Meta-Lamarckian Learning in Three Stage Optimal Memetic Exploration

Ferrante Neri*†, Matthieu Weber*, Fabio Caraffini*†, and Ilpo Poikolainen*
*Centre for Computational Intelligence, School of Computer Science and Informatics,
De Montfort University, The Gateway, Leicester LE1 9BH, England, United Kingdom
Email: {fneri, fcaraffini}@dmu.ac.uk
†Department of Mathematical Information Technology, University of Jyväskylä
P.O. Box 35 (Agora), 40014 Jyväskylä, Finland
Email: {ferrante.neri, matthieu.weber, fabio.caraffini, ilpo.poikolainen}@jyu.fi

***Abstract*—Three Stage Optimal Memetic Exploration (3SOME) is a single-solution optimization algorithm where the coordinated action of three distinct operators progressively perturb the solution in order to progress towards the problem's optimum. In the fashion of Memetic Computing, 3SOME is designed as an organized structure where the three operators interact by means of a success/failure logic. This simple sequential structure is an initial example of Memetic Computing approach generated by means of a bottom-up logic. This paper compares the 3SOME structure with a popular adaptive technique for Memetic Algorithms, namely Meta-Lamarckian learning. The resulting algorithm, Meta-Lamarckian Three Stage Optimal Memetic Exploration (ML3SOME) is thus composed of the same three 3SOME operators but makes use a different coordination logic. Numerical results show that the adaptive technique is overall efficient also in this Memetic Computing context. However, while ML3SOME appears to be clearly better than 3SOME for low dimensionality values, its performance appears to suffer from the curse of dimensionality more than that of the original 3SOME structure.**

***Index Terms*—Memetic Computing, Ockham Razor, Computational Intelligence Optimization, Automatic Algorithmic Design, Meta-Lamarckian Learning**

I. Introduction

In the past few years, the notion of Memetic Algorithm (MA) for solving optimization problems, introduced in [1], has evolved into a more general framework named Memetic Computing (MC), see e.g., [2], [3], and [4]. According to its original definition, a MA is defined as the fusion of one or more local search algorithms within an evolutionary framework, the former being activated within the generation cycle of the latter. When more than one local search algorithm are employed, the designer of the algorithm faces the problem of deciding the manner in which these algorithmic modules (referred to as memes) can be coordinated in order to improve the global performance of the algorithm; these research problems are at the core of the study of MAs. The success and diffusion of MAs is to be searched within their flexibility. The No Free Lunch Theorem [5] proves that there is no universally suitable optimization algorithm and that each optimization problem is a separate story which must be addressed by a specific algorithmic instrument. Since MAs (and MC approaches) are naturally designed each time by selecting their components, they appeared a valid alternative to tackle specific applications, see e.g. [6]. If the concept of algorithmic design is looked from a complementary perspective, most, if not all, optimization algorithms can be considered as a collection of relatively simple modules, the memes, that are in some way coordinated in order to solve optimization problems. In this sense, MC is an umbrella name to identify all the optimization algorithms. Nonetheless, the MC definition is crucially important as it allows to think about optimization algorithms no longer as paradigms but as structured collections of operators. For a given problem, the proper selection of the operators and their coordination rule are at the basis of the success of an algorithm.

The topic of algorithmic coordination in MAs has been extensively disccused over the last years. In [7] a classification is given. In [4] the classification of coordination methods has been extended and updated. The following four categories have been identified: 1) Adaptive Hyper-heuristic, where heuristic rules are employed (e.g., [8], [9], [10], [11]); 2) Meta-Lamarckian learning defined in [12], where the activation of the memes depends on their success, see also [13], [14], [15]; 3) Self-Adaptive and Co-Evolutionary, where the rules coordinating the memes are evolving in parallel with the candidate solutions of the optimization problem or encoded within the solution, see [16], [17], [18], [19]; and 4) Fitness Diversity-Adaptive, where the activation of the memes depends on a measure of the diversity (e.g., [20], [21], [6], [22], [23]). As a general idea, the algorithmic designer attempts to have a system which performs the coordination automatically. The algorithm is supposed to decide itself during runtime the manner in which the different memes are applied, adapting itself to the problem at hand and thus leading to a preliminary form of automatic design of optimization algorithms.

In this paper, we study the effect of employing a Meta-Lamarckian learning approach to coordinate the three operators composing the Three Stage Optimal Memetic Exploration (3SOME) algorithm originally presented in [24]. The 3SOME algorithm, as a choice of the authors, employs a minimalistic coordination scheme simply based on the success of each operator. The 3SOME coordination scheme constitutes the structure of the algorithm. In the present work we attempt

to study the dependency of the algorithmic performance on the coordination of the operators. More specifically, the same 3SOME operators are here tested without the 3SOME structure but by means of the Meta-Lamarckian learning coordination, thus generating the Meta-Lamarckian 3SOME (ML3SOME). The selection of this simple coordination scheme instead of modern relatively complex adaptive systems for parameter setting and component coordination, see [25], [26], [27], has been carried out as a consequence of the Oackham's Razor principle in MC formulated in [24]. It is fundamental to avoid unnecessary complexity while the algorithmic design is performed. MC structures should be constructed, in a bottom-up logic, by progressively adding complexity until the optimization aim is achieved.

The remainder of this paper is organized in the following way. Section II describes the three operators composing 3SOME, while Section III describes in details the two coordination schemes. Section IV displays the experimental test bed and numerical results produced by the two algorithms studied in this paper. Finally, Section V gives the conclusion of this work.

II. Operators of the Three Stage Optimal Memetic Exploration

In order to clarify the notation in this paper, we refer to the minimization problem of an objective function $f(x)$, where the candidate solution x is a vector of n design variables (or genes) in a decision space D.

At the beginning of the optimization problem one candidate solution is randomly sampled within the decision space D. In analogy with compact optimization, see e;g; [28] and [29], we will refer to this candidate solution as elite and indicate it with the symbol x_e. In addition to x_e, the algorithm makes use of another memory slot for attempting to detect other solutions. The latter solution, namely trial, is indicated with x_t.

The following subsections describe the working principle of each operator composing the 3SOME algorithm and the other two variants proposed in this paper. These three operators (memes) are named long-distance, middle-distance, and short-distance exploration, respectively. Further details about the implementation of each operator are available in [24].

A. Long-distance exploration

The purpose of the long-distance operator is to explore the entire decision space and detect a new promising solution. While the elite x_e is retained, at first, a trial solution x_t is generated by randomly sampling a new set of n genes. Subsequently, the DE exponential crossover is applied between x_e and x_t, see [29]. If the trial solution outperforms the elite, a replacement occurs. A replacement has been set also if the newly generated solution has the same performance as the elite, to prevent the search getting trapped in some plateaus of the decision space. This exploration stage performs a global stochastic search and thus attempts to detect unexplored promising areas of the decision space. While this search mechanism extensively explores the decision space, the employed crossover method also promotes retention of a small section of the elite within the trial solution. This kind of inheritance of some genes appears to be extremely beneficial in terms of performance with respect to a stochastic blind search (which would generate a completely new solution at each step). The pseudo-code of this component is shown in Algorithm 1. The long-distance exploration is repeated until it detects a solution that outperforms the original elite.

Algorithm 1 Long-distance exploration

```
generate a random solution x_t within D
generate i = round(n · rand(0,1))
count = 1
x_t[i] = x_e[i]
while rand(0,1) ≤ Cr AND count < n do
    x_t[i] = x_e[i]
    i = i + 1
    if i == n then
        i = 1
    end if
    count = count + 1
end while
if f(x_t) ≤ f(x_e) then
    x_e = x_t
end if
```

B. Middle-distance exploration

The middle-distance exploration operator attempts to focus the search started by the long-distance exploration in order to exploit the detected search directions. At first a a hypercube of side δ, centred around the solution x_e, is constructed. The middle-distance exploration performs the search within the hyper-volume contained in this hyper-cube of side δ. Subsequently, for $4n$ times (n is the dimensionality), a trial point x_t is generated within the hypercube. The trial point x_t is generated from the elite x_e by performing random sampling within the hyper-cube at first and then an exponential crossover between x_e and the randomly generated point. The fitness of this newly generated point is then compared with the fitness of the elite. If the new point outperforms the elite (or has the same performance), x_e is replaced by the new point, otherwise no replacement occurs. The pseudo-code displaying the working principles of this operator is given in Algorithm 2.

Algorithm 2 Middle-distance exploration

```
construct a hyper-cube with side width δ around x_e
for j = 1 : 4n do
    generate a random solution x_t within the hyper-cube
    generate i = round(n · rand(0,1))
    count = 1
    x_t[i] = x_e[i]
    while rand(0,1) ≤ Cr AND count < n do
        x_t[i] = x_e[i]
        i = i + 1
        if i == n then
            i = 1
        end if
        count = count + 1
    end while
    if f(x_t) ≤ f(x_e) then
        x_e = x_t
    end if
end for
```

C. Short-distance exploration

The short-distance exploration is a deterministic search that perturbs the variables of the elite one by one, behaving as a simple steepest descent deterministic local search algorithm. The perturbation is not symmetrical but is heuristically arranged in order to save budget with respect to an exhaustive exploratory step, see [30]. This exploration move attempts to fully exploit promising search directions by performing the descent of promising basins of attraction and possibly finalize the search if the basin of attraction is globally optimal. The short-distance exploration stage requires an additional memory slot, which will be referred to as x_s (s stands for short). Starting from the elite x_e, this local search, explores each coordinate i (each gene) and samples $x_s[i] = x_e[i] - \rho$, where ρ is the exploratory radius. Subsequently, if x_s outperforms x_e, the trial solution x_t is updated (it takes the value of x_s), otherwise a half step in the opposite direction $x_s[i] = x_e[i] + \frac{\rho}{2}$ is performed. Again, x_s replaces x_t if it outperforms x_e. If there is no update i.e., the exploration was unsuccessful, the radius ρ is halved. This exploration is repeated for all the design variables and stopped when a prefixed budget (equal to 150 iterations) is exceeded. The pseudo-code displaying the working principles of the short-distance exploration is given in Algorithm 3.

I should be noted that short distance exploration employs an asymmetric search step as it explores solutions, along each axis, at a ρ distance in one direction verse and $\frac{\rho}{2}$ in the opposite verse. Although a rigorous theoretical explanation of this algorithmic choice is not yet available, experimentally this logic appeared to be much more efficient than a straightforward symmetric exploration, see [30].

As a remark, a toroidal management of the bounds has been implemented for the three operators above. This means that if, along the dimension i, the design variable $x[i]$ exceeds the bounds of a value ζ, it is reinserted from the other end of the interval at a distance ζ from the edge, i.e. given an interval $[a, b]$, if $x[i] = b + \zeta$ it takes the value of $a + \zeta$.

Algorithm 3 Short-distance exploration

```
while local budget condition do
    x_t = x_e
    x_s = x_e
    for i = 1 : n do
        x_s[i] = x_e[i] − ρ
        if f(x_s) ≤ f(x_t) then
            x_t = x_s
        else
            x_s[i] = x_e[i] + ρ/2
            if f(x_s) ≤ f(x_t) then
                x_t = x_s
            end if
        end if
    end for
    if f(x_t) ≤ f(x_e) then
        x_e = x_t
    else
        ρ = ρ/2
    end if
end while
```

III. Coordination of the operators

Let us indicate with L, M, and S, the long-distance, middle-distance, and short-distance exploration respectively. The following subsections describe, at first, the original coordination scheme employed in [24] and then a coordination according to the Meta-Lamarckian learning proposed for the first time in this paper.

A. Original 3SOME memetic structure

In the original 3SOME algorithm, the three operators are coordinated according to a heuristically determined scheme, which is repeated until the termination criterion is met, that is the exhaustion of a budget of fitness evaluations.

The L operator is first applied until it produces a solution that outperforms the elite. This operators has thus the role of exploring the decision space to generate a new promising solution to be further exploit. The M operator is then run repeatedly, until it does not improve anymore upon the elite. This means that this second operator attempts to search within the interesting area of the decision space. If this search leads to an improvement, the research is continued. It must be appreciated that L and M are stopped by diametrically opposite criteria. This is set because while L aims to detect one new basin of attraction or a new promising search direction, M aims to subsequently improve upon the genotype detect by L and exploit the area of interest as much as possible. This explains why L is interrupted when the search succeeded (possibly after numerous failures) and M is interrupted when the exploitation turns out to be unsuccessful.

Finally, S further refines the work performed by M by performing a steepest descent deterministic search to fully exploit the basin of attraction. As a further consideration, S performs a narrow search and is a pretty computationally expensive. Thus, it is used only when the basin of attraction seems promising indeed and when M is no longer capable to perform improvements. If S detects new promising solutions, the exploitation of the area is continued by activating M again (and then S again). If S fails at detecting a new elite solution, the area is likely fully exploited and there is no use in continuing the local search within it's neighbourhood. For this reason, if S fails, L is activated anew to attempt the exploration in other areas of the decision space.

The description of the working principles of the 3 SOME structure is given Algorithm 4.

B. Meta-Lamarckian learning

Meta-Lamarckian learning is a sophisticated and efficient adaptive scheme proposed in [12] in the context of MAs. This adaptive scheme organizes the operators composing the algorithm (originally the local search components) within a pool. A selection probability is associated to each operator. The selection probability of each operator depends on its performance history during the previous activations. More specifically, the performance $\eta_p(t)$ of the operator p at iteration

Algorithm 4 3SOME structure (coordination of the operators)

```
generate the solution x_e
while global budget condition do
    while x_e is not updated do
        apply to x_e the long-distance exploration L
    end while
    while x_e is updated do
        apply to x_e the middle-distance exploration M
    end while
    apply to x_e the short-distance exploration S
    if x_e has been updated then
        apply middle-distance exploration M
    else
        apply long-distance exploration L
    end if
end while
```

t is computed as

$$\eta_p(t) = \frac{fe_p^*(t)}{fe_p(t)} \tag{1}$$

where $fe_p(t)$ is the number of fitness evaluations spent by the operator p at iteration t since the algorithm was started, and $fe_p^*(t)$ is the number of fitness evaluations, at iteration t, used by operator p, that have led to an improvement of the elite. In our case, the probability $P_p(t)$ for operator p to be selected at iteration t is thus defined as

$$P_p(t) = \frac{\eta_p(t)}{\eta_L(t) + \eta_M(t) + \eta_S(t)}. \tag{2}$$

The actual choice of the next operator is performed by the mean of a roulette-wheel selection, as described in [12].

However, since the probability for an operator to be selected depends on its past success, each operator must be given a chance to accumulate some amount of success in order for its selection probability to be above zero. The operators therefore undergo at first a training period, during which their probability of being selected does not follow Equation 2, but instead is equal among all three operators. Every time t that an operator has exhausted its allocated budget (or returns, in the case of the long-distance operator), the number of fitness evaluations $fe_p(t)$ used by each of the three operators is checked. The training period thus ends when $\forall p \in \{L, M, S\}\ fe_p(t) > 0$. For the sake of clarity, this coordination scheme is represented as pseudo-code in Algorithm 5.

Algorithm 5 Meta-Lamarckian coordination

```
t ← 0
while termination condition is not met do
    generate U ← rand(0, 1)
    if fe_L(t) > 0 and fe_M(t) > 0 and fe_S(t) > 0 then
        if U < P_L(t) then
            apply the long-distance operator
        else if U < P_L(t) + P_M(t) then
            apply the middle-distance operator
        else
            apply the short-distance operator
        end if
    else
        if U < 1/3 then
            apply the long-distance operator
        else if U < 2/3 then
            apply the middle-distance operator
        else
            apply the short-distance operator
        end if
    end if
    update P_L(t), P_M(t) and P_S(t)
    t ← t + 1
end while
```

IV. Numerical results

The performance of the original 3SOME structure has been compared with the ML3SOME.

The algorithms in this study have been tested on the test bed defined in [31] (24 problems) in 10, 40, and 100 dimensions and on the testbed defined in [32] (20 problems) in 1000 dimensions. In order to perform a fair comparison, both the algorithms have been run with the same parameters, $\alpha_e = 0.05$, δ and ρ equal to respectively 10 % and 40 % of the total decision space width and the budget for middle length exploration has been fixed equal to $4n$ fitness evaluations at each activation. For an extensive discussion on the parameter setting of the 3SOME framework see [24]. Each algorithm has been allocated a budget of $5000 \times n$ fitness evaluations for each run and for each problem, 100 runs have been performed.

Tables I, II, III, and IV display the numerical results (in terms of final value and standard deviation) for the test problems considered in this work. The best results are highlighted in bold face. In order to strengthen the statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in [33], where the confidence level has been fixed at 0.95: a "+" symbol indicates the case when ML3SOME outperforms the algorithm it is compared against, "−" indicates that ML3SOME is on the contrary outperformed, and "=" indicates that the two algorithms have indistinguishable performance.

The displayed results extend the finding in [12]. While in [12] the meta-Lamarckian learning was proven to be effective for coordinating multiple local search components within a standard MA framework, the results here presented show that the effectiveness of meta-Lamarckian schemes can be etended to algorithms which do not have a population nor an evolutionary structure. It can be observed that in 10 dimensions ML3SOME clearly outperforms 3SOME in 11 cases while it is outperformed for only 3 problems. Thus, for the testbed proposed in [31] and in 10 dimensions, the coordination of the operators by means of a meta-Lamarckian scheme appears preferable. It must be observed that the testbed in [31] is composed of 24 diverse problems which display various features in terms of multimodality, separability, ill-conditioning etc. In this sense, we can conclude that for low dimensionality values the meta-Lamarckian coordination is a robust and valid option for the meme coordination. A similar consideration can be done for the problems in 40 dimensions.

Numerical results in 100 and 1000 dimensions are much more contrasted. The comparison of the meta-Lamarckian learning with the original 3SOME structure show that, for high-dimensional values the performance of the two scheme, albeit different, is equally good. More specifically, the Wilcoxon test indicates that ML3SOME outperforms 3SOME in roughly half of the test cases, while the opposite is true in the other cases, with a small number of undecided cases.

TABLE I
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST FOR 10-DIMENSION PROBLEMS [31] (THE REFERENCE ALGORITHM IS ML3SOME)

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

TABLE II
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST FOR 40-DIMENSION PROBLEMS [31] (THE REFERENCE ALGORITHM IS ML3SOME)

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

Despite the fact that ML3SOME and 3SOME appear to consistently outperform each other on a subset of the test problems across the number of dimensions, the interpretation of the results is not trivial. In 100 dimensions, the original 3SOME structure appears to offer a slightly better performance than the meta-Lamarckian scheme for separable, weakly ill-conditioned, and uni-modal problems. This tendency has anyway some exceptions such as linear slope and step ellipsoidal functions (f_5 and f_7) respectively. For these two problems ML3SOME achieves a better result with an important margin. It is relevant to observe that the meta-Lamarckian learning appears to be regularly more efficient than the 3SOME structure for all the multi-modal functions with adequate global structure ($f_{15} - f_{19}$). Regarding the multi-modal functions with weak global structure, ML3SOME and 3SOME appear to be equally good. In 1000 variables, ML3SOME outperforms 3SOME in half of the problems and is outperformed in most of the other cases. It can be observed that when 3SOME displays a better performance than ML3SOME, the difference in terms of final fitness value is usually small with repect to the total decay (see Fig.s 1 and 5) while in those problems where ML3SOME outperforms 3SOME the margin of difference in the fitness values is remarkably wide (see Fig.s 2, 3, and 4). Although the relevance of the outperformance margin width

TABLE III
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST FOR 100-DIMENSION PROBLEMS [31] (THE REFERENCE ALGORITHM IS ML3SOME)

	ML3SOME	3SOME	
f_1	$\mathbf{7.95e+01 \pm 3.75e-14}$	$\mathbf{7.95e+01 \pm 3.29e-14}$	=
f_2	$\mathbf{-2.10e+02 \pm 5.43e-14}$	$\mathbf{-2.10e+02 \pm 5.69e-14}$	=
f_3	$-4.22e+02 \pm 2.49e+01$	$\mathbf{-4.39e+02 \pm 7.28e+00}$	-
f_4	$-4.04e+02 \pm 3.73e+01$	$\mathbf{-4.27e+02 \pm 8.70e+00}$	-
f_5	$\mathbf{-9.21e+00 \pm 4.84e-12}$	$7.40e+00 \pm 1.65e+02$	+
f_6	$3.59e+01 \pm 9.81e-08$	$\mathbf{3.59e+01 \pm 8.86e-08}$	=
f_7	$\mathbf{3.67e+02 \pm 8.58e+01}$	$5.97e+02 \pm 2.83e+02$	+
f_8	$\mathbf{1.78e+02 \pm 4.10e+01}$	$1.83e+02 \pm 3.31e+01$	+
f_9	$1.94e+02 \pm 3.86e+01$	$\mathbf{1.76e+02 \pm 1.36e+01}$	-
f_{10}	$3.27e+03 \pm 7.21e+02$	$\mathbf{2.68e+03 \pm 6.96e+02}$	-
f_{11}	$7.97e+02 \pm 1.34e+02$	$\mathbf{3.83e+02 \pm 8.22e+01}$	-
f_{12}	$\mathbf{-6.17e+02 \pm 6.16e+00}$	$-6.09e+02 \pm 1.83e+01$	+
f_{13}	$3.69e+01 \pm 5.04e+00$	$\mathbf{3.35e+01 \pm 4.87e+00}$	-
f_{14}	$-5.23e+01 \pm 5.17e-05$	$\mathbf{-5.23e+01 \pm 5.47e-05}$	-
f_{15}	$\mathbf{2.44e+03 \pm 5.95e+02}$	$4.53e+03 \pm 5.89e+02$	+
f_{16}	$\mathbf{8.97e+01 \pm 4.38e+00}$	$9.51e+01 \pm 6.11e+00$	+
f_{17}	$\mathbf{-6.35e+00 \pm 3.68e+00}$	$-2.63e-02 \pm 3.97e+00$	+
f_{18}	$\mathbf{2.24e+01 \pm 1.32e+01}$	$4.55e+01 \pm 1.54e+01$	+
f_{19}	$\mathbf{-9.31e+01 \pm 2.56e+00}$	$-9.08e+01 \pm 3.39e+00$	+
f_{20}	$-5.45e+02 \pm 1.17e-01$	$\mathbf{-5.46e+02 \pm 9.61e-02}$	-
f_{21}	$\mathbf{5.18e+01 \pm 1.17e+01}$	$5.19e+01 \pm 1.21e+01$	=
f_{22}	$\mathbf{-9.84e+02 \pm 1.36e+01}$	$-9.82e+02 \pm 1.47e+01$	=
f_{23}	$8.25e+00 \pm 4.62e-01$	$\mathbf{8.21e+00 \pm 4.93e-01}$	=
f_{24}	$\mathbf{1.88e+03 \pm 4.62e+02}$	$2.79e+03 \pm 4.75e+02$	+

TABLE IV
AVERAGE FITNESS ± STANDARD DEVIATION AND WILCOXON RANK-SUM TEST FOR 1000-DIMENSION PROBLEMS [32] (THE REFERENCE ALGORITHM IS ML3SOME)

	ML3SOME	3SOME	
f_1	$1.90e-10 \pm 1.40e-10$	$\mathbf{1.33e-11 \pm 3.43e-11}$	-
f_2	$\mathbf{9.92e-06 \pm 1.03e-05}$	$1.07e-04 \pm 1.77e-04$	+
f_3	$\mathbf{6.21e-05 \pm 1.16e-05}$	$5.42e-04 \pm 2.85e-04$	+
f_4	$1.89e+13 \pm 5.33e+12$	$\mathbf{7.14e+12 \pm 2.35e+12}$	-
f_5	$\mathbf{5.07e+08 \pm 1.50e+08}$	$7.06e+08 \pm 1.23e+08$	+
f_6	$\mathbf{1.93e+07 \pm 2.60e+06}$	$1.98e+07 \pm 1.01e+05$	=
f_7	$3.42e+09 \pm 9.45e+08$	$\mathbf{1.00e+09 \pm 2.56e+08}$	-
f_8	$8.73e+08 \pm 2.05e+09$	$\mathbf{3.29e+08 \pm 1.42e+09}$	-
f_9	$2.56e+08 \pm 6.34e+07$	$\mathbf{2.12e+08 \pm 4.13e+07}$	-
f_{10}	$\mathbf{3.47e+03 \pm 2.88e+02}$	$6.80e+03 \pm 3.43e+02$	+
f_{11}	$\mathbf{1.50e+02 \pm 5.12e+01}$	$1.98e+02 \pm 1.94e-01$	+
f_{12}	$\mathbf{5.37e+04 \pm 1.14e+04}$	$5.54e+04 \pm 1.18e+04$	=
f_{13}	$6.63e+03 \pm 4.53e+03$	$\mathbf{4.68e+03 \pm 4.77e+03}$	-
f_{14}	$6.38e+07 \pm 3.16e+06$	$\mathbf{5.62e+07 \pm 5.44e+06}$	-
f_{15}	$\mathbf{7.33e+03 \pm 5.57e+02}$	$1.38e+04 \pm 4.63e+02$	+
f_{16}	$\mathbf{8.67e+01 \pm 5.23e+01}$	$3.81e+02 \pm 6.26e+01$	+
f_{17}	$\mathbf{3.69e+04 \pm 7.04e+03}$	$4.78e+04 \pm 1.78e+04$	+
f_{18}	$\mathbf{1.40e+03 \pm 2.59e+03}$	$1.80e+04 \pm 1.24e+04$	+
f_{19}	$4.39e+05 \pm 6.23e+04$	$\mathbf{8.71e+04 \pm 9.95e+03}$	-
f_{20}	$\mathbf{9.94e+02 \pm 1.86e+02}$	$1.04e+03 \pm 1.63e+02$	+

strictly depends on the features of the fitness landscape, it can be conjectured that this result is due to the meta-Lamarckian logic which tends to select the components that mostly produce fitness enhancements.

Fig.s 1, 2, 3, 4, and 5 show some examples of performance trends.

V. CONCLUSION

This paper compares the performance of the original heuristic scheme for coordinating the operators in the 3SOME algorithm against an algorithm composed of the same operators but where the algorithmic structure is replaced by an adaptive scheme, namely meta-Lamarckian learning.

An extensive set of problems have been setup for this comparison. This set includes very diverse problems in terms of problem dimensionality, multimodality, separability, and ill-conditioning. Numerical results show that the meta-Lamarckian coordination appears to be more efficient than the original heuristic structure for low dimensional problems. On the other hand, the advantages of the adaptive coordination are not too evident in high dimensions. In the latter cases, the two coordination schemes display a different but still almost equally good performance. Nonetheless, it can be observed that the meta-Lamarckian learning is, in some cases, much more efficient than than the heuristic structure. Despite the fact that the two algorithms use the same set of operators, the meta-Lamarckian coordination allows a regular achievement of much better results on multi-modal problems with adequate global structure. Also in other isolated cases, the meta-Lamarckian learning allows the detection of final fitness values a few order of magnitude smaller than those detected

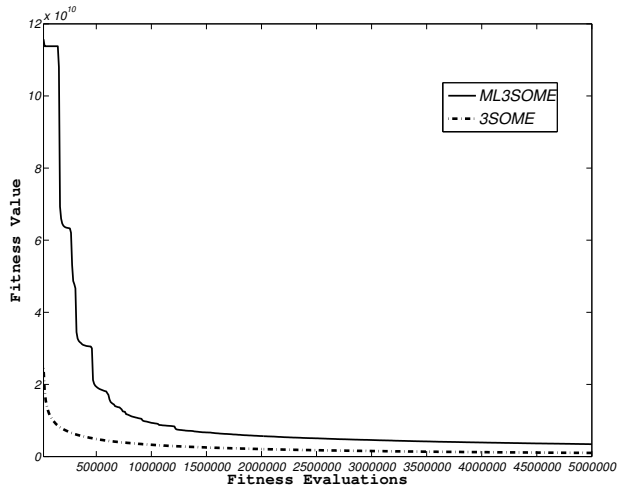

Fig. 1. Performance trends for f_7 from [32] in 1000 dimensions

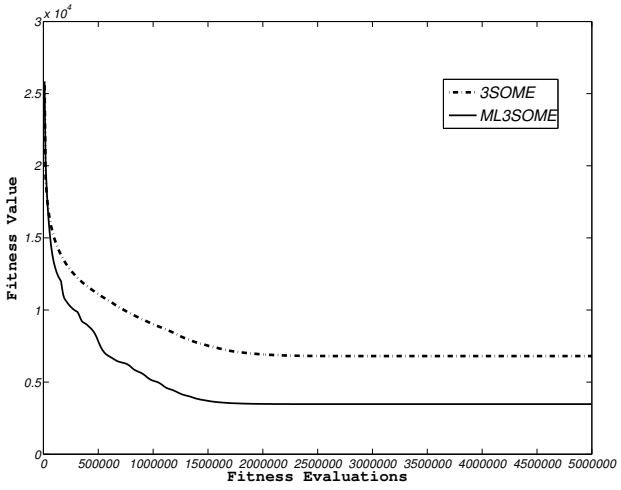

Fig. 2. Performance trends for f_{10} from [32] in 1000 dimensions

by the heuristic scheme. On the other hand, the original algorithm for a limited amount of problems, appears to be capable of detecting slightly better results compared to those detected by ML3SOME. In addition, the 3SOME structure appears, in some cases, very efficient in the early stages of the evolution and capable of quickly finding solutions with a high performance.

This study, although preliminary, has the important role of highlighting the fact that different coordination schemes of the same operators can lead to different results. Future studies focused on the bottom-up algorithmic design will attempt to combine and integrate adaptive coordination schemes within the structure of the algorithms.

ACKNOWLEDGMENTS

This research is supported by the Academy of Finland, Akatemiatutkija 130600.

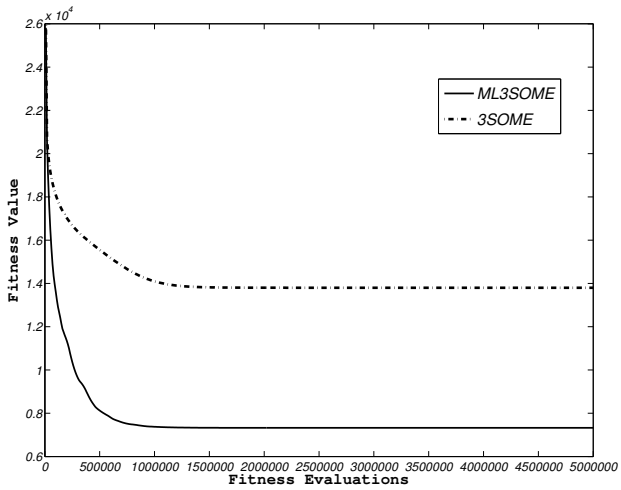

Fig. 3. Performance trends for f_{15} from [32] in 1000 dimensions

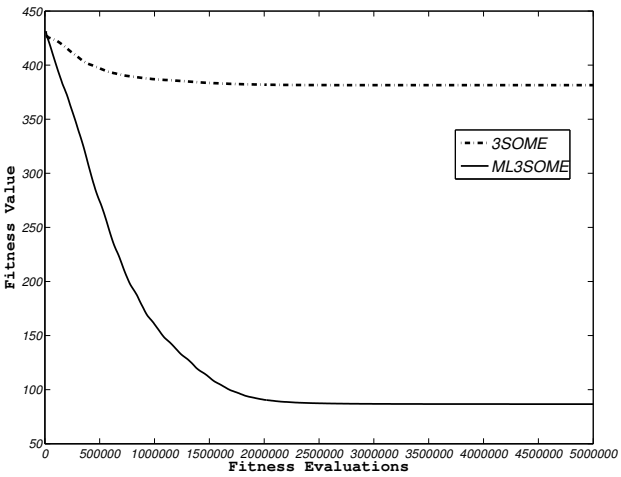

Fig. 4. Performance trends for f_{16} from [32] in 1000 dimensions

REFERENCES

[1] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Tech. Rep. 826, 1989.

[2] Y.-S. Ong, M.-H. Lim, and X. Chen, "Memetic computation-past, present and future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[3] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 379.

[4] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[5] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[6] F. Neri, J. Toivanen, G. L. Cascella, and Y. S. Ong, "An adaptive multimeme algorithm for designing HIV multidrug therapies," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 2, pp. 264–278, 2007.

[7] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Transactions*

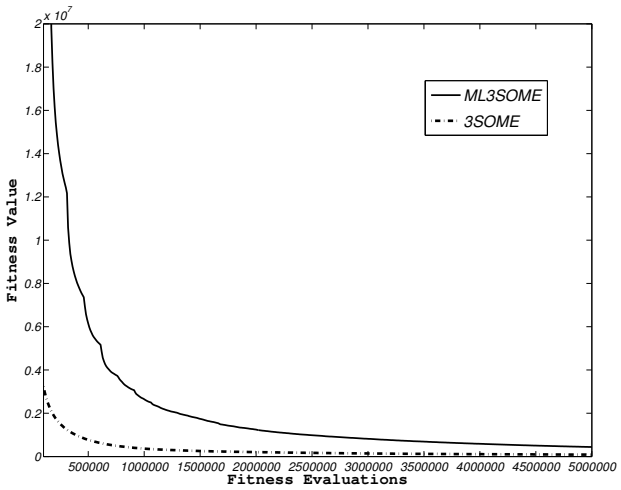

Fig. 5. Performance trends for f_{19} from [32] in 1000 dimensions

On Systems, Man and Cybernetics - Part B, vol. 36, no. 1, pp. 141–152, 2006.

[8] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[9] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling*, ser. Lecture Notes in Computer Science. Springer, 2000, vol. 2079, pp. 176–190.

[10] G. Kendall, P. Cowling, and E. Soubeiga, "Choice function and random hyperheuristics," in *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, 2002, pp. 667–71.

[11] A. V. Kononova, D. B. Ingham, and M. Pourkashanian, "Simple scheduled memetic algorithm for inverse problems in higher dimensions: Application to chemical kinetics," in *Proceedings of the IEEE World Congress on Computational Intelligence*, 2008, pp. 3906–3913.

[12] Y. S. Ong and A. J. Keane, "Meta-lamarkian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[13] P. Korošec, J. Šilc, and B. Filipič, "The differential ant-stigmergy algorithm," *Information Sciences*, 2011, to appear.

[14] M. N. Le, Y. S. Ong, Y. Jin, and B. Sendhoff, "Lamarckian memetic algorithms: local optimum and connectivity structure analysis," *Memetic Computing Journal*, vol. 1, no. 3, pp. 175–190, 2009.

[15] Q. C. Nguyen, Y. S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 604–623, 2009.

[16] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[17] J. E. Smith, "Coevolving memetic algorithms: A review and progress report," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, no. 1, pp. 6–17, 2007.

[18] E. L. Yu and P. N. Suganthan, "Ensemble of niching algorithms," *Information Sciences*, vol. 180, no. 15, pp. 2815–2833, 2010.

[19] J. E. Smith, "Estimating meme fitness in adaptive memetic algorithms for combinatorial problems," *Evolutionary Computation*, vol. 20, no. 2, pp. 165–188, 2012.

[20] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives," *IEEE Transactions on System Man and Cybernetics-part B, special issue on Memetic Algorithms*, vol. 37, no. 1, pp. 28–41, 2007.

[21] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 8, pp. 811–831, 2009.

[22] J. Tang, M. H. Lim, and Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 11, no. 9, pp. 873–888, 2007.

[23] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "An enhanced memetic differential evolution in filter design for defect detection in paper production," *Evolutionary Computation*, vol. 16, no. 4, pp. 529–555, 2008.

[24] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, "Ockham's razor in memetic computing: Three stage optimal memetic exploration," *Information Sciences*, vol. 188, pp. 17–43, 2012.

[25] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "SATzilla: Portfolio-based algorithm selection for SAT," *Journal of Artificial Intelligence Research*, vol. 32, pp. 565–606, 2008.

[26] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith, "Parameter control in evolutionary algorithms," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, F. G. Lobo, C. F. Lima, and Z. Michalewicz, Eds. Springer, 2007, vol. 54, pp. 19–46.

[27] H. H. Hoos, "Automated algorithm configuration and parameter tuning," in *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Springer, 2012, ch. 3, pp. 37–71.

[28] E. Mininno, F. Neri, F. Cupertino, and D. Naso, "Compact Differential Evolution," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 32–54, 2011.

[29] F. Neri, G. Iacca, and E. Mininno, "Disturbed exploitation compact differential evolution for limited memory optimization problems," *Information Sciences*, vol. 181, no. 12, pp. 2469–2487, 2011.

[30] L. Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 3052–3059.

[31] N. Hansen, A. Auger, S. Finck, R. Ros *et al.*, "Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2010.

[32] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the cec'2010 special session and competition on large-scale global optimization," University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): Hefei, Anhui, China, Tech. Rep., 2010.

[33] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.