higher average runs value, but also has a higher total error compared to test case one. More precisely we have the following results:

- Average runs: 187,80
- Average error: 1,1800E-01
- In test case three we have slightly decreased inertia weight by 0.025 and by that also decreased the matching sum of social and cognition along the corresponding MSAPSO special convergence curve. In general, this would mean that we strengthen exploitation capabilities of MSAPSO with the three factors inertia weight, social and cognition. This test case has also worse results compared to test case one, with regard to the combination of average runs and average total error. In detail, it has also a slightly higher average runs value than in test case one. It is to mention here, that the total error is smaller than in test case two, but still higher than in test case one. More precisely we have the following results:

- Average runs: 188,42
- Average error: 4,0056E-02
- In test case four and five we have a faster convergence, but this is due to the higher total error which can be reported in both cases. In detail for case four we have the following results:

- Average runs: 183,86
- Average error: 4,0438E-02

Furthermore, for case five we have the following results:

- Average runs: 183,94

- Average error: 1,1825E-01

So, bottom line we can claim that we have with the MSAPSO in the unchanged version the best results with regard to the combination of average of the total runs and the average of the total error.

5.1.2 Performance influence of probability distributions variation

Another way to test adaptability, performance and stability of MSAPSO is the variation of probability distribution used. In our case we variate the standard deviation of the normal distribution. When the standard deviation is variated then the saddle point of the appropriate convergence curve is moved upwards or downwards. In the upwards case, the standard deviation is reduced and tends towards the order-1 convergence area. In the downwards-case the standard deviation is broad which lowers the saddle point of the corresponding MSAPSO convergence curve. We use the same conditions as in the previous chapter for the test, with the difference that a normal distribution is applied at four levels of different of standard deviations. Also, in this case we have turned off escape-lmin-strategy to get comparable results with regard to the pure influence of the changing parameters of the normal distribution.

Test Type: dynamic variation of normal distribution with a set of different standard deviations, cook-formula										Runs:	100
elective Benchmarks											
	Test Set	Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal	Std	Dyn IWT		type	
	Bird	1	-1,06765E+02	187,7	0,0000E+00	N(0.5/Std)	2,8868E-01	0,42019		multi	
	Bohachevsky	1	0,00000E+00	192,1	0,0000E+00	N(0.5/Std)	2,8868E-01	0,42020		multi	
	Booth	1	0,00000E+00	181,1	0,0000E+00	N(0.5/Std)	2,8868E-01	0,42019		uni	
	Camel	1	0,00000E+00	173,8	0,0000E+00	N(0.5/Std)	2,8868E-01	0,42020		multi	
	Dropwave	1	-1,00000E+00	203,4	3,1900E-03	N(0.5/Std)	2,8868E-01	0,42020		multi	
				187,62	6,3800E-04						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal	Std			type	
	Bird	2	-1,06765E+02	210,5	3,8908E-01	N(0.5/Std)	2,0000E-01	0,56040		multi	
	Bohachevsky	2	0,00000E+00	216,1	0,0000E+00	N(0.5/Std)	2,0000E-01	0,56008		multi	
	Booth	2	0,00000E+00	196,8	0,0000E+00	N(0.5/Std)	2,0000E-01	0,56002		uni	
	Camel	2	0,00000E+00	186,9	0,0000E+00	N(0.5/Std)	2,0000E-01	0,56045		multi	
	Dropwave	2	-1,00000E+00	223,7	3,8300E-03	N(0.5/Std)	2,0000E-01	0,56010		multi	
				206,8	7,8582E-02						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal	Std			type	
	Bird	3	-1,06765E+02	179,2	0,0000E+00	N(0.5/Std)	4,0000E-01	0,28080		multi	
	Bohachevsky	3	0,00000E+00	179,7	0,0000E+00	N(0.5/Std)	4,0000E-01	0,27968		multi	
	Booth	3	0,00000E+00	174,1	0,0000E+00	N(0.5/Std)	4,0000E-01	0,27492		uni	
	Camel	3	0,00000E+00	167,8	0,0000E+00	N(0.5/Std)	4,0000E-01	0,28137		multi	
	Dropwave	3	-1,00000E+00	193,9	8,9300E-03	N(0.5/Std)	4,0000E-01	0,28059		multi	
				178,94	1,7860E-03						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal	Std			type	
	Bird	4	-1,06765E+02	384,4	5,8570E-01	N(0.5/Std)	5,0000E-02	0,86757		multi	
	Bohachevsky	4	0,00000E+00	437,8	0,0000E+00	N(0.5/Std)	5,0000E-02	0,86585		multi	
	Booth	4	0,00000E+00	337,1	0,0000E+00	N(0.5/Std)	5,0000E-02	0,86723		uni	
	Camel	4	0,00000E+00	292,8	0,0000E+00	N(0.5/Std)	5,0000E-02	0,86575		multi	
	Dropwave	4	-1,00000E+00	383,9	0,0000E+00	N(0.5/Std)	5,0000E-02	0,86585		multi	
				367,2	1,1714E-01						

Figure 27: Variation of standard deviation in normal distribution along MSF

From the previous table we can see that we are also able to “control” the MSAPSO behavior with the use of different standard deviations in the normal distribution.

- As in the case with the variation of the inertia weigh values, also in the case with the variation of the standard deviation of the normal distribution the unchanged MSAPSO algorithm is the best algorithm using a $N(\frac{1}{2}, \frac{1}{2\sqrt{3}})$ probability distribution. This is true with regard to the combined view of total average runs and the total error over five benchmarks. For simplicity reasons, the measurements were taken over from the last chapter as it is the same algorithmic test just with a different focus in this chapter. This test case one (grey box) has the best result with regard to the combination of average runs and average total error:

- Average runs: 187,62

- Average error: 6,3800E-04
- In test case two we have reduced the standard deviation to 0.2 and this lifts the MSAPSO convergence curve more into the upper right corner and by that also shift the inertia weigh value more to the right. We can see in the results table an increased value of the total averaged runs as well as an increased error compared to test case one. This test two has the following result with regard to the combination of average runs and average total error:
 - Average runs: 206,8
 - Average error: 7,8582E-02
- In test case three the standard deviation was increased to 0.4 and this shifts the MSAPSO convergence curve more into the lower left corner and by that also shift the inertia weigh value more to the left. We can see in the results table a better value of the total averaged runs, but on the other hand still an increased error compared to test case one. This test case three has the following result with regard to the combination of average runs and average total error:
 - Average runs: 178,94
 - Average error: 1,7860E-03
- In test case four we have extremely reduced the standard deviation to 0.05 and this lifts the MSAPSO convergence curve extremely more into the upper right corner and by that also shift the inertia weigh value extremely more to the right. We can see in the results table a significant increased value of the total averaged runs as well as an increased error compared to test case one. This test four has the

following result with regard to the combination of average runs and average total error:

- Average runs: 367,2
- Average error: 1,1714E-01

Also, in this test series we can finally claim that we have with the MSAPSO in the unchanged version the best results achieved with regard to the combination of average of the total runs and the average of the total error.

5.1.3 Performance and stability influence of variation of social and cognition

Another test to perform is what happens when we variate at the optimal inertia weight the sum of social and cognition combination. We recapture that the most “optimal” parameters of MSAPSO can be found at as previously discussed:

$$\mu_{w1} = \frac{7-2\sqrt{6}}{5}$$

The test will be done based such that we add/subtract an offset in steps of ± 0.01 and ± 0.02 to the sum of social and cognition at the level of the optimal inertia weight. We use again the same set of benchmarks and algorithmic conditions as before.

Test Type: dynamic Sum of social & cognition with a set of offsets along the cook-formula										Runs:	100
Selective Benchmarks											
	Test Set	Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal	Dyn IWT	SocCog	Offset SocCog	type	
	Bird	1	2	-1,06765E+02	187,7	0,0000E+00	N(0.5/Std)	0,42019	2,00578	0,00000	multi
	Bohachevsky	1	2	0,00000E+00	192,1	0,0000E+00	N(0.5/Std)	0,42020	2,00752	0,00000	multi
	Booth	1	2	0,00000E+00	181,1	0,0000E+00	N(0.5/Std)	0,42019	2,00770	0,00000	uni
	Camel	1	2	0,00000E+00	173,8	0,0000E+00	N(0.5/Std)	0,42020	2,00774	0,00000	multi
	Dropwave	1	2	-1,00000E+00	203,4	3,1900E-03	N(0.5/Std)	0,42020	2,00762	0,00000	multi
				187,62	6,3800E-04						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal					type
	Bird	2	2	-1,06765E+02	190,8	7,7815E-01	N(0.5/Std)	0,42019	2,11605	0,10000	multi
	Bohachevsky	2	2	0,00000E+00	195,1	0,0000E+00	N(0.5/Std)	0,42020	2,11819	0,10000	multi
	Booth	2	2	0,00000E+00	184,9	0,0000E+00	N(0.5/Std)	0,42019	2,11562	0,10000	uni
	Camel	2	2	0,00000E+00	176,2	0,0000E+00	N(0.5/Std)	0,42020	2,11727	0,10000	multi
	Dropwave	2	2	-1,00000E+00	214,1	3,8300E-03	N(0.5/Std)	0,42020	2,11630	0,10000	multi
				192,22	1,5640E-01						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal					type
	Bird	3	2	-1,06765E+02	195,3	1,9454E-01	N(0.5/Std)	0,42019	2,21539	0,20000	multi
	Bohachevsky	3	2	0,00000E+00	199,9	0,0000E+00	N(0.5/Std)	0,42020	2,21577	0,20000	multi
	Booth	3	2	0,00000E+00	187,8	0,0000E+00	N(0.5/Std)	0,42019	2,21719	0,20000	uni
	Camel	3	2	0,00000E+00	178,2	0,0000E+00	N(0.5/Std)	0,42020	2,21899	0,20000	multi
	Dropwave	3	2	-1,00000E+00	221,6	5,7400E-03	N(0.5/Std)	0,42020	2,21769	0,20000	multi
				196,56	4,0066E-02						
		Dimensions	Global Optimum	Minimum Average Runs	Minimum Average Error	Normal					type
	Bird	4	2	-1,06765E+02	182,8	7,7815E-01	N(0.5/Std)	0,42019	1,81735	-0,20000	multi
	Bohachevsky	4	2	0,00000E+00	187,1	0,0000E+00	N(0.5/Std)	0,42020	1,81737	-0,20000	multi
	Booth	4	2	0,00000E+00	177,5	0,0000E+00	N(0.5/Std)	0,42019	1,81798	-0,20000	uni
	Camel	4	2	0,00000E+00	170,9	0,0000E+00	N(0.5/Std)	0,42020	1,81676	-0,20000	multi
	Dropwave	4	2	-1,00000E+00	196,9	4,4600E-03	N(0.5/Std)	0,42020	1,81610	-0,20000	multi
				183,04	1,5652E-01						

Figure 28: Variation of social and cognition at optimal inertia along MSF

- As before in the other test scenarios, also in the test case one with the variation of the sum of the social and cognition value, the unchanged MSAPSO algorithm is the best algorithm using a $N(\frac{1}{2}, \frac{1}{2\sqrt{3}})$ probability distribution. This is true with regard to the combined view of total average runs and the total error over five benchmarks. This test case one (grey box) has the best result with regard to the combination of average runs and average total error:
 - Average runs: 187,62
 - Average error: 6,3800E-04
- In test case two we have increased the sum of the social and cognition value by 0.01. Compared to the test case one the results are the following:
 - Average runs: 192,22
 - Average error: 1,5640E-01

Both the total averaged runs and total averaged error is increased and therefore a worse result than in the test case one

- In test case three the sum of the social and cognition value was increased by a total value of 0.02. Compared to the test case one the results are the following:
 - Average runs: 196,56
 - Average error: 4,0056E-02

Both the total averaged runs and total averaged error is increased and therefore a worse result than in the test case one. In relation to test case two there is again an increased value with regard to the total averaged runs and total averaged error, which means that the more we step away from the optimal social and cognition value at an optimal inertia weight the more worse the algorithm will get with regard to performance.

- In test case four the sum of the social and cognition value is decreased by a total value of -0.02 . Compared to the test case one the results are the following:
 - Average runs: 183,04
 - Average error: 1,5652E-01

In this test case the total averaged runs decreased compared to test case one (faster), but the total averaged error is significantly increased and therefore it is also a worse result compared to test case one.

Also, as shown in the previous test series MSAPSO in the unchanged version can claim to have achieved the best results with regard to the combination of average of the total runs and the average of the total error. This is a very promising result with regard to an almost optimal designed MSAPSO algorithm. In the next section MSAPSO will test itself against

twenty-five selected benchmark problems and also introduce a comparative testing to other swarm inspired algorithms.

5.2 Testing method of MSAPSO against benchmark functions

In this section, the detailed test of MSAPSO algorithm will be performed such that it is possible to compare it with other swarm variants such as:

- SPSO (Standard PSO)
- XPSO (Le Clerc PSO)
- UPSO (Unified PSO)
- MFFA (Memetic Firefly algorithm)
- MSAPSO_FERNANDEZ (MSAPSO based on pure Fernandez formula)

MSAPSO with the pure MSF (Multi self-adaptive PSO) will compare to these algorithms. Also, a test set of twenty-five benchmark functions is defined to evaluate the performance of each algorithm including MSAPSO.

All the benchmarks are tested in two dimensions, and a subset of the benchmarks up to five hundred dimensions, if possible. The benchmarks itself fulfill several benchmark characteristics such as:

- Modality
- Basins

- Valleys
- Separability
- Dimensionality

The more exact definition of the above characteristics of the tested benchmark problems can be found in Jamil and Yang (2013). In this paper, the below description is outlined in detail.

Modality

The number of peaks in the benchmark landscape corresponds to the modality of a function. If the algorithms walk up these peaks during a search process, there is a tendency that the algorithm may be trapped in one of such peaks and consequently is stuck into local optima. This will have a negative impact on the overall convergence process, as this situation can move away the algorithm from the true optimal solutions.

Basins

Are characterized by relatively steep decline surrounding a large area is called a basin. Optimization algorithms can be easily attracted to such regions. Once in these regions, the search process of an algorithm is severely hampered. This is due to lack of information to direct the search process towards the minimum. A basin corresponds to the plateau for a maximization problem, and a benchmark problem can have multiple plateaus.

Valleys

A valley occurs when a narrow area of little change is surrounded by regions of steep descent. As with the basins, minimizers are initially attracted to this region. The progress of a search process of an algorithm may be slowed down considerably on the ground of the valley.

Separability

The separability is a measure of the difficulty of the respective benchmark functions. In general, separable functions are relatively easy to solve, when compared with their inseparable counterpart, because each variable of a function is independent of the other variables. If all the parameters or variables are independent, then a sequence of n independent optimization processes can be performed.

As a result, each variable or parameter can be optimized independently. In other words, a function of p variables are called separable, if it can be written as a sum of p functions of just one variable. On the other hand, a function is called non-separable, if its variables show inter-relation among themselves or are not statistically independent.

Dimensionality

The difficulty of a problem generally increases with its dimensionality as the number of variables increases. The search space also increases exponentially. For highly nonlinear problems, this dimensionality may be a significant hurdle for almost all optimization algorithms.

5.2.1 Start and convergence settings for MSAPSO algorithm

MSAPSO do have the following start values and parameter settings:

- Inertia weight: $W_{\text{start}} = \frac{7-2\sqrt{6}}{5}$
- Sum of social & cognition: $\mu_{\varphi_{\text{start}}} = (\mu_{w1} + 1)^2$
- Uniform distribution for the particles distribution at the beginning
- Normal distribution with $N\left(\frac{1}{2}, \frac{1}{2\sqrt{3}}\right)$ during regular convergence phase
- Normal distribution with $N\left(\frac{1}{2}, \text{small value}\right)$ to avoid escape-lmin situations
- Swarm size equals thirty particles for all dimensions
- Convergence is assumed after hundred fifty runs of unchanged Gbest-functional-values
- Acceptable error is defined in the algorithm to be smaller or equal than 10^{-5} compared to the real optimum
- Inertia weight, social and cognition are calculated dynamically in every iteration
- Random numbers for the probability distribution are created from a random number generator called ‘simdTwister’. ‘simdTwister’ is a new variant of Mersenne Twister (MT) introduced by Mutsuo et al. (2006). ‘simdTwister’ is a Linear Feedbacked Shift Register (LFSR) generator that generates a 128-bit pseudorandom integer at one step.

5.2.2 Description of test method

The test method is as following:

- Per algorithm the performance (minimal runs) is detected over the selected twenty-five benchmarks.
 - For all benchmarks, the performance is evaluated over two dimensions
 - For a defined subset of benchmarks, the minimal runs in higher dimensions is evaluated
- The minimal error towards the global optimum per benchmark per algorithm is determined.
 - For all benchmarks the algorithmic convergence preciseness of $< 10^{-5}$
 - Convergence is assumed, when there is no change in the actual global optimum for about hundred fifty runs.
- With the above key performance indicators (minimal runs, error towards global optimum) per benchmark per algorithm, then a total average overall benchmark per algorithm per dimension is calculated.
- We run the above method for the case of minimization as well as maximization for all the tested benchmarks.

With this information, it is possible to detect the most stable or error resistant algorithm with the best performance.

5.2.3 List of benchmarks and their characteristics

Below the selected benchmarks are listed, which are used for the to-be-performed comparative tests of MSAPSO, SPSO, XPSO, UPSO, MFFA and MSAPSO_FERNANDEZ algorithms.

Benchmark	Dimensions	Modality	Separability
Bird	[2]	multimodal	non-separable
Bohachevsky	[2]	multimodal	separable
Booth	[2]	unimodal	non-separable
Camel	[2]	multimodal	non-separable
Dropwave	[2]	multimodal	non-separable
Easom	[2]	multimodal	separable
Shubert	[2]	multimodal	separable
Zettl	[2]	unimodal	non-separable
Eggholder	[2]	multimodal	non-separable
Rana	[2]	multimodal	non-separable
Salomon	[2]	multimodal	non-separable
Schwefel	[2-4-6-15-30]	multimodal	partially-separable
Styblinski	[2-5-10]	multimodal	non-separable
Michalewicz	[2-5-10]	multimodal	non-separable
Ripple1	[2-5-10]	multimodal	non-separable
Trigonometric	[2-5-15]	multimodal	non-separable
Quintic	[2-10-20-30-100]	multimodal	separable
Deb1	[2-10-20-30-100]	multimodal	separable
Ackley1	[2-10-20-30-100]	multimodal	non-separable
Griewank	[2-10-20-30-100-250-500]	multimodal	non-separable
Rastrigin	[2-10-20-30-100-250-500]	multimodal	non-separable
Wavy	[2-10-20-30-100-250-500]	multimodal	non-separable
Sphere	[2-10-20-30-100-250-500]	unimodal	separable
Sphere small	[2-10-20-30-100-250-500]	unimodal	separable

Figure 29: List of benchmarks and their general characteristics

5.3 Test results MSAPSO versus comparative algorithms

In this section, a comparative study is made on how MSAPSO with the MSF performs against other bio-inspired algorithms such as SPSO, UPSO, XPSO, MFFA and MSAPSO_FERNANDEZ. For the studies, the tests are made over twenty-five benchmarks or a subset of the benchmarks against the above-mentioned algorithms.

5.3.1 Short description of comparative algorithms

SPSO

The SPSO is described in LeClerc (2012) with the following characteristics:

“You have a search space. On each point of this search space, you know how to evaluate a fitness, which is a numerical value. Now, you are looking for the best point, i.e. the one that has the best fitness (say the smallest one). This point is called the global optimum point (or simply optimum point, in short). In order to do that, SPSO makes use of “agents” called particles, which move step by step. A step is called an iteration (or sometimes a time step). A particle is made of a position inside the search space, the fitness value at this position, a velocity (in fact a displacement), which will be used to compute the next position, a memory, that contains the best position (called the previous best) found so far by the particle, the fitness value of this previous best. The set of particles is called the swarm. Inside the swarm a topology is defined: it is a set of links between particles, saying “who informs whom”. When a particle is informed by another one, it means that the particle knows the previous best position. The set of particles that informs a particle is called its neighborhood.

In SPSO, the neighborhood contains the particle itself, but is not necessarily symmetrical. The search is performed in two phases: initialization of the swarm, and then a cycle of iterations. The parameters of social, cognition and inertia weight are usually set statically as following: (social, cognition both equals 1.49445, inertia weight is set at 0.7298). A more mathematical described also can be found in the chapter 2.1.1. The basic movement equations are:

- Particle's Velocity :

$$\vec{v}_i(t) = w\vec{w}_i(t-1) + c1\phi1 \left(\vec{P}_i - \vec{x}_i(t-1) \right) + c2\phi2 \left(\vec{P}_g - \vec{x}_i(t-1) \right)$$

- Particle's Position :

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$$

XPSO

In addition to the PSO algorithm, XPSO (Constriction PSO) which is the constricted version of the PSO adds a so-called constriction factor χ to the motion equations of the particle's. This is due to the following argumentation. As the traditional version of the PSO algorithm can explode or diverge from particle movements point of view in the situation when:

$$c1\phi1 + c2\phi2 > 4$$

XPSO examines the condition when the PSO exactly converges. LeClerc (2005/2006, p. 222) and following pages describe how to turn the original motion equations of PSO into a convergence analysis of the XPSO system.

This is done by the following steps: The base equations of SPSO referenced in Equation 2: SPSO – Particles position are transformed via several steps into the XPSO base equations with:

$$V(t + 1) = V(t) + \varphi(P - X(t))$$

$$X(t + 1) = X(t) + V(t + 1)$$

Equation 66: Base equations of XPSO

Then $Y(t) = P - X(t)$ is set, consequently the resorted term is $X(t) = P - Y(t)$

$$V(t + 1) = V(t) + \varphi Y(t)$$

$$-Y(t + 1) = -Y(t) + V(t + 1), \text{ when } P = 0$$

Then $V(t + 1)$ is plugged into the second equation it is possible to reduce the system into

$$-Y(t + 1) = -Y(t) + V(t) + \varphi Y(t)$$

Then both sides are multiplied by -1 by resorting terms we finally get:

$$Y(t + 1) = -V(t) + (1 - \varphi) Y(t)$$

The dynamical system of XPSO composed of velocity and actual position of the particle can then be written as a dynamical system:

$$V(t + 1) = V(t) + \varphi Y(t)$$

$$Y(t + 1) = -V(t) + (1 - \varphi) Y(t)$$

Equation 67: Dynamical system of XPSO

The system matrix C of coefficients can be concluded from the previous equation with

$$C = \begin{pmatrix} 1 & \varphi \\ -1 & 1 - \varphi \end{pmatrix}$$

As a next step the determinant of the system matrix can be calculated. By setting it zero, the eigenvalues can be determined (characteristic polynomial).

$$\det(C - \lambda E) = \det \begin{bmatrix} \chi - \lambda & -\chi\varphi \\ -\chi & \chi(1 - \varphi) - \lambda \end{bmatrix} = 0$$

From here the quadratic equation can be derived with the Cramer rule. Finally, the quadratic eigenvalue equation can be concluded as:

$$\lambda^2 + \chi(\varphi - 2)\lambda + \chi^2 = 0$$

This equation is solved for λ and then resolved for χ . Finally, the constriction factor can be computed with:

$$\chi = \frac{2}{|\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}|} \text{ where } \varphi = c1\varphi1 + c2\varphi2 \text{ and } \varphi \geq 4$$

Equation 68: Constriction factor of XPSO

With the setting of $\varphi = 4.1$ this leads typically to a constriction factor of $\chi \approx 0.7298$

This constriction factor is then applied to the new motion equation so the XPSO will converge safely. The new motion equations of XPSO are denoted as following:

- Particles Velocity :

$$\vec{V}_i(t+1) = \chi \vec{V}_i(t) + c1\varphi1 \left(\vec{P}_i - \vec{X}_i(t) \right) + c2\varphi2 \left(\vec{P}_g - \vec{X}_i(t) \right)$$

- Particles Position :

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) + (1 - \chi) \left(c1\varphi1 \left(\vec{P}_i - \vec{X}_i(t) \right) + c2\varphi2 \left(\vec{P}_g - \vec{X}_i(t) \right) \right)$$

Equation 69: Motion equations of XPSO

UPSO

Unified Particle Optimization (UPSO) is described in detail in the literature review in chapter two.

MFFA

MFFA is a very popular and efficient nature inspired swarm algorithm described in Yang et al. (2013). Please refer to the detailed description in the mentioned literature reference. Also, MFFA was discussed in detail also in chapter two.

MSAPSO_FERNANDEZ

MSAPSO_FERNANDEZ is exactly the same algorithm as MSAPSO, with the following differences: It has no variation of the theoretical inertia weight factor, which is based on a previous search room characterization in MSAPSO. Instead of the stability formula found in chapter three Equation 62: MSAPSO Stability Line (MSL) hypothesis and the special convergence curves for MSAPSO Equation 61: Final convergence room MSAPSO with uniform, normal distribution, it uses the original Martinez Fernandez convergence formula. This formula is described in Equation 57: Order-2 convergence system for generic probability distributions.

5.3.2 Preconditions for the long run test

The preconditions for the long runs for all algorithms tested are the following:

- Preciseness towards the global optimum is defined with 10^{-5}

- Convergence of the algorithm is assumed when the actual Gbest-val at Gbest position has not changed for about hundred fifty runs within an algorithmic run.
- To achieve statistical significance the number of runs is set to the following values for the respective benchmarks:
 - = 2D for all benchmarks with 500 runs
 - > 2D and < 100D with 250 runs
 - $\geq 100D$ and $\leq 500D$ with 100 runs

5.3.3 Benchmarks and Minimum Tests

BENCHMARK	MINIMA GBESTVAL $F(X_n)$	MINIMA GBEST X_n
BIRD	-1,06765E+02	MULTIPLE
BOHACHEVSKY	0,00000E+00	(0,...0)
BOOTH	0,00000E+00	(1,...3)
CAMEL	0,00000E+00	(0,...0)
DROPWAVE	-2,06261E+00	(0,...0)
EASOM	-1,00000E+00	(3.145927,...,3.145927)
SHUBERT	-1,86731E+02	MULTIPLE
ZETTL	-3,79000E-03	(-0.02989,...,0.00000)
EGGHOLDER	-9,59641E+02	(512,...,404.2319)
RANA	-5,13000E+02	MULTIPLE
SALOMON	0,00000E+00	(0,...0)
SCHWEFEL	-8,37966E+02	(420.96874,...,420.96874)
STYBLINSKITANG	-7,83323E+01	(-2.90353,...,-2.90353)
MICHALEWICZ	-1,80130E+00	(2.20290,...,1.57079)
RIPPLE1	-2,00000E-01	(0,...0)
TRIGOCOMETRIC	0,00000E+00	MULTIPLE
QUINTIC	0,00000E+00	MULTIPLE
DEB1	-1,00000E+00	MULTIPLE
ACKLEY1	0,00000E+00	(0,...0)
GRIEWANK	0,00000E+00	(0,...0)
RASTRIGIN	0,00000E+00	(0,...0)
WAVY	0,00000E+00	(0,...0)
SPHERE	0,00000E+00	(0,...0)
SPHERE AT ONE	0,00000E+00	(1,...1)
HOLDERTABLE	-1,92085E+01	(-8.05502,...,-9.66459)

Figure 30: List of benchmarks MIN-TEST in respective dimensions

5.3.3.1 Minimum 2D Long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO MARTINEZ	FTO	SPSO	FTO
BIRD	187,6	0,00000E+00	189,0	0,00000E+00	221,4	0,00000E+00
BOHACHEVSKY	159,4	0,00000E+00	159,9	0,00000E+00	229,4	0,00000E+00
BOOTH	182,0	0,00000E+00	182,5	0,00000E+00	207,3	0,00000E+00
CAMEL	159,3	0,00000E+00	159,7	0,00000E+00	192,1	0,00000E+00
DROPWAVE	159,7	0,00000E+00	159,6	0,00000E+00	222,9	2,68000E-03
EASOM	190,2	0,00000E+00	191,2	0,00000E+00	220,9	2,00000E-03
SHUBERT	203,2	0,00000E+00	203,4	0,00000E+00	251,1	0,00000E+00
ZETTL	175,4	0,00000E+00	175,8	0,00000E+00	194,7	0,00000E+00
EGGHOLDER	192,6	0,00000E+00	193,4	0,00000E+00	249,2	4,38330E+01
RANA	359,7	4,18560E-01	361,9	4,24610E-01	498,5	6,62270E-01
SALOMON	159,6	0,00000E+00	159,3	0,00000E+00	237,8	7,76000E-03
SCHWEFEL	194,8	0,00000E+00	195,7	0,00000E+00	254,6	4,38222E+01
STYBLINSKITANG	182,0	0,00000E+00	182,2	0,00000E+00	208,6	0,00000E+00
MICHALEWICZ	181,8	0,00000E+00	182,1	0,00000E+00	209,6	3,30000E-03
RIPPLE1	158,9	0,00000E+00	159,0	0,00000E+00	244,1	0,00000E+00
TRIGOCOMETRIC	178,8	0,00000E+00	178,9	0,00000E+00	202,9	0,00000E+00
QUINTIC	223,6	0,00000E+00	224,8	0,00000E+00	293,3	0,00000E+00
DEB1	159,6	0,00000E+00	160,1	0,00000E+00	239,8	0,00000E+00
ACKLEY1	159,5	0,00000E+00	159,5	0,00000E+00	271,7	0,00000E+00
GRIEWANK	159,6	0,00000E+00	159,7	0,00000E+00	233,9	1,60000E-03
RASTRIGIN	159,5	0,00000E+00	159,2	0,00000E+00	220,3	0,00000E+00
WAVY	159,6	0,00000E+00	159,3	0,00000E+00	193,1	0,00000E+00
SPHERE	159,5	0,00000E+00	159,7	0,00000E+00	231,7	0,00000E+00
SPHERE AT ONE	156,9	0,00000E+00	156,9	0,00000E+00	179,6	0,00000E+00
HOLDERTABLE	186,7	0,00000E+00	187,6	0,00000E+00	194,4	0,00000E+00
AVERAGE RUNS/ PRECISION	182,0	1,67424E-02	182,4	1,69844E-02	227,3	3,53339E+00

XPSO	FTO	UPSO	FTO	MFFA	FTO
193,2	0,00000E+00	215,6	0,00000E+00	287,2	2,41330E-01
197,7	0,00000E+00	227,4	0,00000E+00	285,2	5,03930E-01
185,7	0,00000E+00	202,5	0,00000E+00	278,5	1,87000E-03
178,9	0,00000E+00	191,7	0,00000E+00	287,0	2,60000E-04
213,1	7,91000E-03	236,4	7,70000E-04	314,3	2,11000E-03
193,8	1,60000E-02	212,7	3,40000E-02	223,8	4,82290E-01
213,4	0,00000E+00	259,0	0,00000E+00	265,9	1,86013E+00
178,2	0,00000E+00	190,9	0,00000E+00	296,9	1,50000E-04
211,6	1,66818E+01	294,9	9,16963E+00	286,0	5,57250E-01
399,9	5,14360E-01	408,0	7,96090E-01	284,3	1,20742E+00
219,6	1,55600E-02	245,8	1,39000E-03	330,7	1,03000E-03
220,2	1,53970E+01	245,2	7,58005E+00	293,3	1,65930E+00
186,2	0,00000E+00	203,6	0,00000E+00	286,7	3,12000E-03
187,0	1,60000E-03	204,4	0,00000E+00	291,2	1,71000E-03
368,2	1,60000E-04	435,4	1,00000E-04	255,6	3,02000E-03
182,2	0,00000E+00	201,4	0,00000E+00	275,8	2,00000E-04
227,4	0,00000E+00	288,4	0,00000E+00	549,8	4,94000E-03
206,2	0,00000E+00	249,4	0,00000E+00	266,2	8,10000E-04
219,5	0,00000E+00	266,8	0,00000E+00	445,0	1,06900E-02
236,6	3,17000E-03	295,6	1,70000E-03	289,9	5,67000E-03
203,8	7,96000E-03	228,6	0,00000E+00	293,6	4,12300E-02
183,0	0,00000E+00	200,1	0,00000E+00	302,4	5,50000E-04
197,2	0,00000E+00	225,7	0,00000E+00	306,5	5,64400E-02
170,6	0,00000E+00	177,5	0,00000E+00	182,4	0,00000E+00
194,0	0,00000E+00	213,5	0,00000E+00	365,3	1,70000E-03
214,7	1,30582E+00	244,8	7,03349E-01	301,7	2,65886E-01

Figure 31: Min 2D Long runs with minimum error over all algorithms

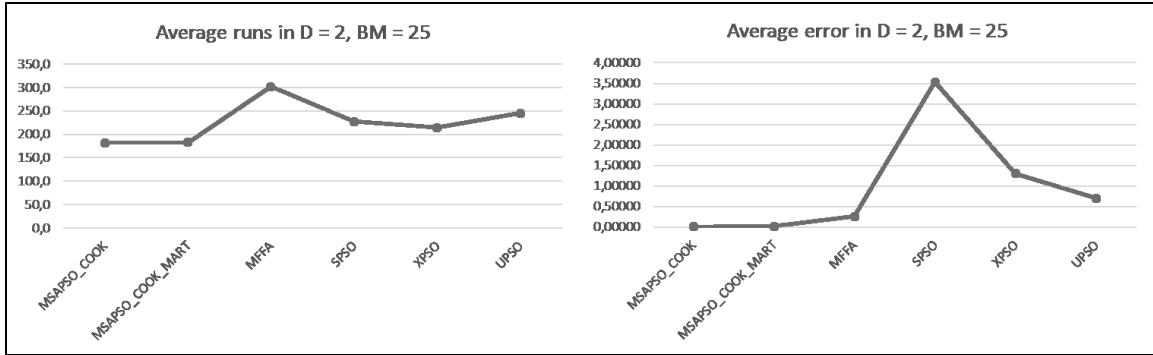


Figure 32: Min 2D Long MSAPSO compared to other algorithms

5.3.3.2 Minimum 5D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
SCHWEFEL	243,3	3,95645E+01	246,7	4,47819E+01	338,7	2,30639E+02
STYBLINSKITANG	215,0	0,00000E+00	215,0	0,00000E+00	265,7	3,78664E+00
MICHALEWICZ	250,4	1,27640E-01	250,1	1,47600E-01	307,3	1,40320E-01
RIPPLE1	289,4	4,00000E-04	295,2	3,30000E-04	576,8	1,30800E-02
TRIGOCOMETRIC	226,4	0,00000E+00	228,8	0,00000E+00	271,5	0,00000E+00
QUINTIC	273,5	0,00000E+00	277,9	0,00000E+00	366,0	0,00000E+00
DEB1	217,1	0,00000E+00	219,8	0,00000E+00	318,7	0,00000E+00
ACKLEY1	231,2	0,00000E+00	227,6	0,00000E+00	340,0	0,00000E+00
GRIEWANK	263,9	2,79000E-03	273,0	2,54000E-01	458,1	3,19700E-02
RASTRIGIN	250,7	0,00000E+00	251,5	0,00000E+00	365,2	7,76070E-01
WAVY	198,7	0,00000E+00	199,8	0,00000E+00	251,8	1,28000E-03
SPHERE	213,6	0,00000E+00	216,8	0,00000E+00	288,3	0,00000E+00
SPHERE AT ONE	178,6	0,00000E+00	179,3	0,00000E+00	212,1	0,00000E+00
AVERAGE RUNS/ PRECISION	234,8	3,30795E+00	237,0	3,76532E+00	335,4	1,96157E+01

XPSO	FTO	UPSO	FTO	MFFA	FTO
334,1	1,87731E+02	392,4	1,55865E+02	305,2	4,66423E+02
236,7	6,78560E-01	259,7	5,65470E-01	314,2	1,57179E+01
272,7	7,85800E-02	375,2	2,43280E-01	325,8	1,27966E+00
393,8	3,71300E-02	487,2	3,15200E-02	273,8	2,31080E-01
249,8	0,00000E+00	262,2	0,00000E+00	303,2	5,75390E-01
299,5	0,00000E+00	340,9	0,00000E+00	1982,2	1,17077E+00
300,2	0,00000E+00	456,6	6,00000E-05	277,7	7,21100E-02
280,7	0,00000E+00	302,1	0,00000E+00	885,6	8,67980E+00
398,5	4,16300E-02	659,5	2,46100E-02	303,3	2,35100E-01
337,9	1,01088E+00	463,3	3,46250E-01	311,3	7,66160E+00
234,4	2,13000E-03	278,5	4,30000E-04	311,0	6,00000E-04
247,4	0,00000E+00	260,4	0,00000E+00	1304,8	1,01290E-01
197,4	0,00000E+00	199,1	0,00000E+00	362,5	6,00000E-05
291,0	1,57983E+01	364,4	1,30897E+01	558,5	4,18457E+01

Figure 33: Min 5D Long runs with minimum error over all algorithms

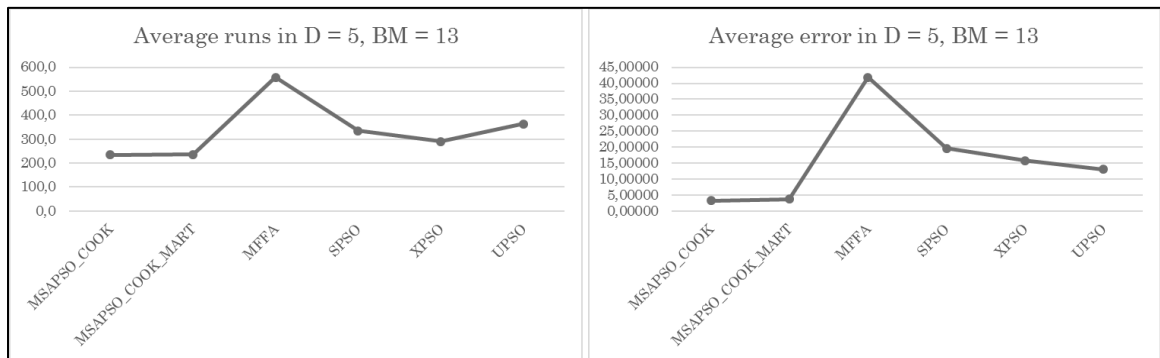


Figure 34: Min 5D Long MSAPSO compared to other algorithms

5.3.3.3 Minimum 10D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
-----------	----------------	-----	------------------------------	-----	------	-----

SCHWEFEL	319,2	1,77100E+02	324,0	1,88489E+02	476,8	7,14615E+02
STYBLINSKITANG	277,0	6,44635E+00	275,6	7,29455E+00	332,6	2,36931E+01
MICHALEWICZ	385,6	6,21380E-01	399,1	5,86240E-01	492,7	1,10240E+00
RIPPLE1	638,6	9,79200E-02	590,1	1,09420E-01	599,9	2,00860E-01
TRIGOCOMETRIC	372,5	0,00000E+00	380,3	4,77461E-01	417,1	0,00000E+00
QUINTIC	357,0	0,00000E+00	364,0	0,00000E+00	471,1	0,00000E+00
DEB1	276,2	0,00000E+00	279,5	0,00000E+00	381,4	0,00000E+00
ACKLEY1	371,4	0,00000E+00	372,6	0,00000E+00	428,2	0,00000E+00
GRIEWANK	406,9	7,15000E-03	410,2	8,45000E-03	419,7	8,37600E-02
RASTRIGIN	386,8	5,97000E-02	402,8	1,15420E-01	477,9	5,28522E+00
WAVY	255,1	0,00000E+00	258,1	0,00000E+00	329,4	3,64700E-02
SPHERE	309,7	0,00000E+00	309,8	0,00000E+00	363,4	0,00000E+00
SPHERE AT ONE	218,1	0,00000E+00	219,4	0,00000E+00	254,6	0,00000E+00
AVERAGE RUNS/ PRECISION	351,9	1,41794E+01	352,7	1,51600E+01	418,8	5,73090E+01

XPSO	FTO	UPSO	FTO	MFFA	FTO
495,0	5,68808E+02	636,8	5,63936E+02	1123,6	1,72791E+03
332,1	6,44560E+00	338,9	1,97914E+01	607,2	5,29056E+01
455,7	5,35700E-01	699,8	1,88136E+00	319,3	4,94244E+00
441,9	2,69780E-01	515,8	2,50880E-01	371,8	3,24370E-01
438,4	0,00000E+00	452,3	8,65300E-01	4958,0	1,65836E+01
420,9	0,00000E+00	452,5	6,86000E-02	1351,2	9,24152E+00
385,2	0,00000E+00	595,2	2,30000E-04	751,8	6,99900E-02
392,1	1,84800E-02	341,6	6,93200E-02	707,1	1,35813E+00
399,6	8,19100E-02	620,3	4,85900E-02	461,5	2,15420E-01
474,2	5,83444E+00	863,2	3,69443E+00	406,0	4,41752E+00
325,4	3,29900E-02	444,8	5,53100E-02	464,5	2,14790E-01
339,1	0,00000E+00	296,7	0,00000E+00	559,6	5,33080E-01
247,1	0,00000E+00	222,9	0,00000E+00	1324,7	0,00000E+00
395,9	4,47713E+01	498,5	4,54355E+01	1031,3	1,39901E+02

Figure 35: Min 10D Long runs with minimum error over all algorithms

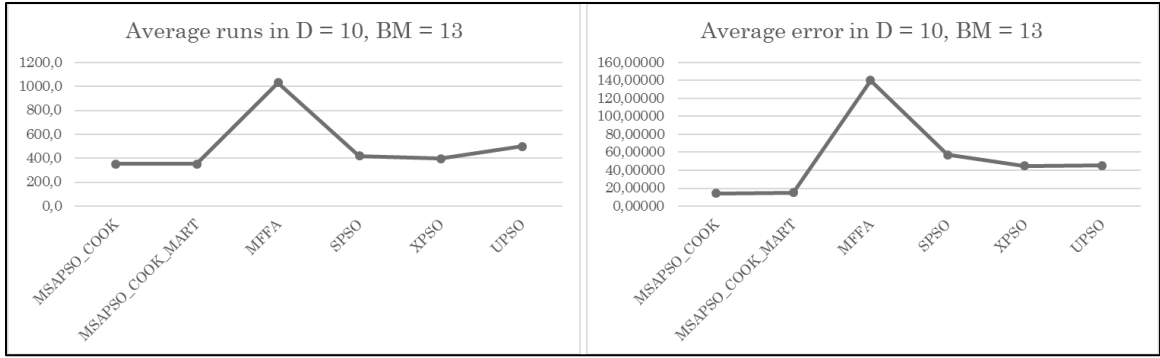


Figure 36: Min 10D Long MSAPSO compared to other algorithms

5.3.3.4 Minimum 30D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
QUINTIC	896,3	8,40000E-03	897,4	0,00000E+00	1485,5	1,40180E-01
DEB1	454,4	0,00000E+00	470,3	0,00000E+00	634,6	6,90000E-04
ACKLEY1	1201,7	0,00000E+00	1244,3	0,00000E+00	684,7	2,23549E+00
GRIEWANK	551,5	0,00000E+00	568,3	0,00000E+00	582,7	1,53600E-02
RASTRIGIN	719,0	2,58690E-01	815,2	1,98990E-01	741,2	4,65680E+01
WAVY	560,0	0,00000E+00	581,0	0,00000E+00	585,6	1,61730E-01
SPHERE	997,0	0,00000E+00	1021,9	0,00000E+00	783,0	0,00000E+00
SPHERE AT ONE	522,0	0,00000E+00	528,3	0,00000E+00	464,7	0,00000E+00
AVERAGE RUNS/ PRECISION	737,7	3,33863E-02	765,8	2,48738E-02	745,3	6,14018E+00
	XPSO	FTO	MFFA	FTO		
	1064,8	0,00000E+00	3438,0	4,85000E+01		
	680,4	0,00000E+00	1657,2	2,17120E-01		
	959,8	4,49600E-01	797,7	1,53842E+00		
	648,0	1,36200E-02	1273,2	1,62100E-02		
	888,7	4,33802E+01	1176,2	1,14095E+01		
	661,5	1,31370E-01	580,4	8,87700E-02		
	845,1	0,00000E+00	2272,2	5,06670E-01		
	510,6	0,00000E+00	1581,8	4,00000E-05		

782,4 5,49684E+00 1597,1 7,78459E+00

Figure 37: Min 30D Long runs with minimum error over all algorithms

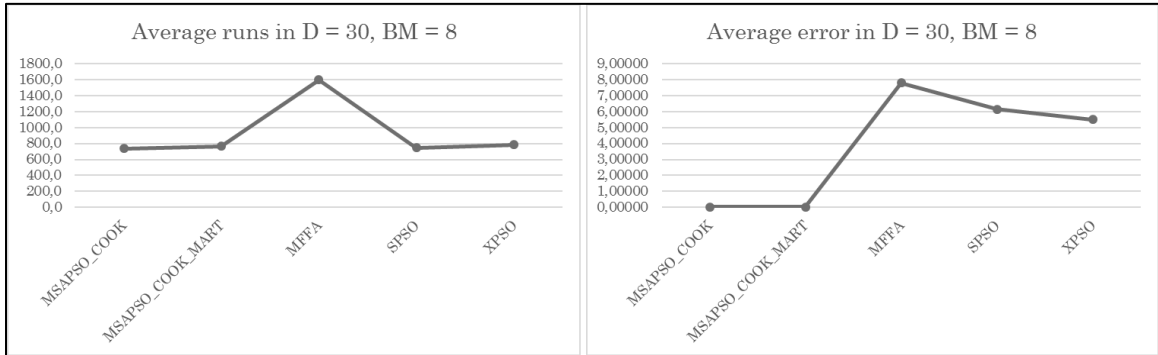


Figure 38: Min 30D Long MSAPSO compared to other algorithms

5.3.3.5 Minimum 100D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
QUINTIC	3846,2	5,66360E-01	4058,3	1,98610E-01	8138,3	1,92171E+01
DEB1	1513,8	1,07000E-03	1586,3	8,60000E-04	2374,4	9,82000E-03
ACKLEY1	2945,3	1,00000E-05	3044,8	1,00000E-05	3002,7	9,90531E+00
GRIEWANK	1275,2	0,00000E+00	1339,8	0,00000E+00	3430,6	9,58900E-02
RASTRIGIN	2144,6	3,45251E+00	2238,3	9,24321E+00	4156,4	2,36237E+02
WAVY	1141,6	0,00000E+00	1171,0	1,00000E-05	60,0	2,51130E-01
SPHERE	2632,2	0,00000E+00	2771,1	0,00000E+00	6198,9	6,06125E+01
SPHERE AT ONE	1250,5	0,00000E+00	1303,4	0,00000E+00	2464,8	9,30000E-03
AVERAGE RUNS/ PRECISION	2093,7	5,02494E-01	2189,1	1,18034E+00	3728,3	4,07923E+01
XPSO	FTO	MFFA	FTO			
	5192,7	5,13200E-02	7398,1	2,16650E+02		
	2181,2	4,10000E-04	4747,0	2,31420E-01		
	4617,7	1,89874E+00	783,9	4,76128E+00		

2742,0	3,63300E-02	1823,5	1,01230E-01
3447,2	2,13647E+02	3088,7	4,11171E+01
2502,2	1,77400E-01	2222,6	1,01250E-01
4234,4	1,00000E-05	7927,9	3,48390E-01
2194,3	3,00000E-05	3372,6	2,80000E-04
3389,0	2,69764E+01	3920,5	3,29139E+01

Figure 39: Min 100D Long runs with minimum error over all algorithms

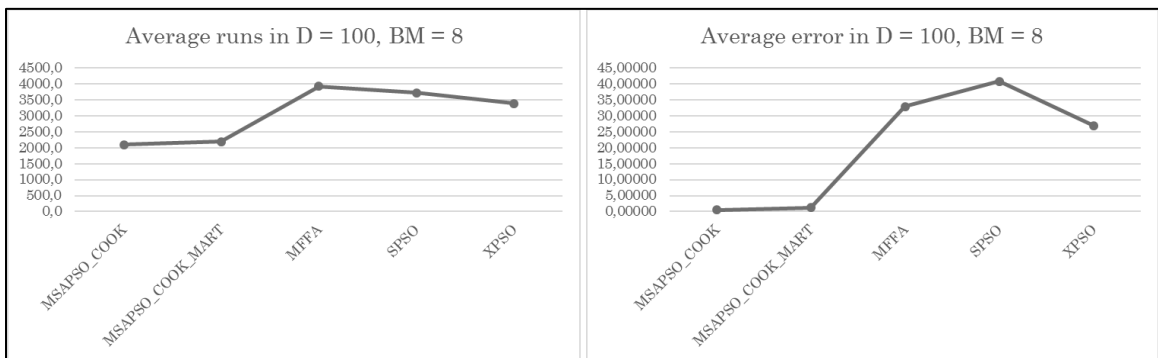


Figure 40: Min 100D Long MSAPSO compared to other algorithms

5.3.3.6 Minimum 250D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	MFFA	FTO
GRIEWANK	1850,3	0,00000E+00	1930,4	1,00000E-05	2960,9	1,66980E-01
RASTRIGIN	4296,9	2,05387E+01	4341,4	3,13322E+01	8169,2	1,77386E+02
WAVY	1399,6	0,00000E+00	1437,8	0,00000E+00	5687,1	7,45900E-02
SPHERE	4141,4	1,00000E-05	4356,8	1,00000E-05	3070,3	8,05225E+01
SPHERE AT ONE	1874,4	0,00000E+00	1992,3	1,00000E-05	6623,5	1,70000E-04
AVERAGE RUNS/ PRECISION	2712,5	4,10774E+00	2811,7	6,26644E+00	5302,2	5,16300E+01

Figure 41: Min 250D Long runs with minimum error over all algorithms

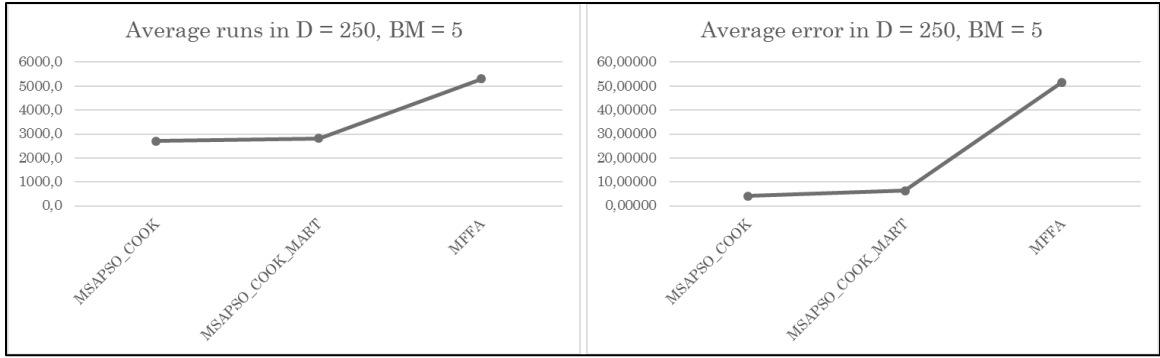


Figure 42: Min 250D Long MSAPSO compared to other algorithms

5.3.3.7 Minimum 500D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	MFFA	FTO
GRIEWANK	2269,2	1,00000E-05	2441,6	1,00000E-05	4599,4	1,89660E-01
RASTRIGIN	9540,2	5,46632E+01	9090,8	8,29688E+01	32565,0	1,89043E+02
WAVY	1532,6	1,00000E-05	1606,1	1,00000E-05	10085,2	7,43900E-02
SPHERE	5150,1	2,00000E-05	5636,4	3,00000E-05	4659,5	1,32408E+02
SPHERE AT ONE	2357,1	1,00000E-05	2524,1	1,00000E-05	10978,6	1,00000E-03
AVERAGE RUNS/ PRECISION	4169,8	1,09327E+01	4259,8	1,65938E+01	12577,5	6,43433E+01

Figure 43: Min 500D Long runs with minimum error over all algorithms

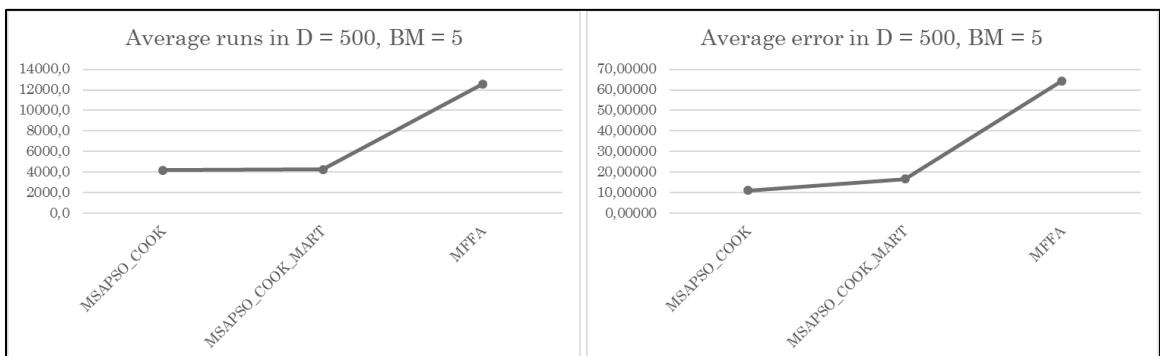


Figure 44: Min 500D Long MSAPSO compared to other algorithms

5.3.4 Benchmarks and Maximum Tests

BENCHMARK	MAXIMA GBESTVAL F(X _n)	MAXIMA GBEST X _n
BIRD	1,75683E+02	MULTIPLE
BOHACHEVSKY	3,00000E+04	(+/- 100,...100)
BOOTH	2,59400E+03	(-10,...-10)
CAMEL	2,04792E+03	(+/-5,...5)
DROPWAVE	0,00000E+00	MULTIPLE
EASOM	9,01000E-03	(1,30500/3.145927,...3.145927/1,30500)
SHUBERT	2,10482E+02	MULTIPLE
ZETTL	3,59875E+03	(-5,...-/5)
EGGHOLDER	1,04913E+03	(-512,...,512)
RANA	5,13000E+02	MULTIPLE
SALOMON	2,03730E+01	(+/- 10,...10)
SCHWEFEL	8,37966E+02	(-420.96874,...-420.96874)
STYBLINSKITANG	2,50000E+02	(5,...5)
MICHALEWICZ	1,80130E+00	(-2.20290,...-1.57079)
RIPPLE1	1,62000E+01	(-0,5,...-0,5)
TRIGOCOMETRIC	1,46599E+02	(-2,98696,...-2,67795)
QUINTIC	2,67408E+05	(-10,...-10)
DEB1	0,00000E+00	MULTIPLE
ACKLEY1	2,23203E+01	(+/- 32,50041...,32,50041)
GRIEWANK	3,18770E+00	(+/- 50...,48,92242)
RASTRIGIN	0,00000E+00	(+/-4,52299,...,4,52299)
WAVY	1,82711E+00	(+/-0,60426,...,0,60426)
SPHERE	5,00000E+05	(+/- 500,...500)
SPHERE AT ONE	2,00000E+00	(0..0/0..0,...,2..2/2..2)
HOLDERTABLE	0,00000E+00	MULTIPLE

Figure 45: List of benchmarks MAX-TEST in respective dimensions

5.3.4.1 Maximum 2D Long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
BIRD	178,4	0,00000E+00	178,9	0,00000E+00	175,3	6,63285E+00
BOHACHEVSKY	152,9	0,00000E+00	153,0	0,00000E+00	153,0	0,00000E+00
BOOTH	153,5	0,00000E+00	153,5	0,00000E+00	153,4	0,00000E+00
CAMEL	153,7	0,00000E+00	153,7	0,00000E+00	153,7	0,00000E+00
DROPWAVE	161,3	0,00000E+00	162,2	0,00000E+00	169,9	0,00000E+00
EASOM	183,9	0,00000E+00	184,4	0,00000E+00	209,9	2,00000E-05
SHUBERT	204,1	0,00000E+00	205,3	0,00000E+00	251,9	0,00000E+00
ZETTL	153,2	0,00000E+00	153,2	0,00000E+00	153,3	0,00000E+00
EGGHOLDER	163,7	0,00000E+00	162,8	0,00000E+00	202,6	6,21012E+01
RANA	381,7	5,42250E-01	381,8	4,54700E-01	497,5	6,73080E-01
SALOMON	152,9	0,00000E+00	153,0	0,00000E+00	153,0	0,00000E+00
SCHWEFEL	194,8	0,00000E+00	195,6	0,00000E+00	251,0	4,64278E+01
STYBLINSKITANG	157,1	0,00000E+00	157,0	0,00000E+00	158,2	9,85000E+00
MICHALEWICZ	181,3	0,00000E+00	182,5	0,00000E+00	209,2	1,60000E-03
RIPPLE1	678,6	1,96000E-03	718,2	2,09000E-03	867,4	2,94000E-03
TRIGOCOMETRIC	186,0	0,00000E+00	186,9	0,00000E+00	215,1	4,95050E-01
QUINTIC	154,8	0,00000E+00	154,9	0,00000E+00	157,3	1,04910E+04
DEB1	152,3	0,00000E+00	152,3	0,00000E+00	152,4	0,00000E+00
ACKLEY1	289,3	5,90000E-04	292,6	3,80000E-04	423,7	1,01000E-03
GRIEWANK	176,1	0,00000E+00	177,2	0,00000E+00	198,3	0,00000E+00
RASTRIGIN	189,5	0,00000E+00	190,5	0,00000E+00	230,3	0,00000E+00
WAVY	179,6	0,00000E+00	179,8	0,00000E+00	203,6	0,00000E+00
SPHERE	152,9	0,00000E+00	152,9	0,00000E+00	153,0	0,00000E+00
SPHERE AT ONE	152,9	0,00000E+00	152,8	0,00000E+00	152,9	0,00000E+00
HOLDERTABLE	153,2	0,00000E+00	153,2	0,00000E+00	153,6	0,00000E+00
AVERAGE RUNS/ PRECISION	201,5	2,17920E-02	203,5	1,82868E-02	232,0	4,24688E+02

XPSO	FTO	UPSO	FTO	MFFA	FTO
168,2	6,60284E+00	211,5	5,36640E-01	294,8	3,64000E-03
152,7	0,00000E+00	153,3	0,00000E+00	160,7	0,00000E+00
153,1	0,00000E+00	153,8	0,00000E+00	165,8	0,00000E+00
153,3	0,00000E+00	154,2	0,00000E+00	167,7	0,00000E+00
168,0	0,00000E+00	169,6	0,00000E+00	156,2	0,00000E+00
187,6	1,70000E-04	206,6	3,30000E-04	244,1	4,36000E-03
217,1	0,00000E+00	254,9	0,00000E+00	262,2	4,02323E+00
152,9	0,00000E+00	153,6	0,00000E+00	163,6	0,00000E+00
178,7	3,91936E+01	194,7	1,87736E+01	153,5	0,00000E+00
400,7	5,26140E-01	406,6	7,95680E-01	273,4	1,36533E+00
152,7	0,00000E+00	153,2	0,00000E+00	192,0	0,00000E+00
221,3	1,80026E+01	251,3	8,05380E+00	296,3	0,00000E+00
155,3	8,15000E+00	163,9	7,00000E-01	153,5	0,00000E+00
187,3	0,00000E+00	204,9	0,00000E+00	271,6	2,22000E-02
677,0	2,66000E-03	192,5	1,84780E-01	164,1	2,22000E-02
187,2	6,88400E-01	214,3	2,42900E-02	290,7	1,80230E-01
154,9	9,89493E+03	158,1	0,00000E+00	153,3	0,00000E+00
152,3	0,00000E+00	152,7	0,00000E+00	152,7	0,00000E+00
348,1	8,00000E-04	317,2	1,02000E-03	557,1	2,31100E-02
183,0	0,00000E+00	175,4	0,00000E+00	254,6	3,00000E-05
198,7	0,00000E+00	213,1	0,00000E+00	268,5	2,57000E-02
183,4	0,00000E+00	206,7	0,00000E+00	278,7	4,50000E-04
152,8	0,00000E+00	153,4	0,00000E+00	156,5	0,00000E+00
152,7	0,00000E+00	153,3	0,00000E+00	152,0	0,00000E+00
190,9	0,00000E+00	175,1	0,00000E+00	253,8	3,00000E-05
209,2	3,98724E+02	197,8	1,16281E+00	225,5	2,26820E-01

Figure 46: Max 2D Long runs with minimum error over all algorithms

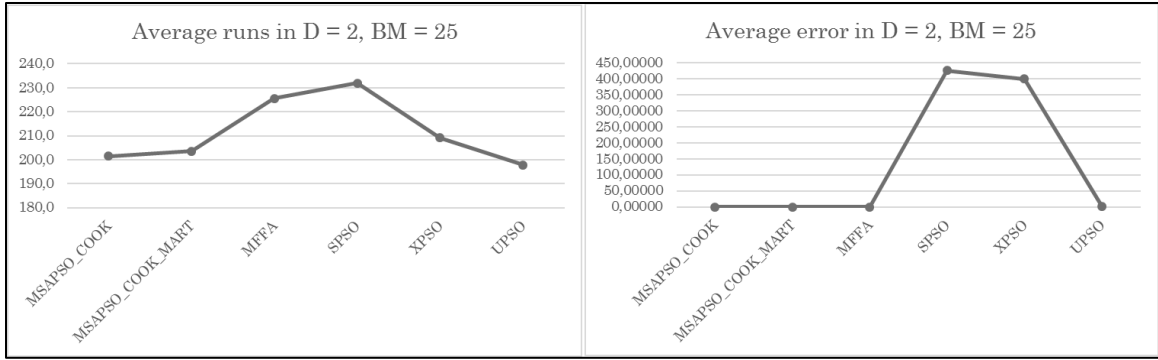


Figure 47: Max 2D Long MSAPSO compared to other algorithms

5.3.4.2 Maximum 5D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
SCHWEFEL	245,4	4,82560E+01	245,3	4,68287E+01	336,0	2,34034E+02
STYBLINSKITANG	187,9	8,50000E+00	195,8	6,30000E+00	185,5	4,62977E+01
MICHALEWICZ	254,2	1,46060E-01	251,5	1,61870E-01	310,8	1,60740E-01
RIPPLE1	509,7	1,09930E-01	514,8	7,96600E-02	676,2	1,28850E-01
TRIGOCOMETRIC	236,4	1,19905E+00	240,0	1,55983E+00	339,5	3,82997E+01
QUINTIC	182,6	4,76900E+03	178,4	5,24550E+03	182,6	7,41524E+04
DEB1	159,0	0,00000E+00	158,1	0,00000E+00	170,8	0,00000E+00
ACKLEY1	555,0	1,42200E-02	574,5	1,48900E-02	838,7	2,20000E-02
GRIEWANK	222,8	0,00000E+00	225,6	0,00000E+00	340,9	0,00000E+00
RASTRIGIN	229,7	3,21600E-02	232,0	0,00000E+00	309,6	3,21600E-02
WAVY	207,5	5,20000E-04	208,5	6,20000E-04	252,9	0,00000E+00
SPHERE	159,3	0,00000E+00	158,4	0,00000E+00	167,6	0,00000E+00
SPHERE AT ONE	155,3	0,00000E+00	155,3	0,00000E+00	168,6	0,00000E+00
AVERAGE RUNS/ PRECISION	254,2	4,02271E+02	256,8	4,41704E+02	329,2	6,20594E+03
	XPSO	FTO	UPSPO	FTO	MFFA	FTO
	327,2	1,72859E+02	409,8	1,18438E+02	354,7	6,34225E+02
	171,7	3,99000E+01	210,2	2,76927E+01	205,5	2,16667E+01

272,6	9,00500E-02	390,2	2,41610E-01	277,2	1,84627E+00
510,9	9,24800E-02	257,8	9,46340E-01	306,8	9,71260E-01
260,2	3,43932E+01	480,7	3,22799E+00	254,6	4,64060E+01
173,0	5,60315E+04	186,0	3,64928E+04	237,6	2,41089E+04
167,1	0,00000E+00	191,7	0,00000E+00	233,5	3,00000E-05
732,9	1,43800E-02	855,1	2,63600E-02	297,5	1,19920E-01
252,9	0,00000E+00	252,9	1,73900E-02	269,4	2,10110E-01
261,2	0,00000E+00	270,1	2,65420E-01	318,5	9,09359E+00
229,0	0,00000E+00	206,4	0,00000E+00	414,2	7,79800E-02
159,6	0,00000E+00	155,7	0,00000E+00	196,9	0,00000E+00
160,0	0,00000E+00	155,8	0,00000E+00	152,0	0,00000E+00
282,9	4,68991E+03	309,4	3,05364E+03	270,6	2,06863E+03

Figure 48: Max 5D Long runs with minimum error over all algorithms

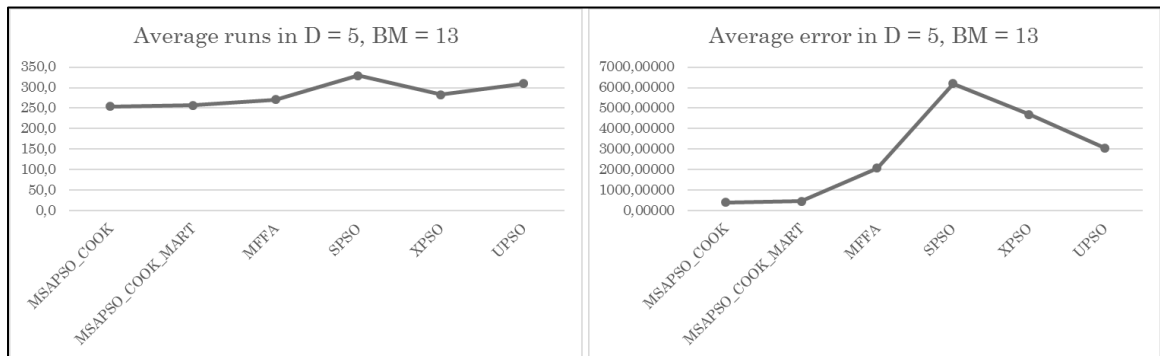


Figure 49: Max 5D Long MSAPSO compared to other algorithms

5.3.4.3 Maximum 10D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
SCHWEFEL	313,4	1,97508E+02	321,5	2,12407E+02	469,4	7,67676E+02
STYBLINSKITANG	336,7	5,83578E+01	331,7	5,61601E+01	873,6	1,34539E+02
MICHALEWICZ	385,6	7,64480E-01	402,4	5,99900E-01	492,8	1,08687E+00
RIPPLE1	559,3	1,41876E+00	546,5	1,10224E+00	828,9	3,27981E+00
TRIGOCOMETRIC	304,1	3,32665E+02	311,4	3,79197E+02	694,6	4,60773E+02

QUINTIC	367,6	3,89223E+04	367,6	3,71954E+04	1111,1	1,82993E+05
DEB1	175,4	0,00000E+00	174,3	0,00000E+00	369,1	0,00000E+00
ACKLEY1	737,4	3,27800E-02	795,3	3,38900E-02	1122,9	5,55500E-02
GRIEWANK	286,9	2,25230E-01	294,9	2,32820E-01	540,3	6,78400E-02
RASTRIGIN	308,0	2,23535E+00	315,6	1,49560E+00	448,6	2,73388E+00
WAVY	250,8	1,62000E-03	253,6	1,03000E-03	330,1	9,30000E-03
SPHERE	359,3	0,00000E+00	353,2	0,00000E+00	1081,6	1,79500E-02
SPHERE AT ONE	156,1	0,00000E+00	156,2	0,00000E+00	634,6	0,00000E+00
AVERAGE RUNS/ PRECISION	349,3	3,03966E+03	355,7	2,91128E+03	692,1	1,41818E+04

XPSO	FTO	UPSO	FTO	MFFA	FTO
503,1	5,69817E+02	664,4	5,74832E+02	294,3	1,81924E+03
426,2	9,75984E+01	194,2	9,85461E+01	297,9	2,11084E+02
447,4	4,95480E-01	690,6	1,82330E+00	266,4	6,28479E+00
494,0	4,68940E-01	399,1	9,65221E+00	250,7	2,65883E+01
416,1	4,44722E+02	502,9	4,69713E+02	285,0	2,11000E+03
506,1	1,43536E+05	186,5	1,78135E+05	280,4	2,68645E+05
304,4	0,00000E+00	384,1	0,00000E+00	371,1	1,73600E-02
1057,5	3,69200E-02	1072,2	7,50500E-02	285,0	2,76380E-01
375,5	1,88890E-01	232,0	5,17430E-01	288,4	7,14250E-01
375,8	4,50290E-01	370,2	1,23993E+01	236,9	9,15596E+01
307,3	2,60000E-04	460,4	2,58800E-02	593,3	6,20800E-02
503,3	0,00000E+00	158,0	0,00000E+00	294,5	4,79370E+04
335,4	0,00000E+00	158,4	0,00000E+00	152,0	0,00000E+00
465,5	1,11269E+04	421,0	1,37925E+04	299,7	2,46806E+04

Figure 50: Max 10D Long runs with minimum error over all algorithms

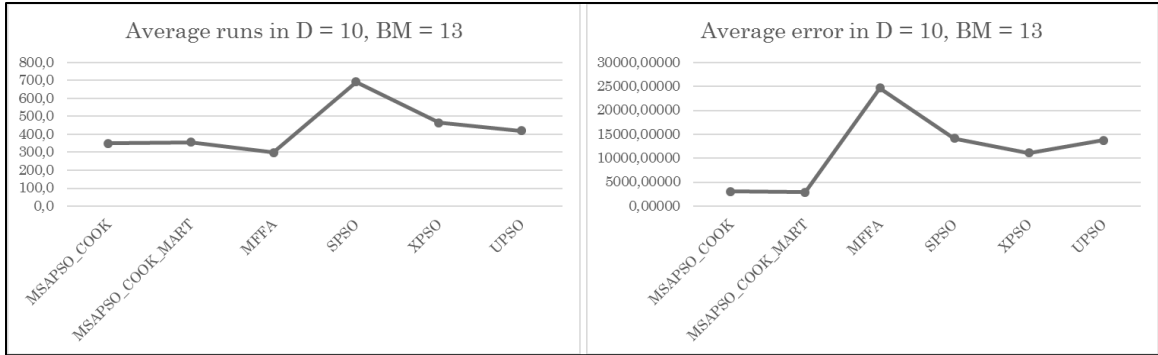


Figure 51: Max 10D Long MSAPSO compared to other algorithms

5.3.4.4 Maximum 30D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
QUINTIC	917,3	6,51498E+04	953,3	6,65225E+04	6199,0	7,51442E+05
DEB1	335,2	0,00000E+00	350,2	0,00000E+00	796,3	0,00000E+00
ACKLEY1	981,5	7,13300E-02	1092,5	6,92700E-02	1575,5	1,33620E-01
GRIEWANK	540,4	0,00000E+00	554,7	0,00000E+00	2845,7	1,03260E-01
RASTRIGIN	656,9	4,45957E+01	685,1	4,36503E+01	857,7	1,40440E+02
WAVY	450,0	1,37700E-02	463,1	1,36600E-02	590,0	9,23000E-02
SPHERE	901,7	0,00000E+00	939,9	0,00000E+00	6382,1	2,55195E+04
SPHERE AT ONE	156,0	0,00000E+00	155,6	0,00000E+00	3013,1	1,22900E-01
AVERAGE RUNS/ PRECISION	617,4	8,14932E+03	649,3	8,32078E+03	2782,4	9,71379E+04

XPSO	FTO	MFFA	FTO
2018,2	4,76626E+05	263,4	1,67740E+06
914,8	2,60000E-04	744,6	2,76100E-02
1441,9	6,86000E-02	259,9	4,72700E-01
1113,4	0,00000E+00	259,6	3,75221E+00
929,8	4,54222E+01	269,6	3,77137E+02
641,6	1,87000E-02	906,8	1,67970E-01
2017,8	0,00000E+00	321,2	1,42383E+06
1152,0	0,00000E+00	152,0	0,00000E+00

1278,7 5,95839E+04 397,1 3,87701E+05

Figure 52: Max 30D Long runs with minimum error over all algorithms

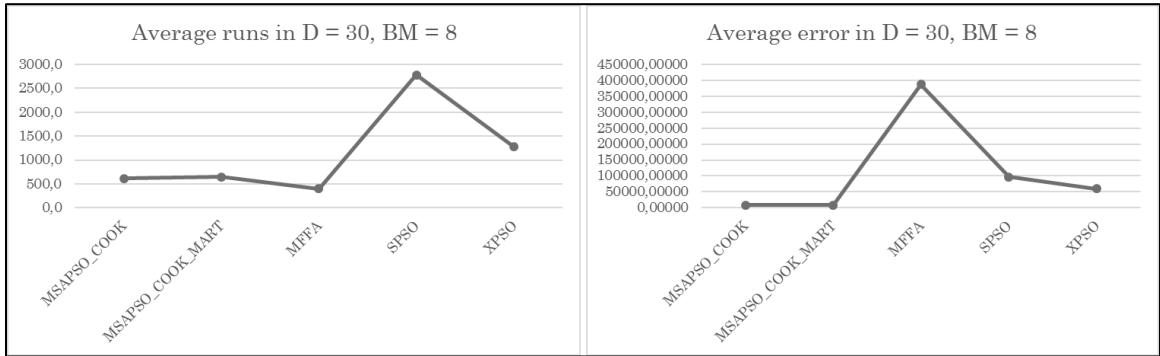


Figure 53: Max 30D Long MSAPSO compared to other algorithms

5.3.4.5 Maximum 100D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	SPSO	FTO
QUINTIC	5341,2	5,05870E+04	5788,0	4,59612E+04	233781,1	5,36990E+06
DEB1	1137,8	1,00000E-05	1284,9	2,00000E-05	1984,9	4,00000E-05
ACKLEY1	1900,9	9,95000E-02	2067,1	9,87800E-02	2960,3	2,62870E-01
GRIEWANK	2661,6	4,05000E-03	2857,9	4,12000E-03	7313,2	1,31046E+01
RASTRIGIN	2954,6	2,78862E+02	3084,2	2,69953E+02	4117,0	7,65945E+02
WAVY	1601,6	1,95000E-02	1607,8	1,36400E-02	2202,6	1,60890E-01
SPHERE	5269,9	3,01560E-01	5800,6	2,14000E-03	29891,9	3,77800E+06
SPHERE AT ONE	155,5	0,00000E+00	155,9	0,00000E+00	7587,2	2,11713E+01
AVERAGE RUNS/ PRECISION	2627,9	6,35829E+03	2830,8	5,77891E+03	36229,8	1,14359E+06

XPSO	FTO	MFFA	FTO
28083,0	1,41810E+06	164,7	1,08572E+07
2803,0	2,03000E-03	151,3	8,49930E-01
2984,3	1,86870E-01	1339,3	5,94000E-02
11007,3	1,33090E-01	293,0	2,08639E+01
3873,9	3,85082E+02	271,3	1,43503E+03
2407,8	2,96900E-02	3413,5	6,16200E-02
28684,2	2,67494E+03	528,5	1,51291E+07
12070,3	1,77800E-01	160,0	0,00000E+00
11489,2	1,77645E+05	790,2	3,24847E+06

Figure 54: Max 100D Long runs with minimum error over all algorithms

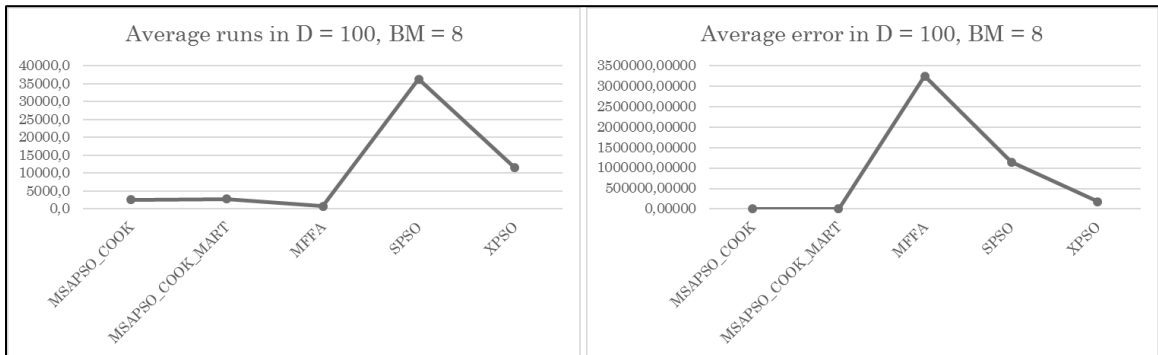


Figure 55: Max 100D Long MSAPSO compared to other algorithms

5.3.4.6 Maximum 250D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	MFFA	FTO
GRIEWANK	12188,7	1,82028E+00	14295,0	1,88474E+00	280,3	6,29300E+01
RASTRIGIN	12043,5	2,51779E+02	12977,9	8,04825E+01	255,7	3,09820E+03
WAVY	4758,4	1,52300E-02	5344,8	1,83600E-02	9529,7	6,37100E-02
SPHERE	22205,3	8,88453E+04	26506,2	1,15230E+05	168,0	3,93130E+07
SPHERE AT ONE	156,1	0,00000E+00	155,7	0,00000E+00	162,0	0,00000E+00

AVERAGE RUNS/
PRECISION 10270,4 1,78198E+04 11855,9 2,30625E+04 2079,1 7,86324E+06

Figure 56: Max 250D Long runs with minimum error over all algorithms

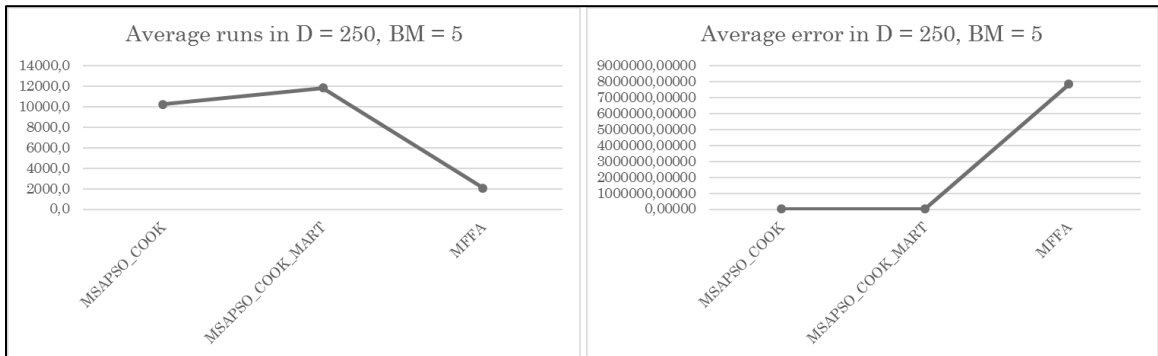


Figure 57: Max 250D Long MSAPSO compared to other algorithms

5.3.4.7 Maximum 500D long runs with minimal runs and error

BENCHMARK	MSAPSO COOK	FTO	MSAPSO COOK & MARTINEZ	FTO	MFFA	FTO
GRIEWANK	26347,1	1,12856E+01	31263,5	1,25000E+01	265,8	1,80599E+02
RASTRIGIN	23605,5	1,34904E+03	27964,0	9,63675E+02	257,7	6,48999E+03
WAVY	9801,9	7,48600E-02	10755,6	6,41400E-02	16620,0	5,81000E-02
SPHERE	39911,2	2,73565E+06	47481,1	3,16124E+06	250,6	5,31081E+07
SPHERE AT ONE	155,8	0,00000E+00	155,9	0,00000E+00	500,0	0,00000E+00
AVERAGE RUNS/ PRECISION	19964,3	5,47402E+05	23524,0	6,32442E+05	3578,8	1,06230E+07

Figure 58: Max 500D Long runs with minimum error over all algorithms

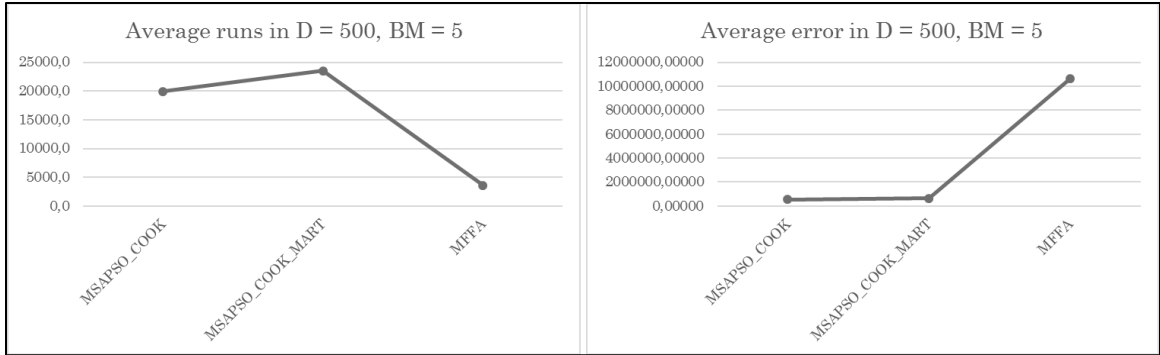


Figure 59: Max 500D Long MSAPSO compared to other algorithms

5.4 Comparison MSAPSO & MFFA with and without ELM

5.4.1 Sphere Function without ELM

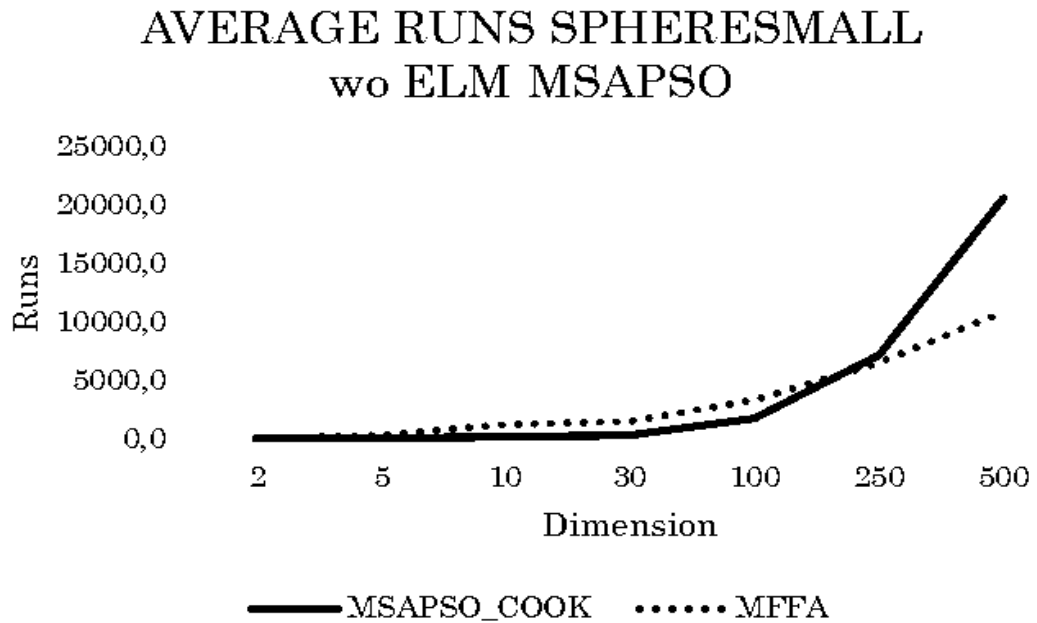


Figure 60: Min 2D-500D Long runs with Sphere Function comparison wo ELM

AVERAGE ERROR SPHERESMALL wo ELM MSAPSO

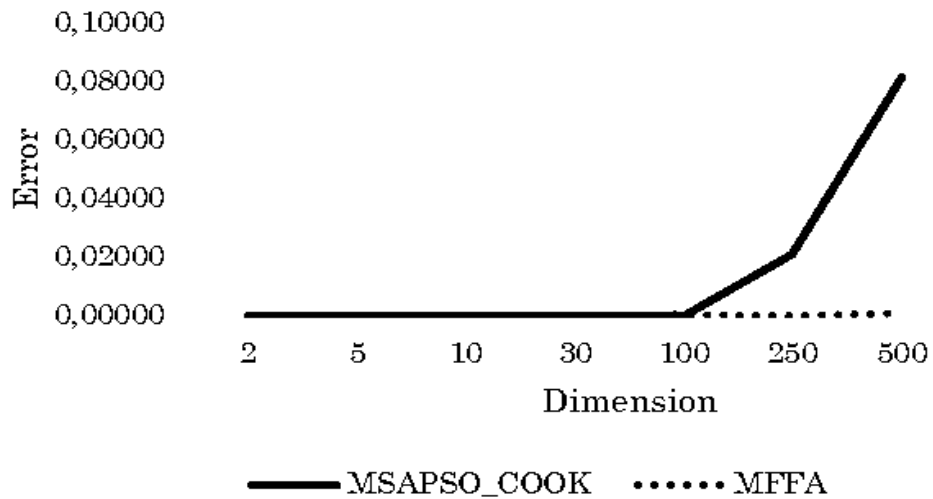


Figure 61: Min 2D-500D Min error with Sphere Function comparison wo ELM

5.4.2 Sphere Function with ELM

AVERAGE RUNS SPHERESMALL with ELM MSAPSO

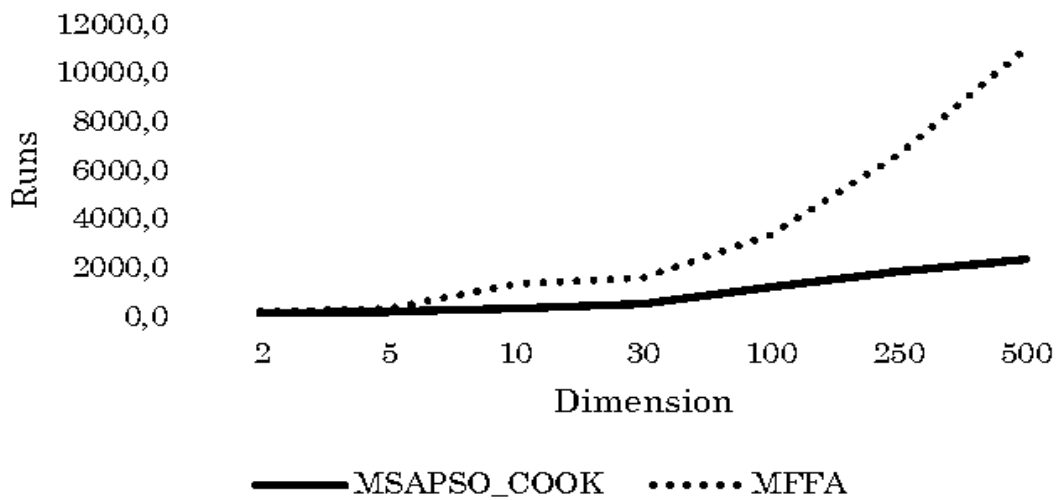


Figure 62: Min 2D-500D Long runs with Sphere Function comparison with ELM

AVERAGE ERROR SPHERESMALL with ELM MSAPSO

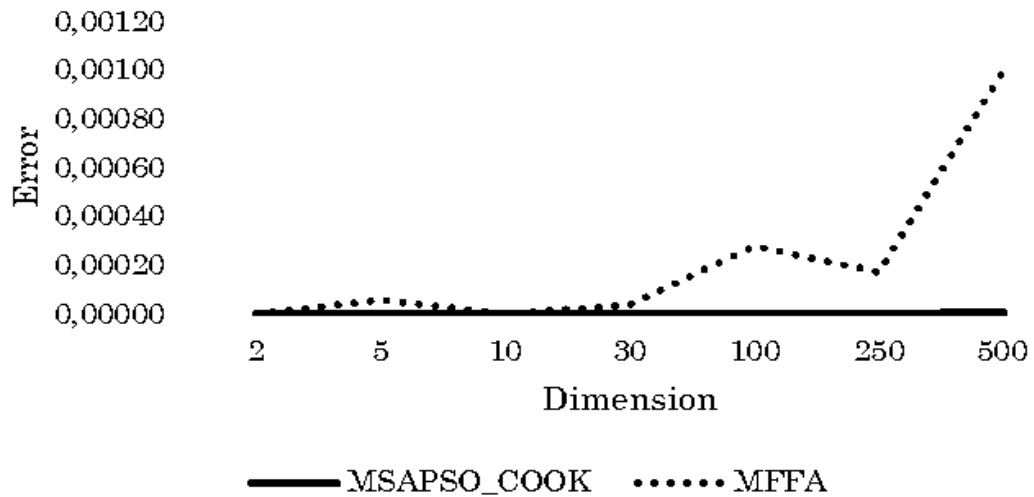


Figure 63: Min 2D-500D Min error with Sphere Function comparison with ELM

5.4.3 Referenced Function comparison MSAPSO versus MFFA

In order to check with officially available performance data from MFFA tests, the following source (Yang, Firefly for Multimodal Optimization, 2010, p. 9) is referenced to check that the previously made performance and stability tests for MSAPSO and MFFA are valid and consistent.

As an example, the De Jong Function and the Ackley Function is used. In Yang’s paper the following performance figures are reported (column two and three). Right to the official reported test, the evaluations within the PhD evaluations are reported (column four, five and six).

The preciseness of the comparison is about $\leq 10^{-5}$ both for the Yang test as well as for the PhD test. The numbers under the respective algorithm names in the table reflect the average runs achieved. The percentage values in brackets is the rate of convergence to the real global optima.

Referenced Function	PSO (Yang)	MFFA (Yang)	PSO (PhD Test)	MFFA (PhD Tests)	MSAPSO (PhD Tests)
De Jong Function 1 D = 256	17040 (100%)	7217 (100%)	14989 (100%)	7369 (99%)	2167 (100%)
Ackley Function 1 D = 128	23407 (92%)	5293 (100%)	19589 (90%)	4853 (96%)	3335 (100%)

Figure 64: Official Performance Data from Yang compared to MSAPSO tests

Some explanation is required for the small differences in the test result between the numbers reported in the Yang Tests versus the PhD Tests, although the preciseness setting is the same in both tests cases. As the PSO in the PhD implementation is a van Neumann topology the performance figures are slightly better than in the Yang tests. Secondly as the

parameter setting for MFFA in the Yang tests are unknown, in the PhD Tests we have chosen the MFFA parameters as following:

MFFA setting for De Jong Function 1: $\alpha = 0.0175$ $\beta = 0.05$ $\gamma = 0.05$

MFFA setting for Ackley Function 1: $\alpha = 0.0002$ $\beta = 0.50$ $\gamma = 1.00$

There is also not an exact statement with regard to the number of particles and fireflies used in the Yang tests. In the PhD test this value is set to thirty. Although there is some uncertainty with regard to parameter configuration details in MFFA as well as the implementation details with regard to the PSO topology used in the Yang tests the tendency in the comparative tests seem to confirm the PhD results when reflecting it with the official tests from Yang.

Finally, also in this direct comparison MSAPSO outperforms MFFA (PhD test) significantly both in high dimensional unimodal and multimodal functions (column five and six). The average performance benefit of MSAPSO is about fifty percent in the combined test case.

5.5 Summary Evaluations Results

- In the tests of the variation of inertia weight, sum of social and cognition and the used probability distribution, MSAPSO with MSL seems to be an optimal strategy with regard to minimal total averaged runs and minimal total averaged error based in the tested benchmarks.

- Furthermore, MSAPSO still have a lot of flexibility built in, along the specific set of convergence curves, represented by the MSF and the sliding concept of the inertia weight value. Also, this flexibility can be used for example in escape-lmin-optima situations.
- For the case of the detailed MSAPSO tests, the MSF is used, as well as the escape-lmin-strategy is turned on. Then we have the following results for the minima and maxima test:
 - In 2D: MSAPSO is significantly superior compared to all other algorithms tested both with regard to total averaged runs and total averaged error.
 - In 5D: MSAPSO is significantly superior compared to all other algorithms tested both with regard to total averaged runs and total averaged error.
 - In 10D: MSAPSO is significantly superior compared to all other algorithms tested both with regard to total averaged runs and total averaged error.
 - In 30D: MSAPSO is significantly superior compared to all other algorithms tested both with regard to total averaged runs and total averaged error. With 30D the number of comparative algorithms and benchmarks needs to be reduced.
 - In 100D-500D: MSAPSO is significantly superior compared to all other remaining algorithms tested both with regard to total averaged runs and total averaged error. With dimensions of 100D-500D the number of comparative algorithms and benchmarks further needs to be reduced, because most algorithms lose the capability to converge in a stable way. Just three algorithms remain with MSAPSO, MSAPPSO_FERNANDEZ and MFFA.

- In comparison with MFFA and with escape-lmin-strategy turned off MSAPSO is superior over MFFA up to round about hundred thirty dimensions (see sphere function tests). From there on MFFA do have a better performance and stability.
- When escape-lmin-strategy in MSAPSO is turned on again, then it is superior also compared to MFFA in higher dimensions.

6 RESEARCH CONTRIBUTION OF MSAPSO

The general and specific research and knowledge contribution of the MSAPSO algorithm is summarized in the below table and sorted by the order of importance and relevance.

General Theme	Specific Research Contribution
<p>General Optimizer</p> <p>with the concept of a “parameter-less” and “self-adaptive” swarm concept, which is agnostic to the underlying benchmark problems and the used probability distributions.</p>	<p>Creation of a new stability criteria (MSL) based on the saddle point of the set of specific convergence curves, which forms a new way to get independent from chosen inertia weight, social, cognition parameter settings and the used probability distribution.</p> $\mu_{\varphi_{\text{stable}}} = (\mu_w + 1)^2$
	<p>Understanding of the relation between MSAPSO order-1 and order-2 convergence room and the order-2 collapse into order-1 zone.</p> <p>Please refer to chapter 3.6.2</p> <p>Mathematical proof of order-1 order-2 convergence zone collapse.</p>

Theoretical understanding of the optimal start inertia weight value of MSAPSO.

Please refer to: chapter 4.5.1

Reasoning of MSAPSO start inertia weight

Specific convergence curve of MSAPSO and the understanding that for uniform and the normal distribution we have the same convergence curve, when we have the same average value and the same standard deviation independent of the type of the probability distribution:

Please refer to: chapter 3.5.4

Application of convergence study to MSAPSO

Understanding, how average and variance based probability distributions can be used to control the level of exploration and exploitation in MSAPSO.

Please refer to:

Figure 17: Raised convergence and stability curve with $N(0.5,0.075)$

Flexible Escape Local Optima Strategy

A HYPERCUBE-, LOCALCUBE-, and GRAVITATIONAL CUBE Escape Local Optima strategy, which works “dimension-less” and

with the concept of a self-adaptive and N – independent from the underlying benchmark dimensional benchmark agnostic escape strategy problem structure. It is flexible in nature such that different portions of the three strategies will be applied for different benchmark types.

Please refer to: chapter

Escape Local Minima Strategy

Dynamic Characterization of the Search Space as a prerequisite to efficiently escape local optima's as well as dynamic inertia weight strategy around optimal inertia weight point.

Please refer to: chapter 4.8.1

For the above research contribution, we have the following limitations:

- MSAPSO convergence study is only valid with probability distributions that have an average value and a corresponding variance.
- For the case of, e.g. Cauchy- and Levy distribution the stability criteria are not applicable, because of the lack of an average value and variance in these distributions, therefore it would be worthwhile to investigate a convergence analysis to embed this into the MSAPSO algorithm.

7 FUTURE RESEARCH

With regard to the MSAPSO there are various areas of future research which can be worked on. In order to outline these fields, we need also to understand what other areas such as Artificial Intelligence (AI) do have as a problem: in that context, the following fields specific to MSAPSO or related to other intelligent bio-inspired algorithms can be named:

In general:

- Deep understanding of the mathematics of other nature-inspired algorithms (convergence behavior, parameter influence on the algorithms).
- Understand self-parameterizing concepts which has their roots in biology (brain function, bio-inspired system in general).
- Criteria's for combining hybrid algorithms and methods to understand their respective influence into the overall results (performance, error-proneness)
- Get a better understanding why dynamical system such as MSAPSO do aim for emergent behavior.

Specific to MSAPSO:

- Analyze the mathematical criteria's and relations of probability distributions used and their influence on the optimal balance points, in the context when to use what kind of a probability distribution when facing different types of benchmark problems.
- In this context, also understand how non-averaged & non-variance value probability distributions such as Cauchy and Levy distributions can be integrated into a self-parametrizing model of MSAPSO.
- Mathematical Analysis of the 3rd (kurtosis) and 4th (skewness) statistical moment and their influence on the MSAPSO convergence analysis shown in this PhD document.
- A more detailed analysis on the particular influence of social and cognition at the sum of both with regard to unimodal and multimodal functions, also in the context of high dimensional benchmark problems.

REFERENCES

- Ashby, R. (1962). Principles of the self-organizing system. 255-278. Retrieved from <http://csis.pace.edu/~marchese/CS396x/Computing/Ashby.pdf>
- Bai, Q. (2010, February). Analysis of Particle Swarm Optimization. Retrieved from <http://www.ccsenet.org/journal/index.php/cis/article/view/5131/4314>
- Bandura, A. (1986). Social Cognitive Theory. Stanford, California, United States.
Retrieved from <http://www.uky.edu/~eushe2/Bandura/Bandura1989ACD.pdf>
- Brits, R., Engelbrecht, A. P., & van den Bergh, F. (2002). Niching PSO. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.65.6137&rep=rep1&type=pdf>
- Camazine, S., Deneubourgh, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-Organization in Biological Systems*. Princeton University Press.
- Chang Li, S. Y. (2009). An Adaptive Particle Swarm Optimizer for Function Optimization. (IEEE, Ed.) Leicester, United Kingdom. Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6069879>

Chang, L., & Yang, S. (2010, July 18-23). Adaptive Learning Particle Swarm Optimizer-II for Global. (IEEE, Ed.) Retrieved from

http://www.cs.le.ac.uk/people/sy11/Papers/CEC10_1.pdf

Changhe Li, S. Y. (2012, June). A self-Learning Particle Swarm Optimizer for Global Optimization Problems (SLPSO). *IEEE Transaction on Systems, Man and Cybernetics - Part B: Cybernetics, Vol 42, No. 3, June 2012*. Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6069879>

Clerc, M. (2005/2006). *Particle Swarm Optimization (L'Optimisation par essais particuliers)*. Hermes Science Lavoisier in France (2005) ISTE Ltd in Great Britain & United States (2006).

Commons Creative - Evolutionary Algorithms. (2011). *Evolutionary Algorithms*. Memphis: www.booksllc.net.

Commons Creative - Optimization Algorithms. (2011). *Optimization Algorithms*. (B. L. Series, Ed.) Memphis: www.booksllc.net.

Esperanza García-Gonzalo, J. L.-M. (2014). Convergence and stochastic stability analysis of particle swarm optimization variants with generic parameter distributions. 286-302. Retrieved from https://ac.els-cdn.com/S0096300314014416/1-s2.0-S0096300314014416-main.pdf?_tid=d687ccb2-d79d-11e7-8585-00000aab0f6c&acdnat=1512246036_55f855e4673fe5face4d83521f878e8f

Festinger. (1954). A Theory of Social Comparison Processes. Retrieved from
<https://www.humanscience.org>:
[https://www.humanscience.org/docs/Festinger%20\(1954\)%20A%20Theory%20of%20Social%20Comparison%20Processes.pdf](https://www.humanscience.org/docs/Festinger%20(1954)%20A%20Theory%20of%20Social%20Comparison%20Processes.pdf)

Haken, H. (1983). *Synergetics An Introduction Nonequilibrium Phase Transitions and Self-Organisation*. Berlin Heidelberg: Springer Verlag.

Heppner, & Grenander. (1990). <https://www.researchgate.net>. Retrieved from
https://www.researchgate.net/profile/Frank_Heppner/publication/216300775_A_Stochastic_Nonlinear_Model_for_Coordinate_Bird_Flocks/links/55acf79d08aea9946727dd2a/A-Stochastic-Nonlinear-Model-for-Coordinate-Bird-Flocks.pdf?origin=publication_detail&ev=pub_in

Hua-Ma, L., Ming, X., Meng, S., & Zhe, M.-L. (2013). Convergence and Spectral Radius Analysis PSO and Parameter Selection for the PSO algorithm based on a Stochastic Process. Retrieved from <http://scialert.net>:
<http://scialert.net/qredirect.php?doi=itj.2013.1480.1490&linkid=pdf>

James Kennedy, R. C. (2001). *Swarm Intelligence, First Edition*. Morgan Kaufmann.

Jamil, M., & Yang, X.-S. (2013). A Literature Survey of Benchmark Functions For Global. Retrieved from <https://arxiv.org/pdf/1308.4008.pdf>

Kennedy, & Eberhard. (1995). Particle Swarm Optimization. Retrieved from
http://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6%201995%20particle%20swarming.pdf

Kennedy, J., & Mendes, R. (2002). Population Structure and Particle Swarm Optimization Performance. Washington, USA. Retrieved from
<http://ieeexplore.ieee.org/document/1004493/>

LeClerc, M. (2005/2006). *Particle Swarm Optimization (L'Optimisation par essaims particulaires)*. Hermes Science Lavoisier in France (2005) ISTE Ltd in Great Britain & United States (2006).

LeClerc, M. (2012, September 23). Standard Particle Swarm Optimization. Retrieved from [r/pso/SPSO_descriptions.pdf](http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf):
http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf

Liang, & et al. (2006). Comprehensive Learning PSO. (IEEE, Ed.) Retrieved from
<https://pdfs.semanticscholar.org/2142/bb3f3c3dbce455506c1e5cd15826f5c3093a.pdf>

Novak, Latane, & Vallacher. (1994). Dynamical Social Psychology An Introduction. Retrieved from
https://www.researchgate.net/publication/236007812_Dynamical_Social_Psychology_An_Introduction

- Olsson, A. e. (2011). *Particle Swarm Optimization - Theory, Techniques and Applications*. Nova Science Publishers Inc.
- Parsopoulos, V. K., & Vrahatis, M. N. (2010). *Particle Swarm Optimization and Intelligence - Advances and Applications*. Information Science Reference.
- Poli, Riccardo. (2009, August). Mean and Variance of the Sampling Distribution of Particle Swarm Optimizers During Stagnation. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 13, NO. 4,* 712-720. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=7F27FDBA20F503AF5C80B12BF8BF98AB?doi=10.1.1.206.4273&rep=rep1&type=pdf>
- Reeves. (1983). Particle Systems - A Technique for Modelling Fuzzy Systems. Retrieved from <https://sealab.cs.utah.edu>: <https://sealab.cs.utah.edu/Courses/CS6967-F08/Papers/Reeves-1983-PSA.pdf>
- Reynolds. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. Retrieved from <https://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf>
- Sedighzadeh, D., & Masehian, E. (2009). Particle Swarm Optimization Methods, Taxonomy and Applications - International Journal of Computer Theory and Engineering, Vol 1 No 5. Retrieved from www.academia.edu: http://www.academia.edu/213172/Particle_Swarm_Optimization_Methods_Taxonomy_and_Applications

- Shannon. (1948). A Mathematical Theory of <Communication. Retrieved from <http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf>
- Teruyoshi Yamaguchi, N. I. (2007, April). Adaptive Particle Swarm Optimization using Information about Global Best. Tokyo, Japan. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/ej.20487/epdf?r3_referer=wol
- Tian, D. (2013). A Review of Convergence Analysis of Particle Swarm Optimization. 117-128. Retrieved from http://www.sersc.org/journals/IJGDC/vol6_no6/10.pdf
- Wang, X., Wang, Y., Zeng, H., & Zhou, H. (2006). Particle Swarm Optimization with Escape Velocity (EVPSO). Retrieved from <http://ieeexplore.ieee.org/document/4072129/>
- Wolpert. (1996, December 31). No Free Lunch Theorems. Retrieved from <http://www.no-free-lunch.org/WoMa96a.pdf>
- Yang. (2010). *BAT algorithm: Review and Applications*. Retrieved from <https://arxiv.org/pdf/1308.3900.pdf>
- Yang. (2010, March). Firefly for Multimodal Optimization. Cambridge, Great Britain. Retrieved from <https://arxiv.org/pdf/1003.1466.pdf>
- Yang, X. S., Cui, Z., Xiao, R., Gandomi, A. H., & Karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Applications*. London: Elsevier.

Yang, X.-S. (2013, August 18). Firefly Algorithm: Recent Advances and Applications.

Retrieved from <https://www.researchgate.net>:

https://www.researchgate.net/publication/255971821_Firefly_Algorithm_Recent_Advances_and_Applications

Yang, X.-S., Cui, Z., Renbin, X., Gandomi, A. H., & Karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Computation*. Elsevier Insights.

Yu Wang, B. L. (2010). Self-adaptive learning based particle swarm optimization.

Retrieved from Science Direct:

<http://www.sciencedirect.com./science/article/pii/S0020025510003312>

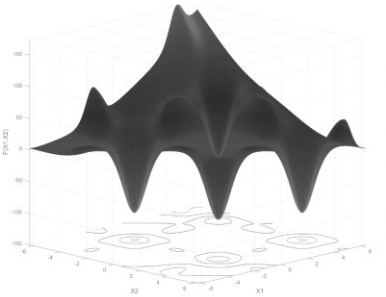
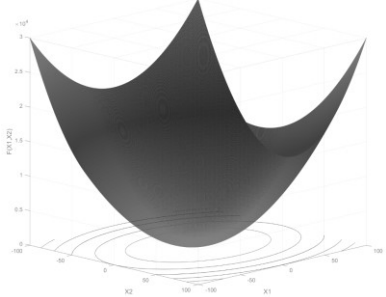
LIST OF ABBREVIATIONS

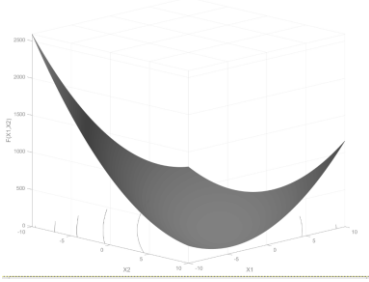
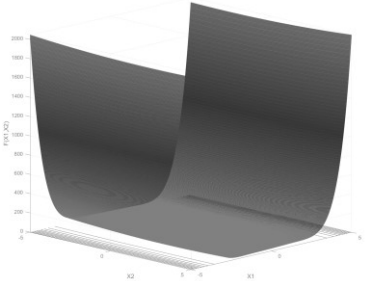
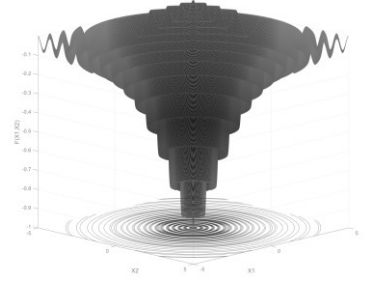
AACA	<i>Artificial Ant Colony Algorithm</i>
ABCA	<i>Artificial Bee Colony Algorithm</i>
ACM	<i>Adaptive Cultural Model</i>
AI	<i>Artificial Intelligence</i>
ALPSO I	<i>Adaptive Learning Particle Swarm Optimization</i>
ALPSO II	<i>Adaptive Learning Particle Swarm Optimization II</i>
APSO	<i>Adaptive Particle Swarm Optimization</i>
BAA	<i>Bat Artificial Algorithm</i>
CA	<i>Cultural Algorithm</i>
CLPSO	<i>Comprehensive Learning PSO</i>
COMPSO	<i>Composite PSO</i>
CSA	<i>Cuckoo Search Algorithm</i>
DbV	<i>Difference based velocity update</i>
EbV	<i>Estimation based velocity update</i>
E-MPSO	<i>Entropy based Memetic PSO</i>
EP	<i>Evolutionary Programming</i>
EVPSO	<i>Escape Velocity PSO</i>
FFA	<i>Firefly Algorithm</i>
Gbest	<i>Global best in Particle Swarm</i>

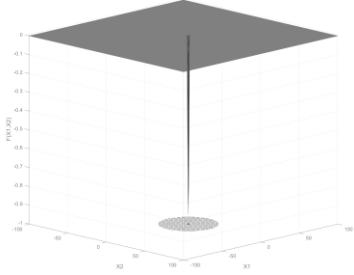
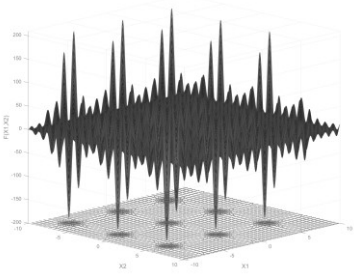
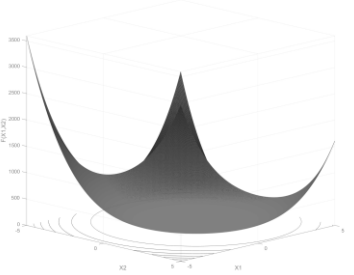
GP	<i>Genetic Programming</i>
GWA	<i>Glowworm Swarm Algorithm</i>
HMP	<i>Hyper-Middle-Point</i>
HSA	<i>Hunting Search Algorithm</i>
Lbest	<i>Local best in Particle Swarm</i>
MFFA	<i>Memetic Firefly Algorithm</i>
MSAPSO	<i>Multi Self Adaptive Particle Swarm Optimization</i>
MSF	<i>MSAPSO Stability Formula</i>
MSL	<i>MSAPSO Stability Line</i>
NC	<i>Natural Computation</i>
NFL	<i>No-Free-Lunch Theorem</i>
NPSO	<i>Niching PSO</i>
PSO	<i>Particle Swarm Optimization</i>
PSO-CL-pbest	<i>PSO Comprehensive Learning Pbest</i>
SIE	<i>Shannon Information Entropy</i>
SLPSO II	<i>Self-Learning Swarm Optimizer for Global Optimization</i>
SLSPO I	<i>Self-adaptive learning based PSO</i>
SPSO	<i>Standard PSO, Standard Particle Swarm Optimization</i>
UPSO	<i>Unified Particle Swarm Optimization</i>
XPSO	<i>Constriction PSO</i>

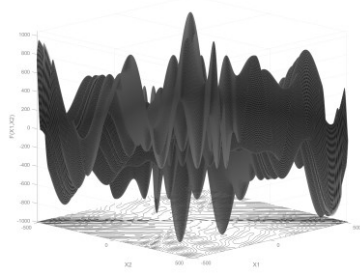
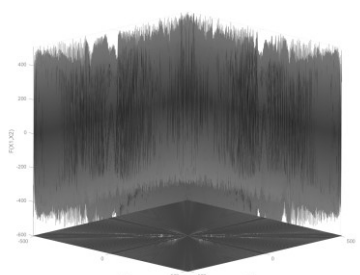
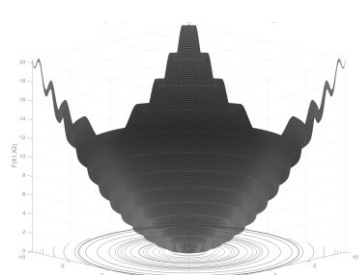
APPENDICES

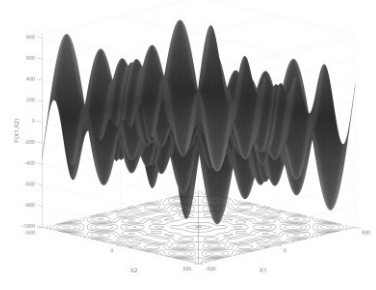
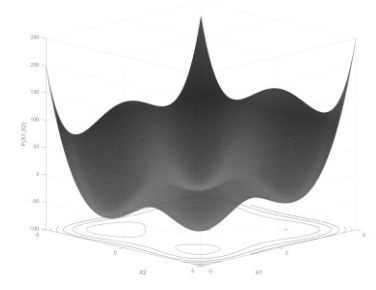
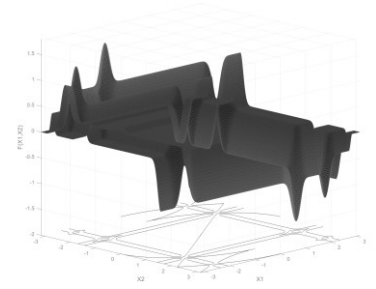
APPENDIX A – MINIMUM BENCHMARK FUNCTIONS

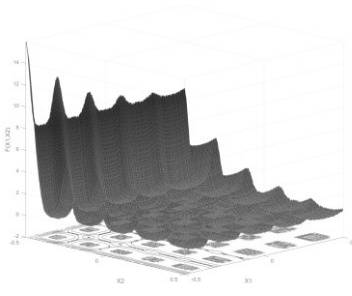
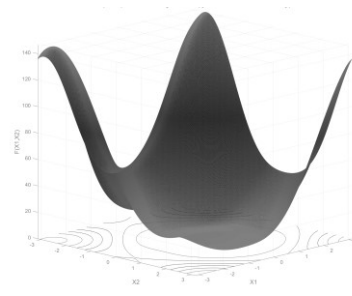
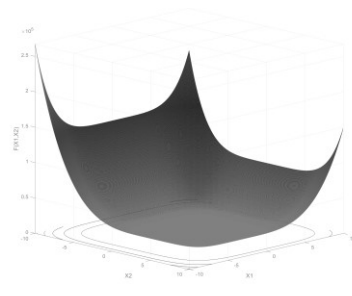
Benchmark Function	2D view	D	X _i Range R	X _{min} x*	Function Value f(x*)
Bird		[2]	$[-2\pi, 2\pi]^D$	$\begin{bmatrix} 1.58214, \\ 3.13024 \end{bmatrix}$	-106.76453
Bohachevsky		[2]	$[-100, 100]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0

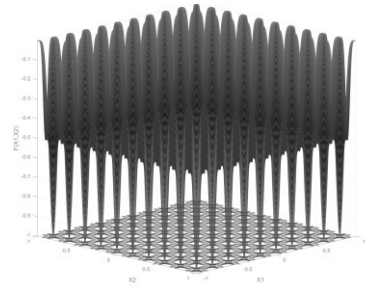
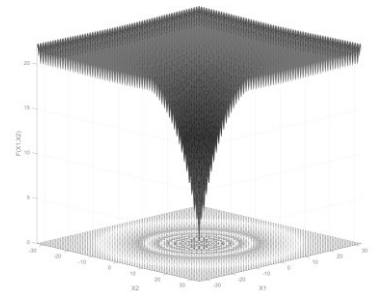
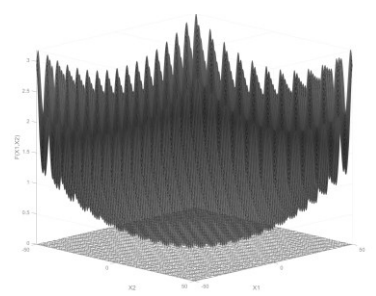
Benchmark Function	2D view	D	X _i Range R	X _{min} x*	Function Value f(x*)
Booth		[2]	$[-10,10]^D$	$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$	0
Camel		[2]	$[-5,5]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0
Dropwave		[2]	$[-5,5]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	-1

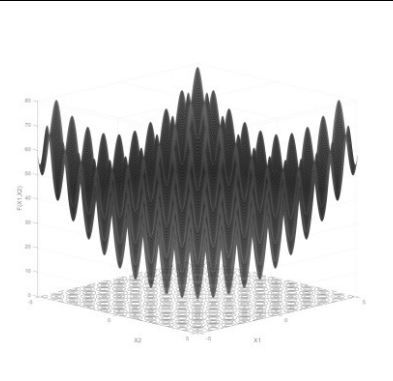
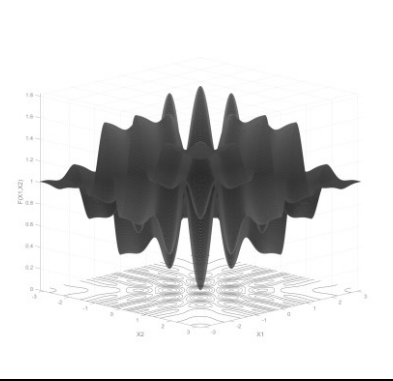
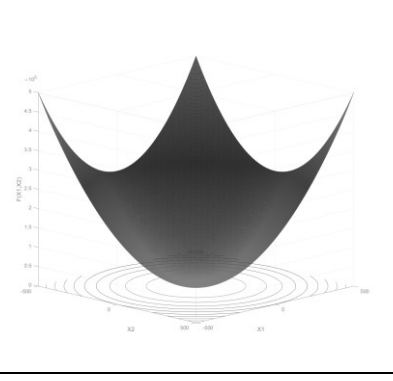
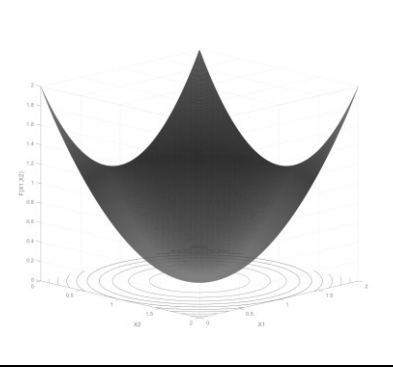
Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Easom		[2]	$[-100,100]^D$	$\begin{bmatrix} \pi \\ \pi \end{bmatrix}$	-1
Shubert		[2]	$[-10,10]^D$	[Multiple]	186,73091
Zettl		[2]	$[-5,5]^D$	[Multiple]	-0,03790

Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Eggholder		[2]	$[-512,512]^D$	$\begin{bmatrix} 512 \\ 404,23181 \end{bmatrix}$	-959,6406
Rana		[2]	$[-512,512]^D$	[Multiple]	-513
Salomon		[2]	$[-10,10]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0

Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Schwefel		[2]	$[-500, 500]^D$	$[420, 96875]$ $[420, 96875]$	-837,9657
Styblinskiang		[2]	$[-5, 5]^D$	$[-2, 90353]$ $[-2, 90353]$	-78,33233
Michalewicz		[2]	$[-\pi, \pi]^D$	$[2, 20290]$ $[1, 57079]$	-1,80130

Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Ripple1		[2]	$[-0,5,0,5]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	-0,2
Trigonometric		[2]	$[-\pi, \pi]^D$	[Multiple]	0
Quintic		[2]	$[-10,10]^D$	[Multiple]	0

Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Deb1		[2]	$[-1,1]^D$	[Multiple]	-1
Ackley1		[2]	$[-32,768,32,768]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0
Griewank		[2]	$[-50,50]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0

Benchmark Function	2D view	D	X_i Range R	X_{min} x^*	Function Value $f(x^*)$
Rastrigin		[2]	$[-5,12,5,12]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0
Wavy		[2]	$[-\pi, \pi]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0
Sphere		[2]	$[0,0]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0
Sphere_at_one		[2]	$[1,1]^D$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	0

APPENDIX B – MSAPSO STRUCTURAL DIAGRAM

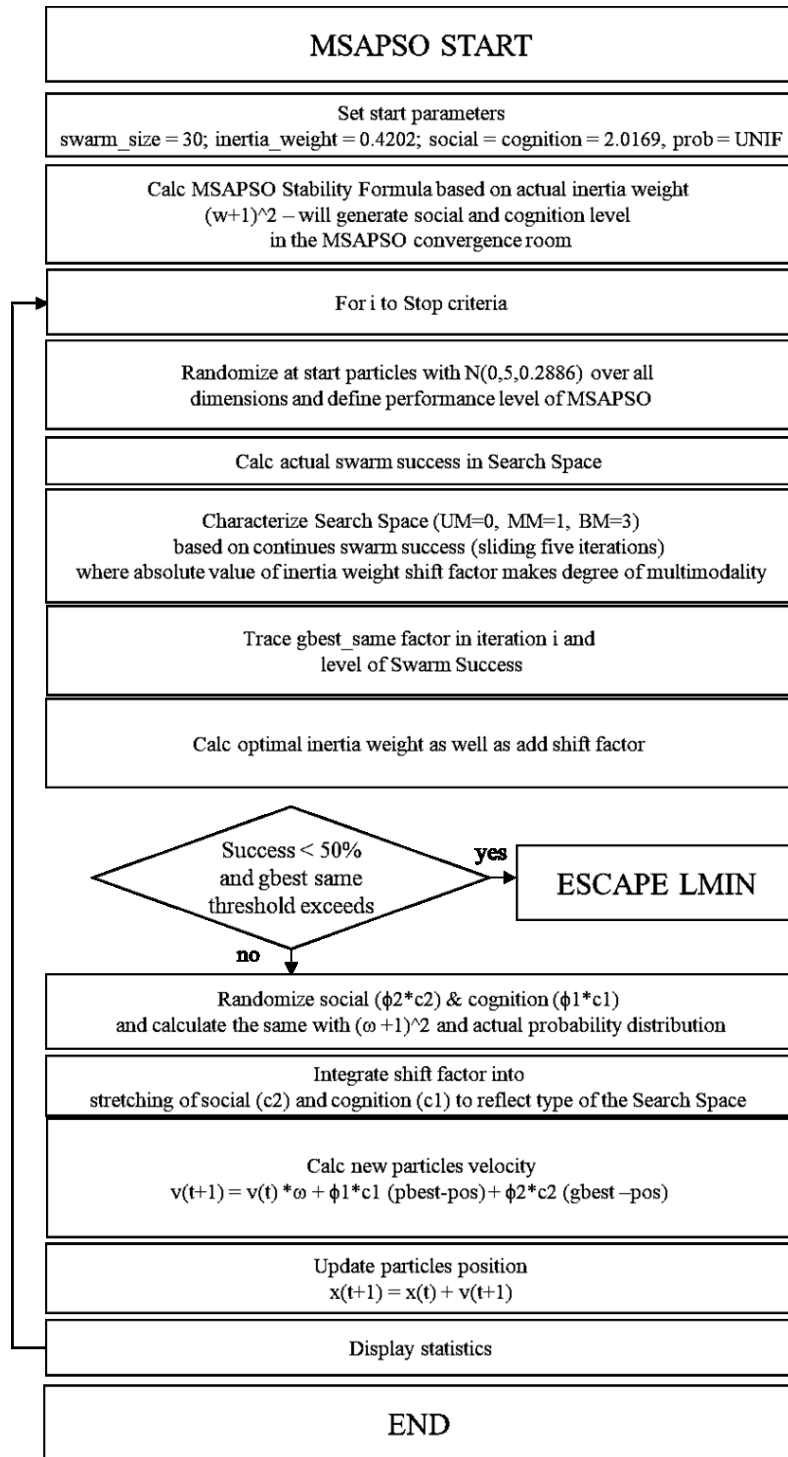


Figure 65: MSAPSO Structural Diagram

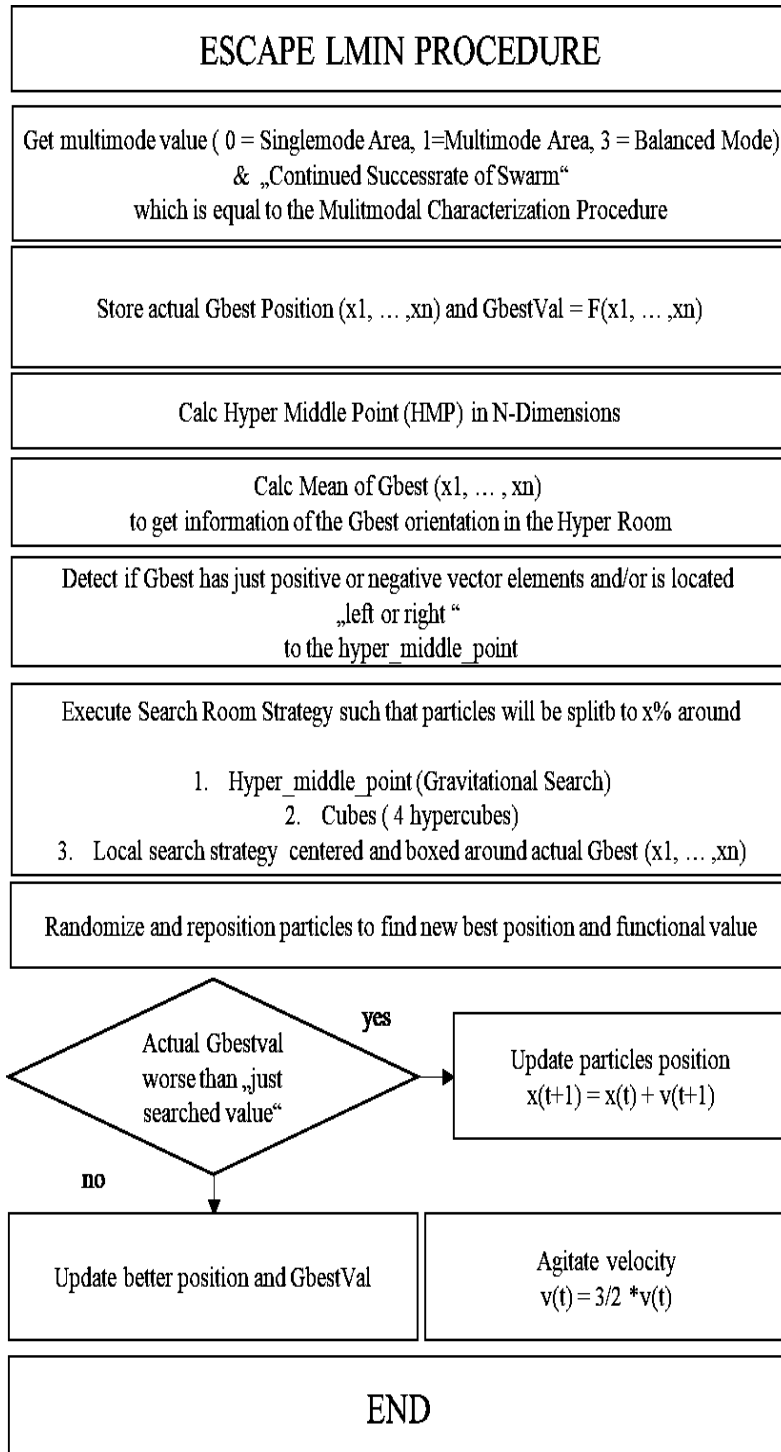


Figure 66: MSAPSO Escape Local Optima Mode

*****CURRICULUM VITAE

Personal Information

LAST NAME: Koch
FIRST NAME: Gerhard
NATIONALITY: German
GENDER: Male
LANGUAGES: German, English, French
PRIVATE PHONE: +49 7471 617238
MOBILE PHONE: +49 171 8614865 (private)
EMAIL-ADDRESS: gkoch3103@gmail.com

Education

UNIVERSITY OF APPLIED SCIENCES
ALBSTADT-EBINGEN: Technical Computer Science
1988-1993
UNIVERSITY OF APPLIED SCIENCES
HAMBURG: Industrial Engineering
2004-2006
UNIVERSITY OF LOUISVILLE
SPEEDSCHOOL OF ENGINEERING: PhD Industrial Engineering
2009-2017

Professional Societies

IEEE: Member since 2007
AUTOISAC: Board Member since 2017