Memetic Algorithms for Continuous Optimisation Based on Local Search Chains

Daniel Molina daniel.molina@uca.es
Department of Computer Engineering, University of Cadiz, Cadiz, 11003, Spain

Manuel Lozano lozano@decsai.ugr.es
Department of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain

Carlos García-Martínez cgarcia@uco.es
Department of Computing and Numerical Analysis, University of Córdoba, 14071, Córdoba, Spain

Francisco Herrera herrera@decsai.ugr.es
Department of Computer Science and Artificial Intelligence, University of Granada, 18071, Granada, Spain

Abstract

Memetic algorithms with continuous local search methods have arisen as effective tools to address the difficulty of obtaining reliable solutions of high precision for complex continuous optimisation problems. There exists a group of continuous local search algorithms that stand out as exceptional local search optimisers. 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 local search methods. Given the potential of this type of local optimisation methods, it is interesting to build prospective memetic algorithm models with them.

This paper presents the concept of local search chain as a springboard to design memetic algorithm approaches that can effectively use intense continuous local search methods as local search operators. Local search chain concerns the idea that, at one stage, the local search operator may continue the operation of a previous invocation, starting from the final configuration (initial solution, strategy parameter values, internal variables, etc.) reached by this one. The proposed memetic algorithm favours the formation of local search chains during the memetic algorithm run with the aim of concentrating local tuning in search regions showing promise. In order to study the performance of the new memetic algorithm model, an instance is implemented with CMA-ES as an intense local search method. The benefits of the proposal in comparison to other kinds of memetic algorithms and evolutionary algorithms proposed in the literature to deal with continuous optimisation problems are experimentally shown. Concretely, the empirical study reveals a clear superiority when tackling high-dimensional problems.

Keywords

Continuous optimisation, memetic algorithms, continuous local search algorithms, adaptive local search intensity, genetic algorithms.

1 Introduction

It is now well established that hybridisation of evolutionary algorithms (EAs) with other techniques can greatly improve the efficiency of search (Davis, 1991; Goldberg and Voessner, 1999). EAs that have been hybridised with local search techniques (LS) are often called memetic algorithms (MAs; Moscato, 1989, 1999; Merz, 2000; Krasnogor and Smith, 2005). 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. Thus, an important aspect concerning MAs is the trade-off between the exploration abilities of the EA and the exploitation abilities of the LS technique used (Krasnogor and Smith, 2001), that is, MAs should combine their two ingredients following a hybridisation scheme that allows them to work in a cooperative way, ensuring synergy among exploration and exploitation.

Many real-world problems may be formulated as optimisation problems of parameters with variables in continuous domains (i.e., continuous optimisation problems). Over the past few years, an increasing interest has arisen in solving this kind of problem using different EA models. They include real-coded genetic algorithms (Herrera et al., 1998), evolution strategies (Beyer and Schwefel, 2002), evolutionary programming (Lee and Yao, 2004), particle swarm optimisation (Kennedy and Eberhart, 1995), and differential evolution (Storn and Price, 1997). A common characteristic of these EAs is that they evolve chromosomes that are vectors of floating point numbers, directly representing problem solutions (hence, they may be called real-coded EAs). Nevertheless, for function optimisation problems in continuous search spaces, an important difficulty must be addressed: *solutions of high precision must be obtained by the solvers* (Kita, 2001). MAs comprising efficient local improvement processes on continuous domains (i.e., continuous LS methods) have been presented to deal with this problem (Hart, 1994; Renders and Flasse, 1996). In this paper, they will be named MACOs (MAs for continuous optimisation problems). Most MACO instances employ real-coded EAs as an EA component (Lozano et al., 2004); however, some MACO researchers prefer traditional binary-coded genetic algorithms (Ong and Keane, 2004; Ong et al., 2006).

Most well-known continuous LS algorithms make use of explicit strategy parameters (e.g., step sizes) to guide the search. Generally, they adapt these parameters with the purpose of increasing the likelihood of producing more effective solutions. Due to their explicit parameter adaptation, these algorithms may require a substantial number of evaluations (high LS intensity) to achieve adequate styles of traversal of solution space to follow certain paths leading to precise final solutions. In this paper, we refer to this kind of LS procedures as intense continuous LS algorithms.

The design of MACO models embedding intense continuous LS operators arises as a profitable way to improve these algorithms, because, nowadays, there are powerful modern metaheuristics available that may be catalogued as intense continuous LS algorithms, which can make their integration into MACOs particularly interesting. However, the incorporation of expensive continuous LS algorithm (potentially wasting too many function evaluations in a MACO) should be made with great care, because the relationship between exploration and exploitation in the MACO may get out of balance.

We are interested in the design of specific MACO models that try to make the most of intense continuous LS operators (i.e., providing them with enough LS intensity) without compromising the synergy between the EA and the proper continuous LS algorithm.

In this paper, we propose a MACO approach that employs the concept of the *LS chain* to adjust the LS intensity assigned to the intense continuous LS method. In our approach, an individual resulting from an LS invocation may later become the initial point of a subsequent LS application, which will adopt the final strategy parameter values achieved by the former as its initial values. Although particular individuals are subject to a limited amount of LS, the occurrence of this chaining process, throughout MA evolution, allows the operation of the LS operator to be extended in certain search zones. More concretely, our MACO proposal encourages the formation of concrete LS chains with the aim of focusing the action of the continuous LS algorithm on promising areas where it might obtain improvements. In this way, the continuous LS method may *adaptively* fit its strategy parameters to the particular features of these zones. In our study, we use CMA-ES (Hansen and Ostermeier, 2001) as our intense continuous LS algorithm, which stands out as an excellent local searcher.

The paper is set up as follows. In Section 2, we review some important aspects of MAs. In Section 3, we overview some contemporary research devoted to improving the behaviour of the continuous LS operator in the MACO scheme. In addition, we outline some relevant features of intense continuous LS algorithms and describe the CMA-ES algorithm. In Section 4, we present the concept of LS chain and explain the way they should be managed in order to obtain profitable integration of intense continuous LS methods into MACOs. In Section 5, we design the experimental framework that allows us to study the behaviour of the proposed MACO model. In particular, we present an instance based on CMA-ES. In Section 6, we analyse experimentally the behaviour of the instance and compare its results with other algorithms proposed in the literature for continuous optimisation problems. Finally, in Section 7, we provide the main conclusions of this work and examine future research lines. In Appendix A, we explain the statistical tests that were used for the experimental study and, in Appendix B, we include tables with the results of certain algorithms.

2 Fundamentals of Memetic Algorithms

Recently, hybrid heuristics have been a hot topic in the fields of both computer science and operational research. It assumes that combining the features of different methods in a complementary fashion may result in more robust and effective optimisation tools. MAs may be considered as the union of population-based global search and local improvements that are inspired by Darwinian principles of natural evolution and Dawkins' notion of a meme, defined as a unit of cultural evolution that is capable of local refinements (Krasnogor and Smith, 2005). In essence, an MA is a search strategy in which a population of optimising agents synergistically cooperates and competes. This behaviour can be accomplished by using LS strategies within a population based search technique such as an EA, although it must be noted that the MA paradigm does not simply reduce itself to this particular scheme. In fact, other hybrid paradigms combining EAs with problem dependent heuristics, such as approximation algorithms, truncated exact methods, and specialised recombination operators, belong to the MA family. In diverse contexts, MAs have also been used under the name of hybrid EAs, Baldwinian EAs, Lamarkian EAs, or genetic local search.

Many different instantiations of MAs have been reported across a wide variety of application domains that range from scheduling and floor-planning problems, to pattern recognition, vehicle routing, control systems, aircraft, and drug design, to name but a few. This large body of evidence has revealed that MAs not only converge to high-quality solutions, but also search vast, and sometimes noisy, solution spaces more efficiently than their conventional counterparts. Thus, MAs are the preferred methodology for many real-world applications, and nowadays receive more attention (Ong et al., 2007; Hart et al., 2004a,b). Among other reasons, MAs are preferred because:

- MAs can be designed adopting the divide and conquer strategy. Their two components may be conceived and implemented independently from each other. In this way, their development becomes easy and, in addition, their specific functioning may also be profitable (as compared with search algorithms that attempt to combine by themselves both exploration and exploitation).
- MAs are intrinsically concerned with exploiting all available knowledge about the problem under study (Bonissone et al., 2006). The inclusion of problem knowledge is also supported by strong theoretical results; the No Free Lunch Theorem (Wolpert and Macready, 1997) states that a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge it incorporates. This fact clearly underpins the exploitation of problem knowledge intrinsic to MAs (Moscato and Cotta, 2003).
- MAs have arisen as a promising approach for improving the convergence speed to the Pareto front of EAs for multiobjective optimisation problems, which actually concentrate increasing research efforts (Ishibuchi et al., 2003; Liu et al., 2007).

The aim of this section is to briefly overview the main characteristics of MAs. In Section 2.1, we get more deeply into their components and, in Section 2.2, we discuss the convenience of adaptively controlling different MA parameters.

2.1 MA Components

Exploration and exploitation are two major issues when designing a global search method. Exploration is important for the search algorithm to achieve global optimality, whereas exploitation searches around the neighbourhood of good solutions to produce higher quality solutions. A search algorithm should strike a tactical balance between the two sometimes-conflicting goals. MAs attempt to accomplish this compromise by putting together two specialised components: an EA that may assume the task of exploring the search space, and an LS algorithm that refines promising individuals being evolved by the EA. The rationale behind MAs is to provide an effective and efficient global optimisation method by compensating for the deficiency of EAs in local exploitation and the inadequacy of LSs in global exploration.

MAs should combine their two ingredients following a hybridisation scheme that allows them to work in a cooperative way, attaining a profitable synergy among their associated exploration and exploitation features. For successful incorporation of an LS algorithm in an EA, several issues must be resolved, as follows.

2.1.1 Election of the LS Algorithm

In recent years, it has been increasingly recognised that the particular choice of LS operator will have a major impact on the efficacy of the MAs (Smith, 2007). With so many LS algorithms available in the literature, it is almost impossible to know which is most relevant to a problem when one has only limited knowledge of its cost surface before one starts (Ong and Keane, 2004). Moreover, LS algorithms by themselves are known to work very differently with different design problems, even among problems from the same design domain. Depending on the complexity of a design problem, LS algorithms that may have proven to be successful in the past might not work so well, or at all, on others.

2.1.2 Selection of the Individuals That Should Be Improved by LS

This election is very important because a wrong choice may mean that the MA might not exploit, with adequate intensity, some promising regions of the search space. Normally, the LS algorithm is invoked to refine the new chromosomes created from the application of crossover and mutation. All these individuals may undergo LS, such as scatter search (Laguna and Martí, 2003) does. However, in this case, the additional fitness function evaluations required for the LS search increment considerably the computational cost of the MA, which may be prohibitive for many problems. Thus, a parameter, called LS probability, p_{LS}, was introduced (Hart, 1994), which determines the probability of applying LS to every created chromosome. In Hart (1994), $p_{\mathrm{LS}} = .0625$ was considered appropriate for many practical cases. Two alternative criteria have been considered as well. In Chelouah and Siarry (2003), the LS algorithm is applied after the EA has detected a new promising region, and in Liang and Suganthan (2005), the LS procedure improves, periodically, the N best individuals in the population.

2.1.3 Computational Cost Allocated for Every LS Algorithm Application

The number of fitness function evaluations required by the LS algorithm during their operation determines its cost. In this paper, this number will be called *LS intensity*. It is fundamental to identify a proper intensity for the LS, because an LS that is too short may be unsuccessful at exploring the neighbourhood of the solution and therefore unsuccessful at improving the search quality. On the other hand, too long an LS may backfire by consuming additional fitness evaluations unnecessarily. However, we should point out that the more commonly employed technique for choosing the LS intensity involves the use of a single value for this parameter (typically that value producing the highest accuracy of the LS procedure) and keeps it constant during the entire optimisation.

The local/global search ratio ($\frac{L}{G}$ratio; Lozano et al., 2004) kept by an MA (defined as the percentage of evaluations spent doing local search from the total assigned to the algorithm's run) is mainly governed by the LS intensity and the number of chromosomes that undergo LS. This ratio determines the trade-off between the exploration abilities of the EA, and the exploitation abilities of the LS algorithm, and then it has an important influence on the final performance of the MA on a particular problem.

2.2 Adaptive MAs

Adaptation of parameters and operators has become a very promising research field in MAs (Ong et al., 2006; Smith, 2007). Adaptive MAs were designed to address the three issues detailed in the previous section:

2.2.1 Adaptive MAs with Multiple LS Operators

A number of authors have investigated and proposed mechanisms for choosing between a set of predefined LS operators that may be used during a particular MA run. Essentially, all these approaches maintain a pool of LS operators, being available to be used by the MA and, at each decision point, choose which one to apply. This form of adaptive MAs promotes both cooperation and competition among various problem-specific LS algorithms and favours neighbourhood structures containing high quality solutions that may be achieved at low computational effort.

Ong et al. (2006) present an excellent recent review of work in this field. This encompasses the works of Krasnogor on multimemetic algorithms (Krasnogor, 2002;

Krasnogor and Smith, 2000, 2001) and hyper-heuristics (Burke and Smith, 2000). In an interesting extension to the use of a set of fixed strategies, Krasnogor and Gustafson have recently proposed a grammar for self-generating MAs, which specifies, for instance, when LS takes place (Krasnogor and Gustafson, 2004). In this research line, Smith (2007) proposed the coevolving MA, a system within which the definitions of LS operators applied within the MA may be changed during the course of optimisation. It maintains two populations; one of genes encoding for candidate solutions and one of memes encoding for LS operators to be used within the MA. The results showed that the coevolving MA was able to discover and exploit certain forms of structure and regularities within the problems.

2.2.2 Adaptive MAs That Control the $\frac{L}{G}$ Ratio

Some adaptive MA approaches that were presented adjust the LS intensity or LS probability with the aim of determining an effective $\frac{L}{G}$ ratio. Simulated heating (Bambha et al., 2004) is one of them. It is a dynamic mechanism that varies the LS intensity gradually with the progress of the search. The idea behind simulated heating is to increase the time allotted to each LS invocation during the optimisation process; thus generating low accuracy of the LS algorithm at the beginning and high accuracy at the end. The goal is to focus on the global search at the beginning and to find promising regions of the search space first; for this phase, run with low accuracy, which in turn allows a greater number of optimisation steps of the global search. Afterward, more time is spent in order to improve the solutions found or to assess them more accurately. As a consequence, fewer global search operations are possible during this phase of optimisation.

Hart (1994) proposed two different strategies for adaptively calculating the probability with which LS is applied to each new chromosome. The two strategies are the fitness-based and the distribution-based strategies. Fitness-based adaptive methods use the fitness information in the population to bias the LS toward individuals that have better fitness. These methods assume that individuals with better fitness are more likely to be in basins of attraction of good local optima. Distribution-based adaptive methods use redundancy in the population to avoid performing unnecessary local searches. In particular, selected solutions will be far away from each other, and ideally span as much of the search space as the population itself. Lozano et al. (2004) introduced a fitness-based adaptive mechanism that determines the probability with which every solution should receive the application of a crossover-based LS algorithm. Authors concluded that the mechanism allows the $\frac{L}{G}$ ratio to be adjusted according to the particularities of the search space, allowing significant performance to be achieved for problems with different difficulties.

3 Continuous LS Algorithms

Reaching accurate solutions becomes of great importance for function optimisation problems in continuous search spaces. Different MACO models have been presented to deal with this problem. The main idea is to use efficient local improvement processes on continuous domains, for example, hill-climbers for nonlinear optimisation, such as Quasi-Newton, conjugate gradient, SQP, random linkage, Solis and Wets' algorithm, and Nelder and Mead's simplex method. Examples of MACOs may be found in Hart (1994); Hart et al. (2000); Joines and Kay (2002); Houck et al. (1997); Mühlenbein et al. (1991); Renders and Bersini (1994); Renders and Flasse (1996); Rosin et al. (1997); Zhang and Shao (2001); and Wei and Zhao (2005).

In his pioneering work on MACOs, Hart (1994) demonstrated that the choice of continuous LS algorithm affects the performance of MACOs significantly on a variety of benchmark problems of diverse properties. Furthermore, in recent years, it has been increasingly recognised that the influence of the continuous LS algorithm employed has a major impact on the search performance of MACOs (Ong and Keane, 2004; Ong et al., 2006; Smith, 2007). This is why most MACO research has been focused on analysing possible ways to improve the operation of this component. Foremost investigations were conducted to tackle two relevant problems related with classic continuous LS operators, which are analysed below.

- **A Particular Continuous LS Method May Be Effective for Some Class of Problems but Not for Others.** Each continuous LS algorithm instance directs the search toward a different zone in the neighbourhood of the solutions. The quality of the solutions that belong to the visited region depends on the particular problem to be solved. This means that different continuous LS algorithms perform differently with respect to different problems, even at the different stages of the memetic process in the same problem. This problem motivated the design of adaptive MACOs with multiple continuous LS operators (Ong and Keane, 2004; Ong et al., 2006; Caponio et al., 2007).

- **Most Existing Continuous LS Algorithms May Require High LS Intensity Values to Work Effectively.** Most well-known continuous LS algorithms make use of explicit strategy parameters (e.g., step sizes) to guide the search. Generally, they adapt the parameters in such a way that the moves being made may be of varying sizes, depending on the success of previous steps, with the purpose of increasing the likelihood of producing more effective solutions. The rules for updating parameters capture some lawful operation of the dynamics of the algorithm over a broad range of problems. Due to their explicit parameter adaptation, these continuous LS algorithms may require high LS intensity values to adapt their strategy parameters to the local topography of the search areas being refined.

The incorporation of expensive continuous LS algorithms wasting too many function evaluations into a MACO should be made judiciously. On the one hand, more time allotted to each continuous LS algorithm invocation implies more thorough local optimisation at the expense of a smaller number of achievable function evaluations, for example, smaller numbers of generations explored with the EA. On the other hand, the MACO search may become too focused; the intense continuous LS algorithm may be applied only to a small proportion of the chromosomes being attained by the MACO (i.e., a small number of search regions received great attention). This means that certain promising search regions that deserve attention may not be refined with enough interest and fall into oblivion, which may be very detrimental for the complex problems. These problems, derived from the use of intense continuous LS algorithms, may incapacitate the MACO to obtain profitable synergetic effects between the EA and the continuous LS algorithm. A direct reaction against this serious difficulty was the design of quick continuous LS algorithms. These are alternative continuous LS techniques that allow high quality solutions to be reached requiring low LS intensity values. One of the approaches that recently received attention concerns real-parameter crossover-based LS (XLS) algorithms (Lozano et al., 2004; Noman and Iba, 2005, 2008). A different approach to face up to this problem involves the approach presented in this paper: to adequately couple intense continuous LS searchers in MACOs.

In Section 3.1, we detail some works on adaptive MACOs with multiple continuous LS operators. In Section 3.2, we describe different XLS algorithms. Finally, in Section 3.3, we provide a brief introduction to intense continuous LS algorithms, paying special attention to the modern continuous LS method called covariance matrix adaptation evolution strategy.

3.1 Meta-Lamarckian Learning in MACOs

Ong and Keane (2004) proposed meta-Lamarckian learning in MAs that adaptively chooses among multiple continuous LS algorithms during the MACO search. The basic idea is to use a pool of continuous LS algorithms and, as the search progresses, the effectiveness of each continuous LS algorithm in dealing with the problem is learned. Then, the continuous LS algorithms with higher fitness improvement measures are rewarded with greater chances of being chosen for subsequent chromosome optimisations.

These authors presented in their work two adaptive strategies, MA-S1 and MA-S2, which utilise the pool formed by a bit climbing algorithm, the complex method of M.J. Box, the Davies, Swann, and Campey search with Gram-Schmidt orthogonalisation, the Hooke and Jeeves direct search, the Fletcher's method, the repeated Lagrangian interpolation, the Simplex method, and two different implementations of the Powell's direct search method. An empirical study showed that the strategies presented are effective in producing search performances that are close to the best traditional MACOs with a continuous LS algorithm chosen to suit the problem in hand.

Ong et al. (2006) extended their investigations on adaptive MACOs with multiple continuous LS operators. In particular, they presented an empirical study of a set of representative models classified according to their type-level adaptations. Numerical results obtained on a range of commonly used benchmark functions of diverse properties indicate that, in general, all the adaptive MACO were capable of selecting a continuous LS method that matches the problem appropriately throughout the search, thus producing search performances that are competitive or superior to the canonical MAs on the benchmark problems.

3.2 Crossover-Based Continuous LS Algorithms

The crossover operator is a recombination operator that produces elements around the parents. For that reason, it may be considered to be a move operator for an LS strategy (Lozano et al., 2004). This is particularly attractive for real coding because there are some real-parameter crossover operators that have a self-adaptive nature in that they can generate offspring adaptively according to the distribution of parents without any adaptive parameter (Beyer and Deb, 2001; Kita, 2001). The aim is to exploit this self-adaptive capacity inside the XLS itself, turning it into a self-adaptive LS algorithm (without any additional adaptive parameter) that may offer an efficient local tuning on the solutions.

One of the first XLS approaches based on self-adaptive real-parameter crossover operators was proposed by Lozano et al. (2004). They presented an XLS model conceived as a micro selecto-recombinative real-coded EA that employs the minimal population size necessary to allow the crossover to be applicable, that is, a pair of chromosomes. The XLS repeatedly performs a real-parameter crossover operator on the pair until some number of offspring, n_{off}, is reached. Then, the best offspring is selected and it replaces the worst parent only if it is better. The process iterates n_{it} times and returns the two final current parents. An experimental study showed that the self-adaptive behaviour of XLS works adequately on many cases (nevertheless some difficulties

```
1. Let us define $C_b$ to be the best chromosome in the population.
2. Build the set $S_p$ composed of $C_b$ and $n_p - 1$ random individuals from the
   population.
3. Apply the simplex crossover considering the elements in $S_p$ as parents (using
   $C_{xls}$ as the returned offspring).
4. If $C_{xls}$ is better than $C_b$ then
5.     $C_b = C_{xls}$.
6. else
7.     return $C_b$.
```

Figure 1: Pseudocode algorithm for the XLS of Noman and Iba (2008).

appeared on some of the more complex problems). In addition, this XLS achieved an acceptable robustness with $n_{\text{off}} = 3$ and $n_{\text{it}} = 3$, which represents a low LS intensity value ($n_{\text{off}} \times n_{\text{it}} = 9$; this result indicates that the XLS model may really be considered as a quick continuous LS method). In a later work, Noman and Iba (2005) proposed a MACO instance that applies, in each generation, an XLS in the neighbourhood of the best solution found by differential evolution (Storn and Price, 1997). The authors claimed that their XLS scheme increases the convergence velocity of differential evolution for high-dimensional optimisation of well-known benchmark functions.

The two aforementioned examples of XLS are fixed length XLS instances; they generate a predetermined number of offspring to search the neighbourhood of the parent individuals (they require looking for a good search length for the XLS operation). Noman and Iba (2008) present an *adaptive length* XLS model that adaptively determines the length of the search by taking feedback from the search, using a simple hill-climbing heuristic. The XLS attempts to refine the best individual of the current generation by applying the simplex real-parameter multi-parent crossover operator (Tsutsui et al., 1999), which is very suitable for neighbourhood search, as suggested by Noman and Iba (2008). The pseudocode algorithm for this XLS instance is depicted in Figure 1, where n_p is the total number of individuals that take part in the crossover operation. These authors suggested that the new XLS approach makes best use of the function evaluations and thereby identifies the optimum at a higher velocity compared to the earlier XLS models. In addition, the authors carried out a performance comparison of differential evolution, integrating the XLS with some other MAs selected from the literature, which include meta-Lamarckian MACOs (Section 3.1). They concluded that the overall performance of their MACO was superior to or at least competitive with the other MACOs.

3.3 Intense Continuous LS Algorithms

Intense continuous LS algorithms may need high LS intensity values to tune their associated strategy parameters to values that allow them to be profitable in the particular search zone being refined. A clear example of intense continuous LS algorithm is the classic Solis and Wets' algorithm (Solis and Wets, 1981), which 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. Note that ρ is the strategy parameter of this continuous LS operator.

The integration of intense continuous LS algorithms into MACOs arises as a particularly attractive research area because, nowadays, there are advanced intense continuous LS algorithms that stand out as formidable local searchers. The covariance matrix adaptation evolution strategy (CMA-ES; Hansen and Ostermeier, 2001; Hansen et al., 2003) is one of them. CMA-ES was originally introduced to improve the LS performance of evolution strategies. Even though CMA-ES even reveals competitive global search performances (Hansen and Kern, 2004), it has exhibited effective abilities for the local tuning of solutions; in fact, it was used as a continuous LS algorithm of an instance of a multi-start LS metaheuristic, which was called L-CMA-ES (Auger and Hansen, 2005b). At the 2005 Congress of Evolutionary Computation, L-CMA-ES was one of the winners of the real-parameter optimisation competition (Suganthan et al., 2005; Hansen, 2005). Thus, investigating the behaviour of CMA-ES as an LS component for MACOs deserves attention.

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, that is, 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:

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

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 Hansen and Ostermeier (2001) and Hansen and Kern (2004). The default strategy parameters are given in Hansen and Kern (2004). Only the initial m and σ parameters have to be set depending on the problem.

Hansen and Ostermeier (2001) interpret any evolution strategy that uses intermediate recombination as an LS strategy. CMA-ES employs intermediate recombination to create a single parent based on the average position of the current population. The next generation of offspring is based on a mutation distribution that surrounds this single parent. Once the initial mutation distribution has decreased, this variation of the traditional evolution strategy will behave much like an LS algorithm. Thus, since CMA-ES is extremely good at detecting and exploiting local structure, it turns out to be a particularly reliable and highly competitive EA for local optimisation (Auger and Hansen, 2005b). In fact, in order to obtain an *advanced* continuous LS algorithm, the LS characteristics of CMA-ES may be stressed by tuning some of its strategy parameters. For example, Auger and Hansen (2005b) recommended using a 100 times smaller initial step-size than is recommended as the default and sticking to the default population size (between 10 and 15 for the search space dimensions). We have explored a different approach to enhance the LS abilities of this algorithm, as described in Section 4.4.

Finally, we highlight that this evolution strategy model may be categorised, specifically, as an intense continuous LS algorithm, because, as noted by Auger and colleagues (2004):

```
1. Select two parents from the population.
2. Create an offspring using crossover and mutation.
3. Evaluate the offspring with the fitness function.
4. Select an individual in the population that may be replaced by the offspring.
5. Decide if this individual will be replaced.
```

Figure 2: Pseudocode algorithm for the steady state GA model.

CMA-ES may require a substantial number of time steps for the adaptation of the covariance matrix.

4 MACOs Based on LS Chains

Due to the potential of the intense continuous LS algorithms, it becomes interesting to build MACO models with them. These MACOs should be specifically designed to accomplish two essential aims:

- The intense continuous LS algorithm can be provided with sufficient LS intensity to make their correct operation possible.
- The MACO should ensure that a profitable synergy between the continuous LS algorithm and the EA is possible.

In this section, we propose a MACO approach conceived to attain these two objectives. A steady state MA model employs the concept of LS chain to adjust the LS intensity assigned to the intense continuous LS method. In particular, our 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. It is worth noting that similar ideas have been exploited to build adaptive MAs, as mentioned in Section 2.2.

In Section 4.1, we introduce the foundations of steady state MAs. In Section 4.2, we present the concept of the LS chain. In Section 4.3, we propose a MACO approach that handles LS chains with the objective to make good use of intense continuous LS methods as LS operators. Finally, in Section 4.4, we present an instance of our MACO model that uses CMA-ES as a continuous LS operator.

4.1 Steady-State MAs

In steady state genetic algorithms (GAs; Sywerda, 1989; Whitley, 1989) 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. The basic algorithm step of the steady state GA is shown in Figure 2.

These steps are repeated until a termination condition is achieved. In Step 4, one can choose the replacement strategy (e.g., replacement of the worst, the oldest, or a randomly chosen individual). In step 5, one can choose the replacement condition (e.g.,

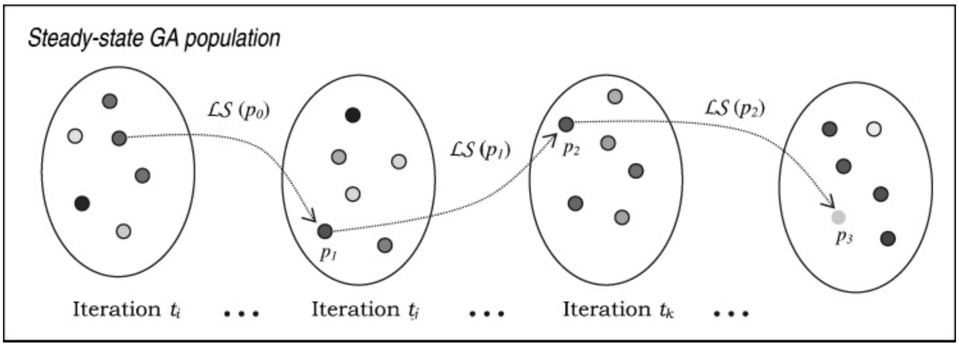
Steady-state GA population
$\mathcal{L}S\ (p_0)$
$\mathcal{L}S\ (p_1)$
$\mathcal{L}S\ (p_2)$
p_1
p_2
p_3
Iteration t_i • • • Iteration t_j • • • Iteration t_k • • •

Figure 3: Example of LS chain. p_{i+1} is the final parameter value reached by the LS algorithm when it started with a value of p_i. p_0 is the default value for the strategy parameter.

either replacement if the new individual is better or unconditional replacement). A widely used combination is to replace the worst individual only if the new individual is better. We will call this strategy the standard replacement strategy. In Goldberg and Deb (1991), it was suggested that the deletion of the worst individuals induced a high selective pressure, even when the parents were selected randomly.

Although steady state GAs are less common than generational GAs, Land (1998) 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. Steady state MAs integrate global and local search more tightly than generational MAs (Land, 1998). This interleaving of the global and local search phases allows the two to influence each other, for example, the steady state GA chooses good starting points, and LS provides an accurate representation of that region of the domain. Contrarily, generational MAs proceed in alternating stages of global and local search. First, the generational GA produces a new population, and then LS is performed. The specific state of LS is generally not kept from one generation to the next, though LS results do influence the selection of individuals.

4.2 LS Chains

In steady state MAs, individuals resulting from the LS invocation may reside in the population for a long time. This circumstance allows these individuals to become starting points of subsequent LS invocations. At this point, we 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 the 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, which is to say, forming an LS chain. Figure 3 shows an example of an LS chain formed by an LS algorithm with only one associated strategy parameter, p.

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

1. 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 the LS intensity stretch (I_{str}). In this way, an 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 $n_{app} \times I_{str}$ fitness function evaluations.

2. After the LS operation, the parameters that define the current state of LS processing are stored along with the final individual reached (in the steady state GA population). When this individual is later 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 as our LS algorithm, the stored strategy parameter may be the current value of the ρ parameter. For the more elaborate CMA-ES (Section 3.3), the state of the LS operation may be defined by the covariance matrix (C), the mean of the distribution ($\vec{m}$), the size (σ), and some additional variables used to guide the adaptation of these parameters.

In this work, we argue that a promising approach to adapt the LS intensity assigned to intense continuous LS algorithms involves the management of LS chains. A MACO may allow LS chains to grow throughout the evolution depending on the quality of the search regions being visited, with the aim of acting more intensely in the most promising areas. As the MACO search progresses, LS chains will be created and extended, remaining in a latent state until one of their links (chromosomes in the current population that belong to chains) is never selected for refinement, or simply disappears from the population. In this fashion, the real LS intensity assigned to the continuous LS algorithm may be adaptively determined throughout the run and depends on two crucial choices:

1. The way the solutions are selected to apply the LS operator to them.

2. The replacement scheme used by the steady state GA.

The designer of the steady state GA is responsible for the second election, whereas the first one should be undertaken during the design of the MACO scheme. In the next section, we take special care to explore this important choice, in order to build the MACO approach that handles LS chains.

Finally, we should point out that the concept of sniffs introduced by Land (1998) bears a slight resemblance to the LS chain concept. Individual solutions are subject to a limited amount of local search (i.e., a sniff). Moreover, those solutions that were in the proximity of a promising basin of attraction received (at a later stage) an extended CPU budget. With that budget, further iterations of local search were performed.

4.3 A MACO Model That Handles LS Chains

In this section, we propose a MACO model (see Figure 4) 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 (Section 2.1) is always kept. With this policy, we easily stabilise this ratio, which has a strong

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

Figure 4: Pseudocode algorithm for the proposed MACO model.

influence on the final MACO behaviour. Without this strategy, the application of intense continuous LS algorithms may induce the MACO to prefer super exploitation.

3. It favours the enlargement of those LS chains that show 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).

The proposed 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:

$$n_{\mathrm{frec}} = I_{\mathrm{str}} \frac{1 - r_{L/G}}{r_{L/G}}. \tag{1}$$

where I_{str} is the LS intensity stretch (Section 4.2). We recall that $r_{L/G}$ is defined as the percentage of evaluations spent doing LS from the total assigned to the algorithm's run (Section 2.1).

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

1. Build the set of individuals from the steady state GA population, S_{LS} that fulfils:

   a. They have never been optimised by the intense continuous LS algorithm, or
   b. They previously underwent LS, obtaining a fitness function improvement greater than $\delta^{\mathrm{min}}_{\mathrm{LS}}$ (a parameter of our algorithm).

2. If $|S_{\mathrm{LS}}| \neq 0$, 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 $\delta_{\mathrm{LS}}^{\mathrm{min}}$ threshold. The last condition is very important in order to avoid the overexploitation of search zones where the LS method may not make substantial progress any more. We should point out that other adaptive MA models employ other measures of improvement obtained by LS methods. In particular, in the meta-Lamarckian MA (Section 3.1), the continuous LS methods with higher fitness improvement measures are rewarded with a greater chance of being chosen for subsequent chromosome optimisations.

4.4 MA-LSCh-CMA

In this section, we build an instance of the proposed MACO model (Figure 4), which applies CMA-ES (Section 3.3) as an intense continuous LS algorithm. It will be called MA-LSCh-CMA. The design decisions for MA-LSCh-CMA were made with the aim of allowing it to be very competitive with the state of the art on EAs for continuous optimisation. With this aspiration, along with the decision of applying CMA-ES following the approach proposed in this paper, we have carefully chosen the configuration of the steady state GA as well, which should attempt to induce *reliability* in the search process by ensuring that different promising search zones are the focus of the LS procedure throughout the run.

Next, we list the main features of MA-LSCh-CMA:

4.4.1 Steady-State GA

MA-LSCh-CMA is a real-coded steady state GA (Herrera et al., 1998) specifically designed to promote high population diversity levels by means of the combination of the BLX-α crossover operator with a high value for its associated parameter ($\alpha = 0.5$) and the negative assortative mating strategy (Fernandes and Rosa, 2001). Diversity is favoured as well by means of the BGA mutation operator (Mühlenbein and Schlierkamp-Voosen, 1993). In the MA literature, keeping population diversity while using LS together with an EA is always an issue to be addressed, either implicitly or explicitly (Krasnogor, 2002; Lozano et al., 2004; Tang et al., 2007). In particular, the application of these three diversification techniques has proved effective for allowing EAs to suitably collaborate with continuous LS methods in MACOs (Lozano et al., 2004). Next, we describe the features of these components.

4.4.2 BLX-α

Let us assume that $C_1 = (c_1^1 \ldots c_n^1)$ and $C_2 = (c_1^2 \ldots c_n^2)$ $(c_i^1, c_i^2 \in [a_i, b_i] \subset \Re, i = 1 \ldots n)$ are two real-coded chromosomes that have been selected to apply the crossover operator to them. BLX-α (Eshelman and Schaffer, 1993) generates an offspring, $Z = (z_1 \ldots z_n)$, where z_i is a randomly (uniformly) chosen number from the interval $[min_i - I \times \alpha, max_i + I \times \alpha]$, where $max_i = max\{c_i^1, c_i^2\}$, $min_i = min\{c_i^1, c_i^2\}$, and $I = max_i - min_i$. Figure 5 shows the operation of BLX-α.

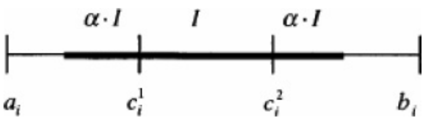

Figure 5: BLX-α.

Nomura and Shimohara (2001) provide a formalisation of this operator to analyse the relationship between the chromosome probability density functions before and after its application, assuming an infinite population. They state that BLX-α spreads the distribution of the chromosomes when $\alpha > \frac{\sqrt{3}-1}{2}$ or otherwise reduces the distribution. This property was verified through simulations. In particular, the authors observed that BLX-0.0 makes the variances of the distribution of the chromosomes decrease, reducing the distribution, whereas BLX-0.5 makes the variances of the distribution increase, spreading the distribution.

4.4.3 Negative Assortative Mating

The mating selection mechanism determines the way the chromosomes are mated by applying the crossover to them (Step 1 in Figure 2). Mates can be selected so as to favour population diversity (Fernandes and Rosa, 2001). A way to do this is the negative assortative mating mechanism. Assortative mating is the natural occurrence of mating between individuals of similar genotype more or less often than expected by chance. Mating between individuals with similar genotype more often is called positive assortative mating and less often is called negative assortative mating.

Fernandes and Rosa (2001) assume these ideas in order to implement a parent selection mechanism for the crossover operator. A first parent is selected by the roulette wheel method and n_{ass} chromosomes are selected with the same method (in our experiments all the parents are selected at random). Then, the similarity between each of these chromosomes and the first parent is computed (similarity between two real-coded chromosomes is defined as the Euclidean distance between them). If assortative mating is negative, then the one with less similarity is chosen. If it is positive, the genome that is most similar to the first parent is chosen to be the second parent. Clearly, the negative assortative mating mechanism increases genetic diversity in the population by mating dissimilar genomes with higher probability.

4.4.4 BGA Mutation Operator

In MACOs, the operator responsible for the local tuning of the solutions is the continuous LS operator. Hence, we require a mutation operator providing acceptable levels of diversity continuously. One of the mutation operators that behaves in this manner is the BGA mutation operator (Mühlenbein and Schlierkamp-Voosen, 1993). Let us suppose $C = (c_1, \ldots, c_i, \ldots, c_n)$ is a chromosome and $c_i \in [a_i, b_i]$ is a gene to be mutated. The gene, c_i', resulting from the application of this operator is:

$$c_i' = c_i \pm rang_i \cdot \sum_{k=0}^{15} \alpha_k 2^{-k},$$

where $rang_i$ defines the mutation range and it is normally set to $0.1 \times (b_i - a_i)$. The $+$ or $-$ sign is chosen with a probability of 0.5 and $\alpha_i \in \{0, 1\}$ is randomly generated with $p(\alpha_i = 1) = \frac{1}{16}$. Values in the interval $[c_i - rang_i, c_i + rang_i]$ are generated using this operator, with the probability of generating a neighbourhood of c_i being very high. The minimum possible proximity is produced with a precision of $rang_i \times 2^{-15}$.

4.4.5 Replacement Strategy

The steady state GA combines the negative assortative mating (that favours high population diversity levels) with the *standard replacement strategy* (that induces high selective pressure, as mentioned in Section 4.1). In this way, many dissimilar solutions are produced during the run and only the best ones are conserved in the population, allowing diverse and promising solutions to be maintained. Other authors have suggested the filtering of high diversity by means of high selective pressure as a GA strategy to provide effective search. For example, in Shimodaira (1996), an algorithm is proposed employing large mutation rates and population-elitist selection, and in Eshelman (1991), a GA is proposed which combines a disruptive crossover operator with a conservative selection strategy.

4.4.6 CMA-ES as Continuous LS Algorithm

MA-LSCh-CMA follows the MACO approach, presented in Section 4.3, to handle LS chains, with the objective of tuning the intensity of CMA-ES, which is employed as an intense continuous LS algorithm (Section 3.3). The application of CMA-ES for refining an individual, C_i, is carried out following the next guidelines:

- C_i becomes the initial mean of distribution ($\vec{m}$).
- The initial σ value is half 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). Later, 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 Hansen and Kern (2004), are assumed for the remaining strategy parameters.

4.4.7 Parameter Setting

For the experiments, MA-LSCh-CMA applies BLX-α with $\alpha = 0.5$. The population size is 60 individuals and the probability of updating a chromosome by mutation is .125. The n_{ass} parameter associated with the negative assortative mating is set to 3. 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 $\delta_{\text{LS}}^{\text{min}}$ threshold.

5 Experimental Framework

We have carried out different experiments to assess the performance of MA-LSCh-CMA (Section 4.4). In order to do this, in this section, we detail the test functions (Section 5.1) and the experimental setup and statistical methods (Section 5.2) that were used for this experimental study.

5.1 Test Functions

The test suite that we have used for different experiments consists of 20 benchmark functions chosen from the set designed for the Special Session on Real Parameter Optimisation Organised in the 2005 IEEE Congress on Evolutionary Computation (CEC2005; Suganthan et al., 2005). See Suganthan et al. (2005) for the complete description of the functions; furthermore, the link to the source code is included in the reference. The set of test functions is composed of the seven basic multimodal functions (F6–F12); two expanded functions (F13 and F14); and 11 hybrid functions (F15–F25). The seven basic multimodal functions are as follows.

1. Shifted Rosenbrock's function.
2. Griewank's function displaced and rotated without frontiers.
3. Ackley's function displaced and rotated with the global optimum in the frontier.
4. Rastrigin's function displaced.
5. Rastrigin's function displaced and rotated.
6. Weierstrass' function displaced and rotated.
7. Schwefel's problem 2.13.

Note that each of the 11 hybrid functions has been defined through compositions of 10 out of the first 14 functions presented by Suganthan et al. (2005) (different in each case).

All functions have been displaced in order to ensure that their optima can never be found in the centre of the search space. In two functions, in addition, the optima cannot be found within the initialisation range, and the domain of search is not limited (the optimum is out of the range of initialisation).

This set of functions may be divided into two subgroups, according to the suggestion given by Hansen (2005) about their degrees of difficulty. The first group is composed of the functions from F6 to F14, in which the existence of an algorithm that achieved the optimum is known. The second group contains the remaining functions, from the function F15 to F25. In these functions, the optimum has been never achieved and they may be categorised as very difficult functions.

We have not considered the unimodal functions (F1–F5) from the CEC2005 test suite, because we are particularly interested in comparing the performance of the algorithms when they tackle complicated test functions. This allows us to analyse their behaviour as global optimisers and determine the effectiveness of the paradigm they follow to face the conflict between accuracy and reliability.

5.2 Experimental Setup and Statistical Analysis

The experiments have been done following the instructions indicated in the document associated with the competition. The main characteristics are:

- Each algorithm is run 25 times for each test function, and the average of error of the best individual of the population is computed. The *function error* value for a solution x is defined as $[f(x) - f(x^*)]$, where x^* is the global optimum of the function.

- The study has been made with dimensions $D = 10$, $D = 30$, and $D = 50$.
- The maximum number of fitness evaluations that we allowed for each algorithm to minimise the error was $10{,}000 \times D$, where D is the dimension of the problem.
- Each run stops either when the error obtained is less than 10^{-8}, or when the maximal number of evaluations is achieved.

We have carried out the experimental study of MA-LSCh-CMA following these guidelines in order to make possible its comparison with the results of all the other algorithms involved in the competition (their results are available in the Proceedings of the Congress).

Nonparametric tests can be used for comparing the results of different search algorithms (García et al., 2009). Given that the nonparametric tests do not require explicit conditions to be conducted, it is recommended that the sample of results would be obtained following the same criterion, which is, to compute the same aggregation (average, mode, etc.) over the same number of runs for each algorithm and problem.

We have chosen this type of statistical test because in our experiments we carry out many comparisons with algorithms presented in the CEC2005 Special Session on Real Parameter Optimisation (Suganthan et al., 2005), and precisely, the average results achieved by them (for the functions listed in the previous section) are directly available, since they were published in the Proceedings of the Congress.

In particular, we have considered two alternative methods based on nonparametric tests to analyse the experimental results:

1. The first method is the application of the Iman and Davenport test and the Holm method as a post hoc procedure. The first test may be used to see whether there are significant statistical differences among the algorithms on a certain group of test algorithms. If differences are detected, then Holm's test is employed to compare the best algorithm (control algorithm) against the remaining ones.
2. The second method is the utilisation of the Wilcoxon matched-pairs signed-ranks test. With this test, the results of two algorithms may be directly compared.

We explain these statistical tests with detail in Appendix A.

6 Analysis of the Results

In this section, we analyse the results obtained from different experimental studies carried out with MA-LSCh-CMA. In particular, our aims are: (1) to investigate its sensitivity to certain parameter choices (Section 6.1); (2) to understand the way this algorithm behaves (Section 6.2); and (3) to ascertain whether the innovative design is suitable to outperform other MACOs (Section 6.3) and EAs for continuous optimisation (Sections 6.4 and 6.5).

6.1 Influence of the LS Intensity Stretch

In our first empirical study, we investigate the influence of I_{str} on the performance of MA-LSCh-CMA. In particular, we analyse the behaviour of this algorithm when different values for this parameter are considered ($I_{\text{str}} = 100$, 500, and 1000).

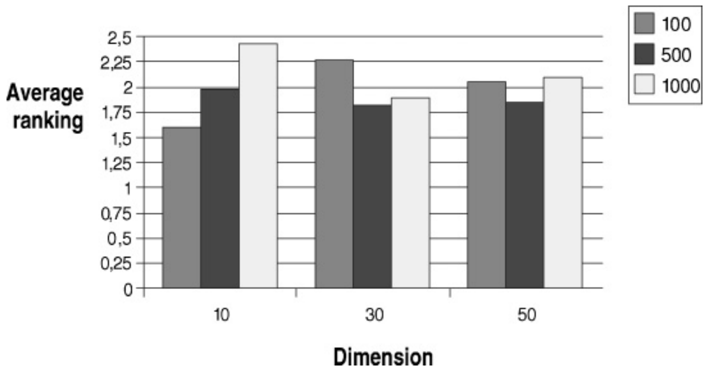

Figure 6: Rankings obtained by MA-LSCh-CMA instances with different I_{str} values.

Table 1: Results of the Iman-Davenport test.

D	Iman-Davenport value	Critical value	Significant differences?
10	3.90	2.77	Yes
30	1.17	2.77	No
50	0.33	2.77	No

Figure 6 shows the average rankings (computed by the Friedman's test) obtained by the MA-LSCh-CMA instances with different I_{str} values on the test functions with dimensions $D = 10$, $D = 30$, and $D = 50$. Each column represents the average ranking obtained by an algorithm; that is, if a certain algorithm achieves rankings 1, 3, 1, 4 and 2, on five test functions, the average ranking is $\frac{1+3+1+4+2}{5} = \frac{11}{5}$. The height of each column is proportional to the ranking. Therefore, the lower a column is, the better its associated algorithm is. In order to study these results, we have introduced Table 1, which outlines the *Iman-Davenport* statistics (see Appendix A) and their critical values at the 5% level when comparing these average rankings.

A visual inspection of Table 1 allows one to conclude that:

- For $D = 30$ and $D = 50$, there are not significant differences among the MA-LSCh-CMA instances (the statistical values are not greater than the critical values). Thus, for these two cases, MA-LSCh-CMA is rather insensitive to the LS intensity stretch. These results indicate that, independent of the value for this parameter, the proposed MACO model may relapse into the refinement of particular search zones (by means of LS chaining) with the aim of achieving a real LS intensity being the most profitable result possible for those areas.

- For $D = 10$, we observe the existence of significant differences among the rankings (the statistical value is greater than the critical value, 2.77). With regard to this result, we compare the best ranked instance, $I_{\text{str}} = 100$ (see Figure 6) with the other instances, by means of Holm's test (a post hoc statistical analysis), where the instance with $I_{\text{str}} = 100$ is the control algorithm. Table 2 contains all the computations associated with Holm's procedure (z, p value, and α/i) with $p = .05$. The last column indicates whether the corresponding algorithm performs statistically

Table 2: Comparison, using Holm's test, of the instance with $I_{str} = 100$ with the remaining ones ($D = 10$).

I_{str}	z	p value	α/i	Significant differences?
1000	2.609	.0090	0.0250	Yes
500	1.185	.2356	0.0500	No

equivalent to the control algorithm (i.e., the equality hypothesis is accepted) or the control algorithm performs significantly better than the corresponding algorithm (i.e., the equality hypothesis is rejected).

In Table 2, we see that only $I_{str} = 1000$ causes performance detrimental with regard to $I_{str} = 100$. With this high intensity stretch value, the process for creating LS chains may consume, frequently, too many evaluations. Given the reduced number of evaluations adjudicated to MA-LSCh-CMA for $D = 10$ (100,000), this circumstance may imply that MA-LSCh-CMA does not have enough computation time to adequately refine diverse search zones.

We extract an important conclusion from this study: MA-LSCh-CMA exhibits a low sensitivity degree to the value selected for I_{str}. We have chosen a particular value for I_{str}, in order to allow the incoming study of our proposal and the comparison with other EA models to be easily understandable. Observing Figure 6, we note that $I_{str} = 500$ is the best choice, because it obtains the lower rankings for $D = 30$ and $D = 50$, and in addition, this setting is able to achieve similar results as the best I_{str} value for $D = 10$ (see Table 2). (See Appendix B, Table 11, for the results of our algorithm with this setting for all test functions).

6.2 Studying the Behaviour of the Proposed MACO Model

In this section, we cover two types of experiments we performed in order to investigate the behaviour of the proposed MACO model.

- In our first experiment, we attempted to show the superiority of MA-LSCh-CMA against an MACO that applies CMA-ES as LS method, following a standard hybridisation strategy (Section 6.2.1). Our aim was to determine whether the proposed MACO may really profit from the use of the intense continuous LS methods as LS operators for MACOs.
- In our second experiment, we compared MA-LSCh-CMA with another kind of continuous optimisation algorithm proposed in the literature that invokes CMA-ES for refining single solutions during the search process (Section 6.2.2). We are interested in investigating the way our MACO approach may take more advantage of the potentiality of this algorithm as the local optimiser.
- Finally, we investigate whether MA-LSCh-CMA adaptively tunes the LS intensity for CMA-ES throughout the evolution depending on the particular problem to be solved, allowing a robust operation to be achieved (Section 6.2.3).

6.2.1 Comparison with a Standard MACO

This section presents a performance comparison between the proposed algorithm and a standard MACO, which will be denoted as S-MACO. The basic difference between

Table 3: Results of the Iman-Davenport test.

D	Iman-Davenport value	Critical value	Significant differences?
10	1.37	2.77	No
30	13.03	2.77	Yes
50	8.54	2.77	Yes

Table 4: Comparison (Holm's test) of S-MACO with $I_{\text{LS}} = 1000$ with the remaining values when $D = 30$ and $D = 50$.

D	I_{LS}	z	p value	α/i	Significant differences?
30	100	4.032	5.53E005	0.025	Yes
30	500	1.897	0.05778	0.050	No
50	100	3.478	0.0050	0.025	Yes
50	500	2.214	0.02685	0.050	Yes

S-MACO and MA-LSCh-CMA is that the former applies the LS method following the MACO approach proposed by Hart (1994): every new chromosome generated by BLX-α and BGA mutation undergoes the CMA-ES LS procedure with a probability p_{LS}. In each invocation, CMA-ES starts from default values for its strategy parameters and consumes I_{LS} evaluations. Three different values for I_{LS} were investigated: 100, 500, and 1000. For each I_{LS} value, a suitable p_{LS} value was calculated with the aim of fitting the $\frac{L}{G}$ ratio kept by S-MACO to 0.5, such as MA-LSCh-CMA does (in particular, $p_{\text{LS}} = .01$, .002, and .001, respectively).

S-MACO represents a simple way to hybridize CMA-ES with the steady state GA employed in MA-LSCh-CMA. Using the same GA scheme and changing the type of hybridization mechanism, we may determine whether the approach to manage LS chains is really able to provide a better operation of CMA-ES as an intense continuous LS operator with regard to a standard MACO that does not handle LS chains.

Firstly, we have attempted to find the value for I_{LS} that performs the best for the different dimensions. For $D = 10$, we see in Table 3 that the Iman-Davenport test did not detect significant differences among the results obtained using the three I_{LS} values. For the case of $D = 30$ and $D = 50$, this test found significant differences, and then, we compared the best ranked value for these two dimensions, $I_{\text{LS}} = 1000$, with the other two values calculated by means of the Holm test (Table 4). This test indicates that, in general, $I_{\text{LS}} = 1000$ outperforms the results of the other values. Then, we conclude that $I_{\text{LS}} = 1000$ becomes a convenient value for S-MACO. (See Appendix B, Table 12, for the results of this algorithm for all test functions.)

Now we can compare S-MACO with $I_{\text{LS}} = 1000$ and MA-LSCh-CMA ($I_{\text{str}} = 500$) using Wilcoxon's test. Table 5 summarizes the results of this procedure, where the values of $R+$ (associated to MA-LSCh-CMA) and $R-$ of the test are specified (the highest ones, which correspond with the best results, are noted in boldface type), together with the critical values.

We note that MA-LSCh-CMA obtains better results than S-MACO for all dimensions (the $R-$ values are lower than the $R+$ ones). But in addition, the statistical test indicates that these improvements are statistically significant for $D = 30$ and $D = 50$ (because these $R-$ values are lower than the critical values).

Table 5: S-MACO versus MA-LSCh-CMA using Wilcoxon's test (p value = .05); best results are in boldface type.

D	$R+$ (MA-LSCh-CMA)	$R-$ (S-MACO)	Critical value	Significant differences?
10	**131.5**	78.5	52	No
30	**193**	17	52	Yes
50	**164.5**	45.5	52	Yes

Table 6: L-CMA-ES versus MA-LSCh-CMA (Wilcoxon's test with p value = .05).

D	$R+$ (MA-LSCh-CMA)	$R-$ (L-CMA-ES)	Critical value	Significant differences?
10	**109.5**	100.5	52	No
30	**165**	45	52	Yes
50	**165**	45	52	Yes

These initial experiments suggest that our specific MACO design may really enhance the operation of CMA-ES as an LS operator for these algorithms. Thus, it is a promising candidate for exploiting the abilities of an intense continuous LS method as an LS operator for MACOs.

6.2.2 Comparison with a Restart Local Search Algorithm

Being fascinated by the potential of CMA-ES as local optimiser, Auger and Hansen (2005b) proposed a restart LS algorithm (referred to as L-CMA-ES in the following) that performs a version of CMA-ES with small population size and small initial step size to stress its LS characteristics. Independent restarts are conducted until the target function value is reached or the maximum number of function evaluations is exceeded.

In this section, we carry out the comparison of MA-LSCh-CMA with L-CMA-ES, which seems natural, because both algorithms invoke CMA-ES instances that specifically emphasise the local refinement abilities of this algorithm. Table 6 has the results of the comparison of these two algorithms, by means of the Wilcoxon test.

MA-LSCh-CMA exhibits overall better performance than L-CMA-ES for $D = 30$ and $D = 50$ (the $R-$ values are lower than both the $R+$ ones and the critical values). For $D = 10$, there are no differences between them. This unbiased comparison with regard to the LS operator has allowed us to know that our memetic framework shows promise as a global search approach based on CMA-ES. Particularly significant improvements are obtained at higher dimensionality, where the work of the proposed hybridisation method outperforms the pure restart local search strategy.

6.2.3 Study of the Adaptation of the LS Intensity

In this section, we are interested in determining whether MA-LSCh-CMA adjusts the length of the LS chains (i.e., the LS intensity levels assigned to CMA-ES) according to the particularities of the problem to be solved, allowing performance improvement to be achieved. In order to do this, we have introduced Figures 7, 8, and 9, which display the percentage in which LS chains with different lengths (number of links) were found throughout the first run of our algorithm on three well-known test functions, the Generalized Rosenbrock's function (F6), Griewangk's function (F7), and the Generalized

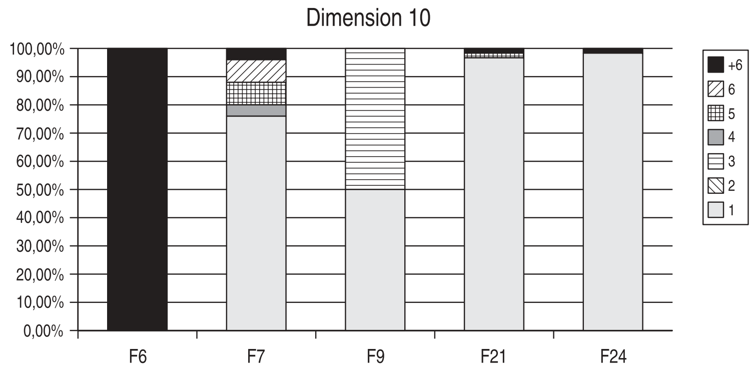

Figure 7: Percentages of LS chains with different lengths ($D = 10$).

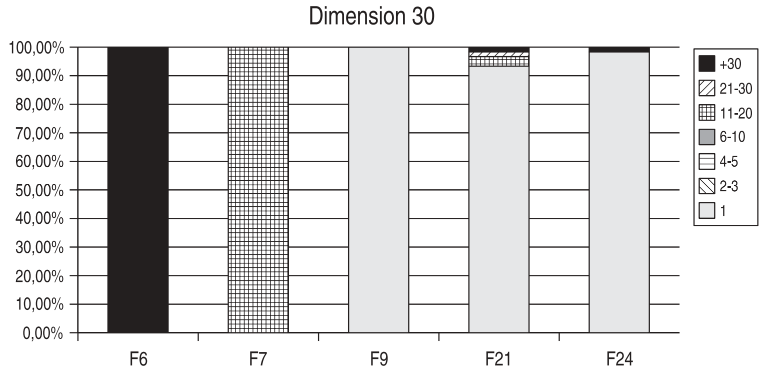

Figure 8: Percentages of LS chains with different lengths ($D = 30$).

Rastringin's function (F9), and two additional functions, F21 and F24, which belong to the group of hybrid functions, which are extremely complex, as was mentioned in Section 5.1.

We note the following two observations.

1. Traditionally, Rosenbrock's function (F6) was claimed to be unimodal. However, Deb and colleagues (2002) noticed that for $D > 3$, this function has more than one minima (and for 20 variables, they identified three minima). The global optimum of this function is located in a steep parabolic valley with a flat bottom. This feature will probably cause slow progress in any optimisation algorithm, because it must continually change search direction to reach the optimum. We observe that for $D = 10$ and $D = 30$, long LS chains were formed during the evolution of MA-LSCh-CMA when tackling F6 (Figures 7 and 8). By assigning high LS intensity to CMA-ES, our algorithm qualifies it to adapt its search process to the complicated access to the global optimum. This option becomes suitable, as well, for this problem with few local optima, because it allows high accuracy to be achieved with an inconsiderable risk of losing reliability.

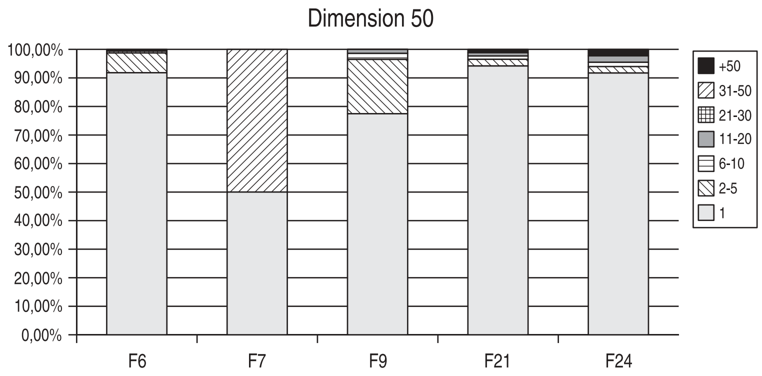

Figure 9: Percentages of LS chains with different lengths ($D = 50$).

2. Griewangk's function (F7) and Rastringin's function (F9) are multimodal functions with many local optima. We may see that when $D = 10$ (Figure 7), the higher percentage for these two functions corresponds to one-length LS chains. This means that CMA-ES often refined individuals that did not belong to previously created chains, that is, it operated with minimal LS intensity. This strategy may be fruitful for high multimodal and complex functions, because, in this fashion, the LS procedure exploits diverse search zones being sampled by the steady state GA, without taking a long time over a particular zone.

The complexity of Rastringin's function increases for $D = 30$. Our algorithm reacted to this circumstance by forming uniquely mono-length LS chains (Figure 8). Interestingly, it has favoured, as well, the proliferation of these LS chains when dealing with the complex hybrid functions F21 and F24, for every dimensionality and, in general, with all functions for $D = 50$ (most functions turn into very hard problems with this dimensionality). Nevertheless, the behaviour of MA-LSCh-CMA on Griewangk's function with $D = 30$ (Figure 8) is clearly different; it managed LS chains from 11 to 20 of length. This function becomes easier when the dimension exceeds 15 (Whitley et al., 1996; Yang and Kao, 2000). Our algorithm was able to adapt its conduct to this situation by inducing the formation of longer LS chains (high LS intensification is well-suited for noncomplex problems).

To sum up, this study shows that the adaptation ability of MA-LSCh-CMA allows the LS intensity for CMA-ES to be adjusted according to the particularities of the search space, allowing an adequate operation to be attained for problems with different difficulties.

6.3 Comparison with State of the Art MACOs

In a recent publication, Noman and Iba (2008) present a MACO model, called DEahcSPX, which combines differential evolution with a quick continuous LS method that consists of an XLS algorithm based on the simplex real-parameter multi-parent crossover operator (see Section 3.2). The authors compared DEahcSPX with other MACO instances reported in the literature, which include meta-Lamarckian MACOs (see Section 3.1) and differential evolution with the XLS strategy proposed by Lozano et al. (2004).

Table 7: DEahcSPX versus MA-LSCh-CMA (Wilcoxon's test with p value = .05). Best values are shown in boldface type.

D	$R+$ (MA-LSCh-CMA)	$R-$ (DEahcSPX)	Critical value	Significant differences?
10	**135**	75	52	No
30	**169.5**	40.5	52	Yes
50	**176.5**	33.5	52	Yes

They found that their proposal was superior to, or at least comparable to, these other instances. Thus, we assume that DEahcSPX is, up to now, the most outstanding representative of the state of the art MACOs.

In this section, we undertake comparative analysis among DEahcSPX and MA-LSCh-CMA using Wilcoxon's test. Table 7 contains the results of this statistical test. (Table 13, in Appendix B, has the results of this algorithm on the test suite.)

The results of MA-LSCh-CMA show higher quality than the ones of DEahcSPX on the three dimensions studied (all the $R-$ values are lower than the $R+$ ones). In addition, the superiority is statistically significant for $D = 30$ and $D = 50$. Thus, as we expected, the design of a specific MACO model making the application of CMA-ES profitable as an intense continuous LS operator may really enhance the performance of this type of hybrid EA. In fact, our proposal has turned out to be very competitive with state of the art MACOs.

6.4 Comparison with the Winner of the CEC2005 Competition: G-CMA-ES

The winner of the Real-Parameter Optimization competition was G-CMA-ES (as recognised by Langdon and Poli, 2008), an algorithm initially proposed by Hansen and Kern (2004) that was investigated in Auger and Hansen (2005a) for optimising the test suite of the CEC2005 competition. In fact, nowadays, it is recognised as a formidable algorithm for continuous optimisation problems (Lunacek et al., 2005; Lunacek and Whitley, 2006; Langdon and Poli, 2008).

G-CMA-ES is a restart CMA-ES variation that detects premature convergence and launches a restart strategy that doubles the population size on each restart. By increasing the population size, the search characteristic becomes more global after each restart, which empowers the operation of the CMA-ES on multimodal functions (Hansen and Kern, 2004).

Next, we attempt to determine the quality of MA-LSCh-CMA as continuous optimiser confronting it with the current best algorithm for continuous optimisation, G-CMA-ES. In order to detect the differences among these algorithms, we have applied Wilcoxon's test. Table 8 contains the results for $p = .05$ and .1.

The results presented in Table 8 reveal the following.

- For $D = 30$ and $D = 50$, our newly proposed MACO outperformed G-CMA-ES (for these two cases, the $R-$ values are lower than the $R+$ values associated with MA-LSCh-CMA).
- Particularly, meaningful advantage is achieved at the highest dimensionality, $D = 50$ (considering $p = .1$).
- G-CMA-ES clearly beats our algorithm only for the lowest dimension, $D = 10$.

Table 8: G-CMA-ES versus MA-LSCh-CMA (Wilcoxon's test with p value $= .05$ and p value $= .1$). Best values are shown in boldface type.

D	$R+$ (MA-LSCh-CMA)	$R-$ (G-CMA-ES)	Critical value ($p = .05/p = .1$)	Significant differences? ($p = .05$)	Significant differences? ($p = .1$)
10	32.5	**177.5**	52/60	Yes	Yes
30	**139**	71	52/60	No	No
50	**154**	56	52/60	No	Yes

We note that MA-LSCh-CMA arises as one of the most prominent algorithms for global optimisation over continuous spaces; especially as dimensionality starts posing real challenges for any continuous function optimisation program. To affront high complexity, MA-LSCh-CMA simultaneously puts into effect two important search processes.

1. The steady state GA induces a scattered search by means of the application of techniques promoting population diversity: BLX-α, negative assortative mating, and BGA mutation (see Section 4.4). This is essential to provide reliability for multimodal and complex problems.

2. The proliferation of long LS chains in the best regions becomes suitable to obtain adequate accuracy levels, but also, the activation of LS chains in alternative regions (supplied by the steady state GA) ensures effective refinement of diverse prospective areas of the search space.

To sum up, the way MA-LSCh-CMA faces the conflict between accuracy and reliability allows this algorithm to be specifically well-suited to deal with complicated search spaces, even outperforming G-CMA-ES in these cases.

6.5 Comparison with the Other CEC2005 Competitors

The main aim of this section is to compare our MACO model with the CEC2005 competitors (L-CMA-ES and G-CMA-ES have not been considered because they were compared in previous sections). For $D = 10$, the results of nine algorithms listed below are available.

- Three GA instances for continuous optimisation: K-PCX (Sinha et al., 2005), SPC-PNX (Ballester et al., 2005), and CoEvo (Posik, 2005).
- One hybrid EA: BLX-GL50 (García-Martínez and Lozano, 2005).
- One MACO instance: BLX-MA (Molina et al., 2005).
- One particle swarm optimiser: DMS-L-PSO (Liang and Suganthan, 2005).
- Two differential evolution algorithms: DE (Ronkkonen et al., 2005) and L-SADE (Qin and Suganthan, 2005).
- One instance of estimation distribution algorithm: EDA (Yuan and Gallagher, 2005).

For $D = 30$, the results of only six out of the previous nine algorithms were reported. Finally, we should point out that none of these algorithms was studied for $D = 50$. The

Table 9: Comparison of MA-LSCh-CMA with CEC2005 competitors for $D = 10$ (Wilcoxon's test with p value = .05). Best results are in boldface type.

Algorithm	$R+$ (MA-LSCh-CMA)	$R-$ (CEC2005)	Critical value	Significant differences?
BLX-GL50	92.5	**117.5**	52	No
BLX-MA	79	**131**	52	No
CoEvo	**157**	53	52	No
DE	**122**	88	52	No
DMS-L-PSO	54.5	**155.5**	52	No
EDA	98	**112**	52	No
K-PCX	**128**	82	52	No
L-SaDE	48.5	**161.5**	52	Yes
SPC-PNX	95	**115**	52	No

Table 10: Comparison of MA-LSCh-CMA with CEC2005 competitors for $D = 30$ (Wilcoxon's test with p value = .05). Best results are in boldface type.

Algorithm	$R+$ (MA-LSCh-CMA)	$R-$ (CEC2005)	Critical value	Significant differences?
BLX-GL50	**166**	44.5	52	Yes
BLX-MA	**198**	11.5	52	Yes
CoEvo	**210**	0	52	Yes
DE	**199.5**	10.5	52	Yes
K-PCX	**174**	36	52	Yes
SPC-PNX	**169.5**	40.6	52	Yes

performance comparison among MA-LSCh-CMA and each one of the algorithms was carried out by means of Wilcoxon's test. Tables 9 and 10 have the results for $D = 10$ and $D = 30$, respectively.

We note the following observations concerning Tables 9 and 10.

- Table 9 points out that six algorithms exhibited superior performance compared to MA-LSCh-CMA for $D = 10$ (their associated $R-$ values are higher than the corresponding $R+$ values for our algorithm). Nevertheless, only L-SaDE might achieve statistically significant improvements.
- The most significant remark from Table 10 is that MA-LSCh-CMA achieves better performance than all these CEC2005 competitors for $D = 30$. Again, as complexity increases, our proposal reveals satisfactory potential as a search algorithm for continuous optimisation problems.

These results, along with the remarks from Section 6.4, allow us to conclude that MA-LSCh-CMA is very competitive, as well, with the state of the art on EAs for continuous optimisation.

7 Conclusions

This paper presented the concept of the LS chain as a means to build a new MACO model that can effectively apply intense continuous LS methods as LS operators. One of the most outstanding features of this model is that it forces the continuous LS algorithm to act more intensely in the visited search zones that show higher quality.

An instance of this MACO model was implemented, called MA-LSCh-CMA, which employs a local optimiser devised by enhancing LS aptitudes of CMA-ES.

An experimental study carried out following guidelines recommended for the CEC 2005 Special Session on Real-Parameter Optimisation, has shown that it is very competitive with the state of the art on both MACOs and EAs for continuous optimisation. Particularly significant improvements were obtained for high-dimensional optimisation problems, where it might outperform G-CMA-ES, which, nowadays, has arisen as a reference point in the literature. In addition, we have confirmed empirically that MA-LSCh-CMA may really make good use of CMA-ES, and therefore, the presented MACO model is a good candidate for exploiting the introduction of the intense continuous LS method into MACOs.

Acknowledgments

The authors thank Nasimul Noman and Hitoshi Iba for letting us use their DEahcSPX code, and N. Hansen for kindly providing his implementation of CMA-ES in Matlab.

References

Abramowitz, M. (1974). *Handbook of mathematical functions, with formulas, graphs, and mathematical tables*. Dover Publications.

Auger, A., and Hansen, N. (2005a). A restart CMA evolution strategy with increasing population size. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1769–1776.

Auger, A., and Hansen, N. (2005b). Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1777–1784.

Auger, A., Schoenauer, M., and Vanhaecke, N. (2004). LS-CMAES: A second-order algorithm for covariance matrix adaptation. In *Proceedings of the Parallel Problem Solving for Nature - PPSN VIII*, Birmingham, AL.

Ballester, P. J., Stephenson, J., Carter, J., and Gallager, K. (2005). Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 498–505.

Bambha, N. K., Bhattacharyya, S. S., Teich, J., and Zitzler, E. (2004). Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155.

Beyer, H. G., and Deb, K. (2001). On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270.

Beyer, H. G., and Schwefel, H. P. (2002). Evolution strategies. *Natural Computing*, 1:3–52.

Bonissone, P. P., Subbu, R., Eklund, N., and Kiehl, T. R. (2006). Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280.

Burke, E., and Smith, A. (2000). Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE Transactions on Power Systems*, 15(1):122–128.

Caponio, A., Cascella. G. L., Neri, F., Salvatore, N., and Sumner, M. (2007). A fast adaptive memetic algorithm for online and offline control design of PMSM drives. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics, 37(1):28–41.

Chelouah, R., and Siarry, P. (2003). Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of Operational Research*, 148(2):335–348.

Davis, L. (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.

Deb, K., Anand, A., and Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter evolution. *Evolutionary Computation Journal*, 10(4):371–395.

Eshelman, L. J. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlin (Ed.), *Foundations of Genetic Algorithms 1* (pp. 265–283). San Mateo, CA: Morgan Kaufmann.

Eshelman, L. J., and Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 187–202). San Mateo, CA: Morgan Kaufmann.

Fernandes, C., and Rosa, A. (2001). A study on non-random mating and varying population size in genetic algorithms using a royal road function. In *Proceedings of the 2001 Congress on Evolutionary Computation* (pp. 60–66). Piscataway, NJ: IEEE Press.

García, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, 15:617–644.

García-Martínez, C., and Lozano, M. (2005). Hybrid real-coded genetic algorithms with female and male differentiation. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 896–903.

Goldberg, D. E., and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 69–93). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E., and Voessner, S. (1999). Optimizing global-local search hybrids. In W. Banzhaf et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference'99* (pp. 220–228). San Mateo, CA: Morgan Kaufmann.

Hansen, N. (2005). Compilation of results on the CEC benchmark function set. *Technical Report. Institute of Computational Science*, ETH Zurich, Switerland. Retrieved from http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf.

Hansen, N., and Kern, S. (2004). Evaluating the CMA evolution strategy on multimodal test functions. In Xin Yao et al. (Eds.), *Proceedings of the Parallel Problem Solving for Nature, PPSN VIII, LNCS 3242*, pp. 282–291. Berlin: Springer.

Hansen, N., Müller, S. D., and Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18.

Hansen, N., and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9(2):159–195.

Hart, W. E. (1994). *Adaptive global optimization with local search*. Ph.D. Thesis, University of California, San Diego.

Hart, W. E., Krasnogor, N., and Smith, J. E. (Eds.). (2004a). *Recent advances in memetic algorithms. Studies in Fuzzyness and Soft Computing Series*, Vol. 166. Berlin: Springer. New York: Heidelberg.

Hart, W. E., Krasnogor, N., and Smith, J. E. (2004b). Editorial introduction special issue on memetic algorithms. *Evolutionary Computation*, 12(3):v–vi.

Hart, W. E., Rosin, C. R., Belew, R. K., and Morris, G. M. (2000). Improved evolutionary hybrids for flexible ligand docking in autodock. In *Proceedings of the International Conference on Optimization in Computational Chemistry and Molecular Biology*, pp. 209–230.

Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for the behavioral analysis. *Artificial Intelligence Review*, 12(4):265–319.

Houck, C. R., Joines, J. A., Kay, M. G., and Wilson, J. R. (1997). Empirical investigation of the benefits of partial Lamarckianism. *Evolutionary Computation*, 5(1):31–60.

Iman, R. L., and Davenport, J. M. (1980). Approximations of the critical region of the Friedman statistic. *Communications in Statistics*, 18:571–595.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Joines, J. A., and Kay, G. M. (2002). Hybrid genetic algorithms and random linkage. In *Proceedings of the 2002 Congress of Evolutionary Computation*, pp. 1733–1738. Piscataway, NJ: IEEE Press.

Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference of Neural Networks*, pp. 1942–1948.

Kita, H. (2001). A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms. *Evolutionary Computation*, 9(2):223–241.

Krasnogor, N. (2002). Studies on the theory and design space of memetic algorithms. Ph.D. Thesis, University of the West of England, Bristol, United Kingdom.

Krasnogor, N., and Gustafson, S. (2004). A study on the use of self-generation in memetic algorithms. *Natural Computing*, 3(1):53–76.

Krasnogor, N., and Smith, J. (2000). A memetic algorithm with self-adapting local search: TSP as a case study. In *Proceedings of the 2000 International Conference on Genetic and Evolutionary Computation*, pp. 987–994. San Mateo, CA: Morgan Kaufmann.

Krasnogor, N., and Smith J. E. (2001). Emergence of profitable search strategies based on a simple inheritance mechanism. In *Proceedings of the 2001 International Conference on Genetic and Evolutionary Computation*, pp. 432–439. San Mateo, CA: Morgan Kaufmann.

Krasnogor, N., and Smith, J. E. (2005). A tutorial for competent memetic algorithms: Model, taxonomy, and design issue. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488.

Laguna, M., and Martí, R. (2003). *Scatter search. Methodology and implementation in C.* Dordrecht, The Netherlands: Kluwer Academic Publishers.

Land, M. W. S. (1998). Evolutionary algorithms with local search for combinatorial optimization. Ph.D. Thesis, University of California, San Diego.

Langdon, W. B., and Poli, R. (2008). Evolving problems to learn about particle swarm optimizers and other search algorithms. *IEEE Transactions on Evolutionary Computation*, 11(5):561–578.

Lee, C.-Y., and Yao, X. (2004). Evolutionary programming using mutations based on the Levy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13.

Liang, J. J., and Suganthan, P. N. (2005). Dynamic multi-swarm particle swarm optimizer with local search. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 522–528.

Liu., D., Tan, K. C., Goh, C. K., and Ho, W. K. (2007). A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 37(1):42–50.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004). Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302.

Lunacek, M., and Whitley, D. (2006). The dispersion metric and the CMA evolution strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)*, pp. 477–484.

Lunacek, M., Whitley, D., and Knight, J. N. (2005). Measuring mobility and the performance of global search algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, pp. 1209–1216.

Merz, P. (2000). Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies. Ph.D. Thesis, University of Siegen, Germany.

Molina, D., Herrera, F., and Lozano, M. (2005). Adaptive local search parameters for real-coded memetic algorithms. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 888–895.

Moscato, P. A. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Technical Report. Caltech Concurrent Computation Program Report 826*, Caltech, Pasadena, CA.

Moscato, P. A. (1999). Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glower (Eds.), *New ideas in optimization* (pp. 219–234). New York: McGraw-Hill.

Moscato, P. A., and Cotta, C. (2003). A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 105–144). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Mühlenbein, H., and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm in continuous parameter optimization. *Evolutionary Computation*, 1:25–49.

Mühlenbein, H., Schomisch, M., and Born, J. (1991). The parallel genetic algorithm as function optimizer. In R. Belew and L. B. Booker (Eds.), *Fourth International Conference on Genetic Algorithms* (pp. 271–278). San Mateo, CA: Morgan Kaufmann.

Noman, N., and Iba, H. (2005). Enhancing differential evolution performance with local search for high dimensional function optimization. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO 2005)*, pp. 967–974.

Noman, N., and Iba, H. (2008). Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125.

Nomura, T., and Shimohara, K. (2001). An analysis of two-parent recombinations for real-valued chromosomes in an infinite population. *Evolutionary Computation*, 9(3):283–308.

Ong, Y. S., and Keane, A. J. (2004). Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):99–110.

Ong, Y. S., Krasnogor, N., and Ishibuchi, H. (2007). Guest editorial: Special issue on memetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):2–5.

Ong, Y. S., Lim, M.-H., Zhu, N., and Wong, K. W. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics*, 36(1):141–152.

Posik, P. (2005). Real parameter optimisation using mutation step co-evolution. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 872–879.

Qin, A .K., and Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1785–1791.

Renders, J. M., and Bersini, H. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 312–317. Piscataway, NJ: IEEE Press.

Renders, J. M., and Flasse, S. P. (1996). Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(2):243–258.

Ronkkonen, J., Kukkonen, S., and Price, K. V. (2005). Real-parameter optimization with differential evolution. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 506–513.

Rosin, C. D., Halliday, R. S., Hart, W. E., and Belew, R. K. (1997). A comparison of global and local search methods in drug docking. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 221–228. San Mateo, CA: Morgan Kaufmann.

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*. Boca Raton, FL: CRC Press.

Shimodaira, H. (1996). A new genetic algorithm using large mutation rates and population-elitist selection (GALME). In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 25–32.

Sinha, A., Tiwari, S., and Deb., K. (2005). A population-based, steady-state procedure for real-parameter optimization. In *2005 IEEE Congress on Evolutionary Computation*, pp. 514–521.

Smith, J. E. (2007). Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 37(1):6–17.

Solis, F. J., and Wets, R. J.-B. (1981). Minimization by random search techniques. *Mathematical Operations Research*, 6:19–30.

Storn, R., and Price, K. V. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical Report. Nanyang Technological University. Retrieved from http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/Tech-Report-May-30-05.pdf.

Sywerda, G. (1989). Uniform crossover in genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms*, pp. 2–9. San Mateo, CA: Morgan Kaufmann.

Tang, J., Lim, M. H., and Ong, Y. S. (2007). Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. *Soft Computing*, 11(9):873–888.

Tsutsui, S., Yamamura, M., and Higuchi, T. (1999). Multi-parent recombination with simplex crossover in real-coded genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 657–664. San Mateo, CA: Morgan Kaufmann.

Wei, L., and Zhao, M. (2005). A niche hybrid genetic algorithm for global optimization of continuous multimodal functions. *Applied Mathematics and Computation*, 160(3):649–661.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the International Conference on Genetic Algorithms*, pp. 116–121. San Mateo, CA: Morgan Kaufmann.

Whitley, D., Rana, S., Dzubera, J., and Mathias, E. (1996). Evaluating evolutionary algorithms. *Artificial Intelligence*, 85:245–276.

Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Yang, J.-M., and Kao, C.-Y. (2000). Integrating adaptive mutations and family competition into genetic algorithms as function optimiser. *Soft Computing*, 4:89–102.

Yuan, B., and Gallagher, M. (2005). Experimental results for the special session on real-parameter optimization at CEC 2005: A simple, continuous EDA. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1792–1799.

Zar, J. H. (1999). *Biostatistical analysis*. Englewood, NJ: Prentice Hall.

Zhang, C.-K., and Shao, H. H. (2001). A hybrid strategy: Real-coded genetic algorithm and chaotic search. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics 2001*, pp. 2361–2364.

A Statistical Tests

In this paper, we considered two alternative methods based on nonparametric tests to analyse the experimental results: (1) Application of Iman and Davenport's test and Holm's method as a post hoc procedure (Section A.1); and (2) Utilization of the Wilcoxon matched-pairs signed-ranks test (Section A.2).

A.1 Iman-Davenport's Test and Holm's Method as a Post Hoc Procedure

In order to detail Iman-Davenport's test, we first describe Friedman's test. This 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?* Friedman's test is a nonparametric procedure employed in a hypothesis testing situation involving a design with two or more samples. It is the analogue of the repeated-measures ANOVA for nonparametrical statistical procedures; therefore, it is a multiple comparison test that aims to detect significant differences between the behaviour of two or more algorithms.

The null hypothesis for Friedman's test is $H_0 : \theta_1 = \theta_2 = \cdots = \theta_k$; the median of the population i represents the median of the population $j, i \neq j, 1 \leq i \leq k, 1 \leq j \leq k$. The alternative hypothesis is $H_1 : Not\ H_0$, so it is nondirectional.

In the following, we describe the tests' computations. It computes the ranking of the observed results for the algorithm (r_j for algorithm j with k algorithms) for each function, assigning to the best of them the ranking 1, and to the worst the ranking k. Under the null hypothesis, formed from supposing that the results of the algorithms are equivalent and, therefore, their rankings are also similar, Friedman's statistic

$$\chi^2_{\mathrm{F}} = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right],$$

is distributed according to χ^2_{F} with $k-1$ degrees of freedom, where $R_j = \frac{1}{N}\sum_i r_i^j$, and N is the number of functions. The critical values for the Friedman's statistic coincide with the ones determined by the χ^2 distribution when $N > 10$ and $k > 5$. In a contrary case, the exact values can be seen in Sheskin (2003) and Zar (1999).

Iman and Davenport (1980) proposed a derivation from Friedman's statistic given that this last metric produces a conservative undesirable effect. The proposed statistic is

$$F_{\mathrm{F}} = \frac{(N-1)\chi^2_{\mathrm{F}}}{N(k-1)-\chi^2_{\mathrm{F}}},$$

and it is distributed according to an F distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

Computation of the p values given a χ^2 or F_F statistic can be done by using the algorithms in Abramowitz (1974). Also, most of the statistical software packages include it.

The rejection of the null hypothesis in both tests described above does not involve the detection of the existing differences among the algorithms compared. They only inform us about the presence of differences among all samples of results compared. In order to conduct pairwise comparisons within the framework of multiple comparisons, we can proceed with a post hoc procedure. In this case, a control algorithm (perhaps a proposal to be compared) is usually chosen. Then, the post hoc procedures compare the control algorithm with the remaining $k-1$ algorithms.

The Holm method is one of these procedures. It sequentially checks the hypotheses ordered according to their significance. We will denote the p values ordered by $p_1, p_2, \ldots,$ in the way that $p_1 \leq p_2 \leq \cdots \leq p_{k\text{-}1}$. Holm's method compares each p_i with $\alpha/(k-i)$ starting from the most significant p value. If p_1 is below $\alpha/(k-1)$, the corresponding hypothesis is rejected and it leaves us to compare p_2 with $\alpha/(k-2)$. If the second hypothesis is rejected, we continue with the process. As soon as a certain hypothesis cannot be rejected, all the remaining hypotheses are maintained as supported. The statistic for comparing the ith algorithm with the jth algorithm is:

$$z = (R_i - R_j) \Big/ \sqrt{\frac{k(k+1)}{6N}}.$$

The value of z is used for finding the corresponding probability from the table of the normal distribution (p value), which is compared with the corresponding value of α.

A.2 The Wilcoxon Matched-Pairs Signed-Ranks Test

Wilcoxon's test is used for answering this question: *do two samples represent two different populations?* It is a nonparametric procedure employed in a hypothesis testing situation involving a design with two samples. It is the analogue of the paired t test in non-parametrical statistical procedures; therefore, it is a pairwise test that aims to detect significant differences between the behavior of two algorithms.

The null hypothesis for Wilcoxon's test is $H_0 : \theta_D = 0$; in the underlying populations represented by the two samples of results, the median of the difference scores equals zero. The alternative hypothesis is $H_1 : \theta_D \neq 0$, but another alternative can also be used, $H_1 : \theta_D > 0$ or $H_1 : \theta_D < 0$ as a directional hypothesis.

In the following, we describe the test computations. Let d_i be the difference between the performance scores of the two algorithms on ith out of N functions. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of ranks for the functions on which the second algorithm outperformed the first, and R^- the sum of ranks for the opposite. Ranks of $d_i = 0$ are split evenly among the sums; if there is an odd number of them, one is ignored:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \text{ and } R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$$

Let T be the smallest of the sums, $T = min(R^+, R^-)$. If T is less than or equal to the value of the distribution of the Wilcoxon for N degrees of freedom (Table B.12 in Zar, 1999), then the null hypothesis of equality of means is rejected.

The obtaining of the p value associated with a comparison is performed by means of the normal approximation for the Wilcoxon T statistic (Section VI, Test 18 in Sheskin, 2003). Furthermore, the computation of the p value for this test is usually included in well-known statistical software packages (SPSS, SAS, R, etc.).

B Results of the Experiments

Table 11: Average of the errors achieved by MA-LSCh-CMA.

Test Function	Dimension 10	Dimension 30	Dimension 50
F6	7.919168e–9	1.191003e+1	6.571714e+1
F7	1.576340e–2	8.871392e–4	2.365166e–3
F8	2.025390e+1	2.027016e+1	2.048963e+1
F9	7.955018e–9	7.827714e–9	6.904902e–9
F10	2.547095e+0	1.838684e+1	3.745090e+1
F11	4.996535e–1	4.350834e+0	1.081456e+1
F12	1.830865e+2	7.690185e+2	2.762453e+3
F13	5.483822e–1	2.344814e+0	3.510274e+0
F14	2.184448e+0	1.268192e+1	2.229458e+1
F15	2.437411e+2	3.080000e+2	2.880000e+2
F16	9.273844e+1	1.363134e+2	6.402193e+1
F17	9.299357e+1	1.345630e+2	8.318522e+1
F18	8.335419e+2	8.156512e+2	8.454191e+2
F19	8.436303e+2	8.163714e+2	8.454539e+2
F20	8.091376e+2	8.157765e+2	8.414372e+2
F21	7.756537e+2	5.120000e+2	5.449624e+2
F22	7.376647e+2	5.258481e+2	5.000145e+2
F23	9.242833e+2	5.341643e+2	5.809396e+2
F24	2.643189e+2	2.000000e+2	2.000000e+2
F25	4.234632e+2	2.108472e+2	5.809396e+2

Table 12: Average of the errors achieved by S-MACO.

Test Function	Dimension 10	Dimension 30	Dimension 50
F6	2.559080e+0	6.827221e+1	3.321231e+0
F7	4.228532e–3	3.350353e–3	3.946206e–10
F8	2.032996e+1	2.079407e+1	4.985846e+0
F9	7.493495e–9	7.717354e–9	7.181764e–9
F10	3.803494e+0	3.235029e+1	8.973962e+1
F11	1.104567e+0	1.828145e+1	4.429612e+1
F12	1.596333e+2	3.281777e+3	1.299585e+4
F13	5.035934e–1	4.374800e+0	1.274590e+1
F14	2.947613e+0	1.306008e+1	2.276375e+1
F15	2.572330e+2	3.080000e+2	3.240408e+2
F16	9.466710e+1	1.864778e+2	1.602585e+2
F17	9.837184e+1	2.226743e+2	1.610696e+2
F18	8.145547e+2	8.441590e+2	8.768230e+2
F19	8.741588e+2	8.424100e+2	8.693493e+2
F20	8.276920e+2	8.402913e+2	8.780883e+2
F21	7.687822e+2	5.000000e+2	5.388100e+2
F22	7.223773e+2	5.454707e+2	5.189131e+2
F23	1.008124e+3	5.592260e+2	5.391635e+2
F24	2.202741e+2	2.000000e+2	2.000000e+2
F25	5.112336e+2	2.107127e+2	2.161979e+2

Table 13: Average of the errors achieved by DEahcSPX.

Test Function	Dimension 10	Dimension 30	Dimension 50
F6	7.973158e–1	1.000000e–9	7.359388e–1
F7	1.604629e–1	1.163264e–3	9.542600e–2
F8	2.054373e+1	2.094711e+1	2.113583e+1
F9	1.144203e+0	1.000000e–9	1.000000e–9
F10	1.567427e+1	9.449920e+1	2.342322e+2
F11	4.487526e+0	2.921885e+1	5.699720e+1
F12	2.304905e+2	2.956616e+4	1.718276e+5
F13	4.191988e–1	2.365826e+0	5.566580e+0
F14	3.482625e+0	1.279216e+1	2.253116e+1
F15	2.522842e+2	3.506300e+2	2.860367e+2
F16	1.227970e+2	1.294508e+2	1.799057e+2
F17	1.461102e+2	2.048724e+2	2.743071e+2
F18	7.908927e+2	9.060900e+2	9.306997e+2
F19	7.175312e+2	9.061617e+2	9.316814e+2
F20	7.273535e+2	9.065054e+2	9.313493e+2
F21	7.596139e+2	5.000000e+2	6.901346e+2
F22	8.083156e+2	9.120960e+2	9.684325e+2
F23	9.428651e+2	5.341641e+2	7.904251e+2
F24	3.128550e+2	2.000000e+2	2.000000e+2
F25	4.086841e+2	2.105413e+2	2.484206e+2