

Study of the Influence of the Local Search Method in Memetic Algorithms for Large Scale Continuous Optimization Problems

Daniel Molina¹, Manuel Lozano², and Francisco Herrera²

¹ Universidad of Cádiz, Department of Computer Languages and Systems, Cádiz,
daniel.molina@uca.es,

² Universidad de Granada, Department of Computer Science and Artificial
Intelligence, Granada,
lozano@decsai.ugr.es, herrera@decsai.ugr.es

Abstract. Memetic algorithms arise as very effective algorithms to obtain reliable and high accurate solutions for complex continuous optimization problems. Nowadays, high-dimensional optimization problems are an interesting field of research. Its high dimension introduces new problems for the optimization process, making recommendable to test the behavior of optimization algorithms to large-scale problems. In memetic algorithms, the local search method is responsible of exploring the neighborhood of the current solutions; therefore, the dimensionality has a direct influence over this component. The aim of this paper is to study this influence. We design different memetic algorithms that only differ in the local search method applied, and they are compared using two sets of continuous benchmark functions: a standard one and a specific set with large-scale problems. The results show that high dimensionality reduces the differences among the different local search methods.

1 Introduction

It is now well established that hybridization of *evolutionary algorithms* (EAs) with other techniques can greatly improve the efficiency of search [4, 5]. EAs that have been hybridized with local search (LS) techniques are often called *memetic algorithms* (MAs) [18, 19, 16]. One commonly used formulation of MAs applies LS to members of the EA population after recombination and mutation, with the aim of exploiting the best search regions gathered during the global sampling done by the EA.

MAs comprising efficient local improvement processes on continuous domains (*continuous LS methods*) have been presented to address the difficulty of obtaining reliable solutions of high precision for complex continuous optimization problems [12, 15, 22, 21, 17]. In this paper, they will be named MACOs (MAs for continuous optimization problems).

Nowadays, high-dimensional optimization problems arise as a very interesting field of research, because they appear in many important new real-world

problems (bio-computing, data mining, etc.). Unfortunately, the performance of most available optimization algorithms deteriorates rapidly as the dimensionality of the search space increases [26]. Thus, the ability of being scalable for high-dimensional problems becomes an essential requirement for modern optimization algorithm approaches.

In recent years, it has been increasingly recognized that the influence of the continuous LS algorithm employed has a major impact on the search performance of MACOs [22]. There exists a group of continuous LS algorithms that stand out as brilliant local search optimizers. They include the Solis and Wets's algorithm [23], the Nelder and Mead's simplex method [20], and the CMA-ES algorithm [11]. However, on some occasions, they may become very expensive, because of the way they exploit local information to guide the search process. In this paper, they are called *intensive* continuous LS methods. Given the potential of these LS methods, a specific MACO model, called *MACO based on LS chains*, has been designed that can effectively use them as local search operators [17]. An instance of this MACO was experimental studied, which employed CMA-ES as local optimizer. The results showed that it was very competitive with state-of-the-art on both MACOs and EAs for continuous optimization problems. Particularly, significant improvements were obtained for the problems with the highest dimensionality among the ones considered for the empirical study (in particular, $D = 30$ and $D = 50$), which suggests that the application of this MACO approach to optimization problems with higher dimensionalities is indeed worth of further investigations.

In this paper, we undertake an extensive study of the performance of three MACOs based on LS chains that apply different instances of intense LS procedures on a specific set of benchmark functions with large scale problems, which was defined for the *CEC'2008 Special Session on Large Scale Global Optimization* ($D = 100$ and $D = 200$). In addition, since we are interested on investigating the ways these MACOs respond as dimensionality increases, we have evaluated them, as well, on the benchmark functions defined for the *CEC'2005 Special Session on Real-Parameter Optimization* ($D = 10, 30$, and 50).

The paper is set up as follows. In Section 2, we describe the different continuous LS methods analyzed. In Section 3, we review some important aspects of the MACO used for our study. In Section 4, we present the experimental study of three MACO instances based on continuous LS mechanisms on test problems with different dimensionalities. Finally, in Section 5, we provide the main conclusions of this work.

2 Continuous LS Methods

In this section, we present a detailed description of the continuous LS methods used in our empirical study. They are three well-known continuous local searchers: the Solis and Wets's algorithm [23], the Nelder and Mead's simplex method [20], and the CMA-ES algorithm [11]. Next, we present a detailed description of these procedures.

2.1 Solis and Wets' Algorithm

The classic *Solis and Wets' algorithm* [23] is a randomised hill-climber with an adaptive step size. Each step starts at a current point x . A deviate d is chosen from a normal distribution whose standard deviation is given by a parameter ρ . If either $x+d$ or $x-d$ is better, a move is made to the better point and a success is recorded. Otherwise, a failure is recorded. After several successes in a row, ρ is increased to move quicker. After several failures in a row, ρ is decreased to focus the search. It is worth noting that ρ is the strategy parameter of this continuous LS operator. Additionally, a bias term is included to put the search momentum in directions that yield success. This is a simple LS method that can adapt its size search very quickly. More details about this procedure may be found in [23].

2.2 Nelder-Mead Simplex Algorithm

This is a classical and very powerful local descent algorithm. A simplex is a geometrical figure consisting, in n dimensions, of $n+1$ points s_0, \dots, s_n . When a point of a simplex is taken as the origin, the n other points are used to describe vector directions that span the n -dimension vector space. Thus, if we randomly draw an initial starting point s_0 , then we generate the other n points s_i according to the relation $s_i = s_0 + \lambda e_j$, where the e_j are n unit vectors, and λ is a constant that is typically equal to one.

Through a sequence of elementary geometric transformations (reflection, contraction, expansion and multi-contraction), the initial simplex moves, expand or contracts. To select the appropriate transformation, the method only uses the values of the function to be optimized at the vertices of the simplex considered. After each transformation, a better vertex replaces the current worst one. A complete picture of this algorithm may be found in [20].

This method has the advantage of creating initially a simplex composed by movements in each direction. This is a very useful characteristic to deal with high-dimensional spaces.

2.3 CMA-ES method

The *covariance matrix adaptation evolution strategy* (CMA-ES) [11, 8] was originally introduced to improve the LS performances on evolution strategies. Even though CMA-ES reveals competitive global search performances [10], it has exhibited effective abilities for the local tuning of solutions; in fact, it was used as continuous LS algorithm to create *multi-start LS metaheuristics*, L-CMA-ES [2], and G-CMA-ES [1]. At the 2005 congress of evolutionary computation, these algorithms were ones of the winners of the real-parameter optimisation competition [24].

In CMA-ES, not only is the step size of the mutation operator adjusted at each generation, but so too is the step direction in the multidimensional problem space, i.e., not only is there a mutation strength per dimension but their combined update is controlled by a covariance matrix whose elements are

updated as the search proceeds. In this paper, we use the (μ_W, λ) CMA-ES model. For every generation, this algorithm generates a population of λ offspring by sampling a multivariate normal distribution:

$$x_i \sim N(m, \sigma^2 C) = m + \sigma N_i(0, C) \text{ for } i = 1, \dots, \lambda,$$

where the mean vector m represents the favourite solution at present, the so-called step-size σ controls the step length, and the covariance matrix C determines the shape of the distribution ellipsoid. Then, the μ best offspring are recombined into the new mean value using a *weighted intermediate recombination*: $\sum_{i=1}^{\mu} w_i x_{i:\lambda}$, where the positive weights sum to one. The covariance matrix and the step-size are updated as well following equations that may be found in [11] and [10]. The default strategy parameters are given in [10]. Only the initial m and σ parameters have to be set depending on the problem.

3 MACOs Based on LS Chains

In this section, we describe a MACO approach proposed in [17] that employs continuous LS methods as LS operators. It is a steady-state MA model that employs the concept of *LS chain* to adjust the LS intensity assigned to the intense continuous LS method. In particular, this MACO handles LS chains, throughout the evolution, with the objective of allowing the continuous LS algorithm to act more intensely in the most promising areas represented in the EA population. In this way, the continuous LS method may adaptively fit its strategy parameters to the particular features of these zones.

In Section 3.1, we introduce the foundations of steady-state MAs. In Section 3.2, we explain the concept of *LS chain*. Finally, in Section 3.3, we give an overview of the MACO approach presented in [17], which handles LS chains with the objective of make good use of intense continuous LS methods as LS operators.

3.1 Steady-State MAs

In *steady-state* GAs [25] usually only one or two offspring are produced in each generation. Parents are selected to produce offspring and then a decision is made as to which individuals in the population to select for deletion in order to make room for the new offspring. Steady-state GAs are *overlapping* systems because parents and offspring compete for survival. A widely used replacement strategy is to replace the worst individual only if the new individual is better. We will call this strategy the *standard replacement strategy*.

Although steady-state GAs are less common than generational GAs, Land [14] recommended their use for the design of *steady-state MAs* (steady-state GAs plus LS) because they may be more stable (as the best solutions do not get replaced until the newly generated solutions become superior) and they allow the results of LS to be maintained in the population.

3.2 LS Chains

In steady-state MAs, individuals resulting from the LS invocation may reside in the population during a long time. This circumstance allows these individuals to become starting points of subsequent LS invocations. In [17], Molina et al. propose to *chain* an LS algorithm invocation and the next one as follows:

The final configuration reached by the former (strategy parameter values, internal variables, etc.) is used as initial configuration for the next application.

In this way, the LS algorithm may continue under the same conditions achieved when the LS operation was previously halted, providing an *uninterrupted connection between successive LS invocations*, i.e., forming a *LS chain*.

Two important aspects that were taken into account for the management of LS chains are:

- Every time the LS algorithm is applied to refine a particular chromosome, a fixed LS intensity should be considered for it, which will be called *LS intensity stretch* (I_{str}).

In this way, a LS chain formed throughout n_{app} LS applications and started from solution s_0 will return the same solution as the application of the continuous LS algorithm to s_0 employing $n_{app} \cdot I_{str}$ fitness function evaluations.

- After the LS operation, the parameters that define the current state of the LS processing are stored along with the reached final individual (in the steady-state GA population). When this individual is latter selected to be improved, the initial values for the parameters of the LS algorithm will be directly available. For example, if we employ the Solis and Wets' algorithm (Section 2.1) as LS algorithm, the stored strategy parameter may be the current value of the ρ parameter. For the more elaborate CMA-ES (Section 2.3), the state of the LS operation may be defined by the covariance matrix (C), the mean of the distribution (\mathbf{m}), the size (σ), and some additional variables used to guide the adaptation of these parameters.

3.3 A MACO Model that Handles LS Chains

In this section, we introduce a MACO model that handles LS chains (see Figure 1) with the following main features:

1. It is a steady-state MA model.
2. It ensures that a fixed and predetermined local/global search ratio is always kept. With this policy, we easily stabilise this ratio, which has a strong influence on the final MACO behaviour. Without this strategy, the application of intense continuous LS algorithms may induce the MACO to prefer super exploitation.

1. Generate the **initial population**.
2. Perform the **steady-state GA** throughout n_{frec} evaluations.
3. Build the set S_{LS} with those individuals that **potentially may be refined** by LS.
4. Pick the **best individual** in S_{LS} (Let's c_{LS} to be this individual).
5. If c_{LS} belongs to an **existing LS chain** then
6. Initialise the LS operator with the **LS state** stored together with c_{LS} .
7. Else
8. Initialise the LS operator with the **default LS state**.
9. Apply the LS algorithm to c_{LS} with an LS intensity of I_{str} (Let's c_{LS}^r to be the resulting individual).
10. Replace c_{LS} by c_{LS}^r in the **steady-state GA population**.
11. Store the **final LS state** along with c_{LS}^r .
12. If (*not termination-condition*) go to step 2.

Fig. 1. Pseudocode algorithm for the MACO based on LS chains

3. It favours the enlargement of those LS chains that are showing promising fitness improvements in the best current search areas represented in the steady-state GA population. In addition, it encourages the activation of innovative LS chains with the aim of refining unexploited zones, whenever the current best ones may not offer profitability. The criterion to choose the individuals that should undergo LS is specifically designed to manage the LS chains in this way (Steps 3 and 4).

This MACO scheme defines the following relation between the steady-state GA and the intense continuous LS method (Step 2): *every n_{frec} number of evaluations of the steady-state GA, apply the continuous LS algorithm to a selected chromosome, c_{LS} , in the steady-state GA population.* Since we assume a fixed $\frac{L}{G}$ ratio, $r_{L/G}$, n_{frec} may be calculated using the following equation:

$$n_{frec} = I_{str} \frac{1 - r_{L/G}}{r_{L/G}}. \quad (1)$$

where n_{str} is the LS intensity stretch (Section 3.2) and $r_{L/G}$ is defined as the percentage of evaluations spent doing local search from the total assigned to the algorithm's run.

The following mechanism is performed to select c_{LS} (Steps 3 and 4):

1. Build the set of individuals in the steady-state GA population, S_{LS} that fulfils:
 - (a) They have never been optimized by the intense continuous LS algorithm, or
 - (b) They previously underwent LS, obtaining a fitness function improvement greater than δ_{LS}^{min} (a parameter of our algorithm).

2. If $|S_{LS}| \neq 0$, then apply the continuous LS algorithm to the best individual in this set. If this condition is not accomplished, the LS operator is applied to the best individual in the steady-state GA population.

With this mechanism, when the steady-state GA finds a new best so far individual, it will be refined immediately. In addition, the best performing individual in the steady-state GA population will always undergo LS whenever the fitness improvement obtained by a previous LS application to this individual is greater than the δ_{LS}^{min} threshold. The last condition is very important in order to avoid the overexploitation of search zones where the LS method may not make substantial progresses any more.

4 Experiments

In this section, we present the experimental study carry out with three MACOs based on LS chains that use the Solis and Wets's algorithm, the Nelder and Mead's simplex method, and the CMA-ES algorithm, respectively, on two different test suites:

- *CEC'2005 test suite*. Benchmark functions recommended for the *Special Session on Real Parameter Optimization organized in the 2005 IEEE Congress on Evolutionary Computation*. It is possible to consult in [24] the complete description of the functions, furthermore in the link the source code is included. The set of test functions is composed of 5 unimodal functions, 7 basic multimodal functions, 2 expanded multimodal functions, and 11 hybrid functions. Three dimension values were considered for these problems: $D = 10$, $D = 30$, and $D = 50$.
- *CEC'2008 test suite*. A set of benchmark functions specifically designed as large-scale problems, which were defined for *CEC'2008 Special Session on Large Scale Global Optimization*. This test suite is used to study the behavior of the MACOs on problems with high dimensionality. It consists in 7 test functions with two different dimension values: $D = 100$ and $D = 200$.

This section is structured in the following way. In Section 4.1, we describe the three instances of MACO used for the experiments. In Section 4.2, we detail the experimental setup and statistical methods that were used for this experimental study. In Section 4.3, we analyze the results on the *CEC'2005 test suite* and, in Section 4.4, the ones on the *CEC'2008 test suite*. In Section 4.5, we study how evolve the influence of the LS method when the dimensionality is increased. In Section 4.6, we compare our best model with two algorithms based in CMA-ES, L-CMA-ES [3] and G-CMA-ES[1], that give very good results in continuous optimization.

4.1 Three Instances of MACO Based on LS Chains

In this section, we build three instances of the MACO model described in Figure 1, which apply the Solis and Wets's algorithm (Section 2.1), the Nelder and

Mead’s simplex method (Section 2.2), and the CMA-ES algorithm (Section 2.3), respectively, as intense continuous LS algorithms.

Next, we list their main features:

Steady-state GA. It is a real-coded steady-state GA [13] specifically designed to promote high population diversity levels by means of the combination of the BLX- α crossover operator (see [13]) with a high value for its associated parameter ($\alpha = 0.5$) and the *negative assortative mating* strategy [6]. Diversity is favored as well by means of the BGA mutation operator (see [13]).

Continuous LS algorithms. The three instances follow the MACO approach that handles LS chains, with the objective of tuning the intensity of the three LS algorithms considered, which are employed as intense continuous LS operators. In particular, the application of CMA-ES for refining an individual, C_i , is carried out following the next guidelines:

- We consider C_i as the initial mean of distribution (\mathbf{m}).
- The initial σ value is half of the distance of C_i to its nearest individual in the steady-state GA population (this value allows an effective exploration around C_i).

CMA-ES will work as local searcher consuming I_{str} fitness function evaluations. Then, the resulting solution will be introduced in the steady-state GA population along with the current value of the covariance matrix, the mean of the distribution, the step-size, and the variables used to guide the adaptation of these parameters (B, BD, D, p_c and p_σ). Latter, when CMA-ES is applied to this inserted solution, these values will be recovered to proceed with a new CMA-ES application. When CMAE-ES is performed on solutions that do not belong to existing chains, default values, given in [10], are assumed for the remaining strategy parameters.

The Solis and Wets’s algorithm and the Nelder and Mead’s simplex method are used in a similar fashion.

Parameter setting. For the experiments, the three MACO instances apply BLX- α with $\alpha = 0.5$. The population size is 60 individuals and the probability of updating a chromosome by mutation is 0.125. The n_{ass} parameter associated with the negative assortative mating is set to 3. They use $I_{str} = 500$ and the value of the $\frac{L}{G}$ ratio, $r_{L/G}$, was set to 0.5, which represents an equilibrated choice. Finally, a value of 10^{-8} was assigned to the δ_{LS}^{min} threshold. All these parameter values are recommended in [17].

4.2 Experimental Setup and Statistical Analysis

The experiments have been carried out following the instructions indicated in the documents associated to each set of benchmark functions. The main characteristics are:

- Each algorithm is run 25 times for each test function, and the error average of the best individual of the population is computed.
- The study has been made with dimensions $D = 10$, $D = 30$, and $D = 50$ for the *CEC'2005 test suite* and $D = 100$ and $D = 200$ for the *CEC'2008 test suite*.
- The maximum number of fitness evaluations is $10,000 \cdot D$ for the *CEC'2005 test suite* and $5,000 \cdot D$ for the *CEC'2008 test suite*.
- Each run stops either when the error obtained is less than 10^{-8} , or when the maximal number of evaluations is achieved.

Table 1. Parameters used for the experiments

Parameters	Functions Benchmark CEC'2005	Functions Benchmark CEC'2008
Execution Numbers	25	25
Dimensions	10, 30, 50	100, 200
Maximum Evaluations Number (NE)	$10,000 * D$	$5,000 * D$
Stopping Criterion	NE achieved or $error < 1e - 8$	NE achieved

To analyse the results we have used *non-parametric tests*, because it has been shown that parametric tests can not be applied with security for these test suites [7]. We have applied the non-parametric recommended in [7], thus it can be consulted this paper to obtain a detailed explanations of them. Next, these tests are briefly explained:

- The *Iman-Davenport's* test. This non-parametric test is used for answering this question: *In a set of k samples (where $k \geq 2$), do at least two of the samples represent populations with different median values?*. It is a non-parametric procedure employed in a hypothesis testing situation involving a design with two or more samples; therefore, it is a multiple comparison test that aims to detect significant differences between the behaviour of two or more algorithms.
- The *Holm's* test as a post-hoc procedure, to detect whose algorithms are worse than the algorithm with best results. This test only can be applied if the Iman-Davenport's test detects a significant difference. It sequentially checks the hypotheses ordered according to their significance. If p_i is lower than $\alpha/(k - i)$, the corresponding hypothesis is rejected and the process continues. In other case, this hypothesis and all the remaining hypotheses are maintained as supported.

4.3 Results for the CEC'2005 Test Suite

Firstly, we applied the *Iman-Davenport's* test to see if there is a significant difference between the different MACO instances, considering the three different dimension values. Table 2 shows the results of this statistical test.

Table 2. Results of the Iman-Davenport's test comparing the MACO instances for $D = 10, 30,$ and 50

<i>Dimension</i>	Iman-Davenport value	Critical value	Sig. differences?
10	2,2991	3,19	No
30	9,2012	3,19	Yes
50	5,5276	3,19	Yes

We may observe in Table 2 that for dimension 30 and 50 there exist significant differences among the rankings of the algorithms (the statistical value is greater than the critical one, 3,19). Then, attending on these results, we compare the MACO instances by means of the *Holm's* test. Table 3 shows the results.

Table 3. Comparison using the Holm's test of the MACO instances with respect the best one (based on CMA-ES), with $D = 30, 50$

<i>Dimension</i>	<i>LSMethod</i>	z	p -value	α/i	Sig. differences?
30	Simplex	3,2527	0,00114	0,0250	Yes
	Solis Wets	2,4749	0,01333	0,0500	Yes
50	Simplex	2,9698	0,00298	0,0250	Yes
	Solis Wets	2,1213	0,03389	0,0500	Yes

From Table 3, we can see that the election of the continuous LS method is crucial for $D = 30$ and $D = 50$. For $D = 10$, the problems become easy and all instances achieve similar results. In this case, the MACO based on CMA-ES is the best algorithm.

4.4 Results for CEC'2008 Test Suite

In this section, we compare the different instances on the large-scale problems in the CEC'2008 test suite. The error values are shown in the Appendix, in Table 7. First, we have applied the *Iman-Davenport's* test. Table 4 shows the results.

Table 4. Results of the Iman-Davenport's test comparing the MACO instances for $D = 100$ and $D = 200$

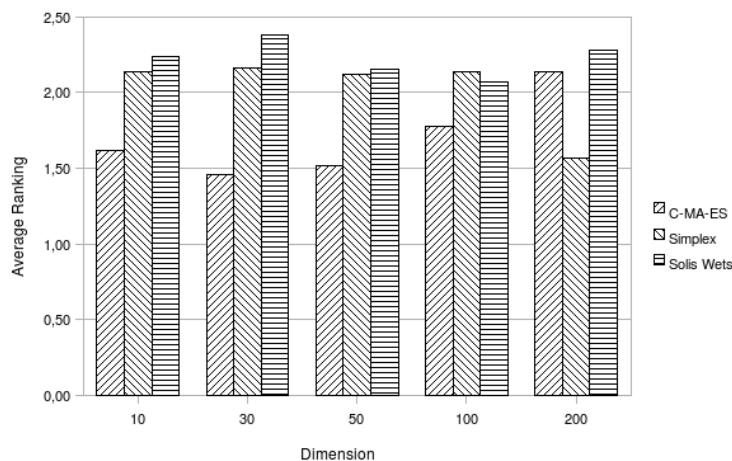
<i>Dimension</i>	Iman-Davenport value	Critical value	Sig. differences?
100	0,2222	3,89	No
200	1,0000	3,89	No

We can see in Table 4 that for high dimensionality (greater or equal than 100) there are no differences between the results achieved by MACO instances based on the different LS methods. Thus, when MACOs based on LS chains are applied on problems with high dimensionality, the election of the continuous LS method does not provide significant statistical differences.

4.5 Remarks from the Results

In the previous sections, we have used non-parametric test to determine when the differences between the MACO instances are statistically significant. Figure 2 shows the average rankings of these algorithms for the different dimension values considered in this study, with the aim of showing, graphically, the differences between them.

Fig. 2. Mean ranking of each MACO instance on the problems with different dimensionalities



We can obtain the following conclusions from this table:

- The improvements obtained from the best LS method, CMA-ES, decrease as dimensionality increases (for $D = 10$, the difficulty is not enough to obtain a clear difference), achieving, finally, a situation ($D = 200$), where it might not be the best LS method. The difference between the Solis Wets' method and simplex method is also reduced.
- For $D = 200$, the simplex method outperforms the others continuous LS methods.

4.6 Comparisons with other CMA-ES algorithms

In the *CEC'2005 Special Session on Real-Parameter Optimization* two algorithms arise as the best algorithms [9]: the L-CMA-ES [3] and G-CMA-ES [1], both of them invoke CMA-ES instances that specifically emphasise the local refinement abilities of this algorithm. In this section we compare our instance with the CMA-ES algorithm (called MA-CMA-ES in this section) with them.

CEC'2005 test suite. In [17] it is proven that, for CEC'2005 test suite, our model with CMA-ES exhibits overall better performance than L-CMA-ES and G-CMA-ES, in particular, at higher dimensionality, where the proposed hybridisation method outperforms the others. With the higher dimension, it is the best algorithm, and statistically better than the pure restart local search strategy (L-CMA-ES). In [17] this analysis is explained in detail.

Table 5. Results of the Iman-Davenport's test, comparing L-CMA-ES, G-CMA-ES, and MA-CMA-ES for $D = 100$ and $D = 200$

<i>Dimension</i>	<i>Iman-Davenport value</i>	<i>Critical value</i>	<i>Sig. differences?</i>
100	1,0000	3,89	No
200	0,3913	3,89	No

CEC'2008 Test Suite. Table 5 shows the results of Iman-Davenport's test comparing the two based CMA-ES algorithms (G-CMA-ES, L-CMA-ES) with MA-CMA-ES, using the CEC'2008 test suite, for dimension 100 and 200. Also, Table 6 shows the results for each one of these algorithms for CEC'2008.

Table 6. Results of each CMA-ES based algorithm for the CEC'2008 test suite

LS Method	Dimension 100						
	F1	F2	F3	F4	F5	F6	F7
G-CMA-ES	1,04E-13	3,41E-10	7,97E-1	2,29E+2	1,72E-12	1,71E+1	-8,63E+2
L-CMA-ES	4,49E-14	3,98E-11	6,38E-1	2,01E+2	9,44E-13	2,13E+1	-8,58E+2
MA-CMA-ES	9,50E-9	5,89E-3	1,65E+2	2,31E+0	9,73E-9	9,76E-9	-1,42E+3
LS Method	Dimension 200						
	F1	F2	F3	F4	F5	F6	F7
G-CMA-ES	9,72E-14	6,96E-10	9,56E-1	4,69E+2	5,06E-12	1,63E+1	-3,96E+8
L-CMA-ES	4,99E-14	2,53E-9	4,78E-1	5,39E+2	2,74E-12	2,14E+1	-3,12E+7
MA-CMA-ES	9,69E-9	1,57E+0	2,31E+2	1,28E+1	2,96E-4	9,87E-9	-2,70E+3

Although we can observe that there is no statistically differences between them, from Table 6 we can obtain several conclusions. First, in unimodal func-

tions (F1 and F2), L-CMA-ES and G-CMA-ES obtain the best results. For more complex functions (F4, F6, and F7 in dimension 100) our model with CMA-ES obtain clearly the best results. In general, for easy functions where every algorithm obtains a good result (a error lower than 10^{-8}), the restart models achieve the best results, due to a greater exploitation in the search. For more complex functions, our model can avoid local optima and obtain better results, achieving a clear improvement over the multistart models.

5 Conclusions

In this paper, we have built several instances of MACO based on LS chains that differ in the intense continuous LS method applied: the Solis and Wets's algorithm, the Nelder and Mead's simplex method, and the CMA-ES algorithm. We have compared their performance on two set of test problems including problems with different dimensionality, the *CEC'2005* and *CEC'2008* test suites. The main conclusions obtained are:

- Our instance with CMA-ES gives better results than other restart algorithms that use CMA-ES.
- The dimensionality strongly affects the performance of the MACO instances based on the different LS methods; not only the differences among them, but also, which algorithm results the best one.
- We have observed that, while there are significant differences for medium and low dimension values (and the election of the LS method is crucial), when the dimensionality increases, the differences between them are reduced.
- The simplex method improves as the dimension of the problems increase. Maybe the reason is that this is an algorithm capable of exploring better the changes in reduced group of dimensions. This is an aspect to consider for future LS methods specifically designed for high-dimension problems.

A Appendix: Results for CEC'2008

Table 7. Results of each MACO instance for the CEC'2008 test suite

LS Method	Dimension 100						
	F1	F2	F3	F4	F5	F6	F7
CMA-ES	9,50E-9	5,89E-3	1,65E+2	2,31E+0	9,73E-9	9,76E-9	-1,42E+3
Simplex	9,43E-9	5,33E+0	8,21E+1	2,83E+0	6,90E-4	8,93E-8	-1,45E+3
Solis Wets	9,92E-9	1,71E+1	1,50E+2	2,55E+0	5,91E-4	9,94E-9	-1,45E+3
LS Method	Dimension 200						
	F1	F2	F3	F4	F5	F6	F7
CMA-ES	9,69E-9	1,57E+0	2,31E+2	1,28E+1	2,96E-4	9,87E-9	-2,70E+3
Simplex	9,66E-9	1,74E+1	1,22E+2	3,26E+0	9,82E-9	9,88E-9	-2,76E+3
Solis Wets	9,95E-9	4,15E+1	2,85E+2	5,29E+0	2,07E-3	9,98E-9	-2,73E+3

Bibliography

- [1] A. Auger and N. Hansen. A Restart CMA Evolution Strategy with Increasing Population Size. In *2005 IEEE Congress on Evolutionary Computation*, pages 1769–1776, 2005.
- [2] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proc. of the 2005 IEEE Congress on Evolutionary Computation*, pages 1777,1784, 2005.
- [3] A. Auger and N. Hansen. Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In *2005 IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.
- [4] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.
- [5] W. Banzhaf et al., editor. *Optimizing global-local search hybrids*. Morgan Kaufmann, San Mateo, California, 1999.
- [6] C. Fernandes and A. Rosa. A Study of non-Random Matching and Varying Population Size in Genetic Algorithm using a Royal Road Function. *Proc. of the 2001 Congress on Evolutionary Computation*, pages 60–66, 2001.
- [7] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics. In Press.*, 2008.
- [8] Hansen, Müller, and Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.
- [9] N. Hansen. Compilation of Results on the CEC Benchmark Function Set. Technical report, Institute of Computational Science, ETH Zurich, Switzerland, 2005. available as http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf.
- [10] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *Proc. of the Parallel Problem Solving for Nature - PPSN VIII, LNCS 324*, pages 282–291, 2004.
- [11] N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *Proceeding of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pages 312–317, 1996.
- [12] W.E. Hart. *Adaptive Global Optimization With Local Search*. PhD thesis, Univ. California, San Diego, CA., 1994.
- [13] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling Real-coded Genetic Algorithms: Operators and Tools for the Behavioral Analysis. *Artificial Intelligence Reviews*, 12(4):265–319, 1998.
- [14] M.W. Shannon Land. *Evolutionary Algorithms with Local Search for Combinational Optimization*. PhD thesis, Univ. California, San Diego, CA., 1998.
- [15] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded Memetic Algorithms with Crossover Hill-climbing. *Evolutionary Computation*, 12(2):273–302, 2004.

- [16] Peter Merz. *Memetic Algorithms for Combinational Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Gesamthochschule Siegen, 2000.
- [17] Daniel Molina, Manuel Lozano, Carlos García-Martínez, and Francisco Herrera. Memetic algorithms for continuous optimization based on local search chains. *Evolutionary Computation*. *In Press.*, 2008.
- [18] P.A. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California, 1989.
- [19] P.A. Moscato. *Memetic algorithms: a short introduction*, pages 219–234. McGraw-Hill, London, 1999.
- [20] J.A. Nelder and R. Mead. A simplex method for functions minimizations. *Computer Journal*, 7(4):308–313, 1965.
- [21] Nasimul Noman and Hitoshi Iba. Accelerating differential evolution using an adaptive local search. In *IEEE Transactions on evolutionary Computation*, volume To appear, 2008.
- [22] Y. Soon Ong and A. J. Keane. Meta-Lamarckian Learning in Memetic Algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):99–110, 2004.
- [23] F. J. Solis and R. J. Wets. Minimization by Random Search Techniques. *Mathematical Operations Research*, 6:19–30, 1981.
- [24] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical report, Nanyang Technical University, 2005.
- [25] G. Sywerda. Uniform crossover in genetic algorithms. In J. Schaffer, editor, *Proc. of the International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, San Francisco, CA, USA, 1989.
- [26] F. van den Bergh and A.P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, pages 225–239, 2004.