Memetic Algorithm with Local Search Chaining for Large Scale Continuous Optimization Problems

Daniel Molina Manuel Lozano Francisco Herrera

Abstract— **Memetic algorithms arise as very effective algorithms to obtain reliable and high accurate solutions for complex continuous optimization problems. Nowadays, high dimensional optimization problems are an interesting field of research. The high dimensionality introduces new problems for the optimization process, making recommendable to test the behavior of the optimization algorithms to large-scale problems. The Local search method must be applied with a higher intensity, specially to most promising solutions, to explore the higher domain space around each solution. In this work, we present a preliminar study of a memetic algorithm that assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. This algorithm have obtained good results in continuous optimization and we study whether is a good algorithm for large scale optimizations problems.**

We make experiments of our proposal using the benchmark problems defined in the Special Session or Competition on *Large Scale Global Optimisation*, on the IEEE Congress on Evolutionary Computation in 2008. First, we test different local search methods to identify the best one. Then, we compare the proposed algorithm with the algorithms used into the competition, obtaining that our proposal is a very promising algorithm for this type of high-dimensional problems: with dimension 500 our proposal is the second best of the compared algorithms, and the best memetic algorithm.

I. Introduction

It is well known that the hybridisation of *evolutionary algorithms* (EAs) with other techniques can greatly improve the search efficiency [4, 6]. EAs that have been hybridised with local search (LS) techniques are often called *memetic algorithms* (MAs) [16, 17, 14]. One commonly MA scheme improves the new created solutions using an LS method, with the aim of exploiting the best search regions gathered during the global sampling done by the EA. That allows us to design MAs for continuous optimization (MACO) that obtain high accurate solutions for these problems [9, 22, 11], like SMFDE[3] or FAMA[2].

Nowadays, high-dimensional optimization problems arise as a very interesting field of research, since they appear in many recent real-world problems (bio-computing, data mining, etc.). Unfortunately, the performance of most available optimization algorithms deteriorates very quickly when the dimensionality increases [27]. Thus, the ability of being scalable for high-dimensional problems becomes an essential requirement for modern optimization algorithm approaches.

Daniel Molina is with the Department of Computer Languages and Systems, Cádiz, Spain; (email: daniel.molina@uca.es). Manuel Lozano and Francisco Herrera are with the Department of Computer Science and Artificial Inteligence, Granada, Spain; (emails: lozano@decsai.ugr.es, herrera@decsai.ugr.es).

In MAs, the LS method is the component more directly affected by dimensionality. These methods are used to explore in a nearly region around the current solutions. Thus, a higher dimension increases the domain search and the region to explore. This larger area to explore suggests the convenience to search around them with higher intensity.

In a previous work, we have defined a MA for continuous optimization specifically designed to adapt the LS intensity, exploiting with higher intensity the most promising individuals [15]. To adapt the LS intensity in the proposed model the LS can be applied several times over the same individual, using a fixed LS intensity, and storing its final parameters, creating *LS chains*. Using these *LS chains* an individual previously improved by an LS invocation may later become the initial point of a next LS application, using the final strategy parameter values achieved by the previous one as its initial ones in the following application. In this way, the continuous LS method may *adaptively* fit its strategy parameters to the particular features of these regions. This MA using LS chaining obtains very good results for continuous optimization for medium scale problems [15].

In this work, we want to extend this study to test its behavior again large scale problems. In the *Special Session on Large Scale Global Optimisation* in the 2008 IEEE Congress on Evolutionary Computation (CEC'2008 in the following) there was proposed several benchmark functions specifically designed for this type of problems. In this session different algorithms were proposed to deal with these problems using the same guideline for the experiments. Comparing our proposal using the same guideline give us the opportunity of comparing our model with them. We first undertake a study of the performance of different MACOs based on LS chains that apply different instances of LS searches Then, the instance that gives the best results is compared with all the algorithms presented into the CEC'2008 competition.

This contribution is set up as follows. In Section II, we present the different LS methods considered. In Section III, we describe the proposed MA. In Section IV, we present a experimental study to identify the best LS method and then our MA is compared with the algorithms proposed in the CEC'2008 Competition for Large Scale Global Optimization. Finally, in Section V, we provide the main conclusions of this work.

II. Local Search and Large Scale Optimisation

In this section, we present a detailed description of the continuous LS methods used in our empirical study. They are composed by two well-known continuous local searchers:

978-1-4244-2959-2/09/$25.00 ©

the Solis and Wets's algorithm [20], the Nelder and Mead's simplex method [19]; and two new individuals previously defined in the algorithm *Multiple Trajectory Search*[26].

Next, we present a detail description of these procedures, remarking the most interesting features for large scale optimization problems.

A. Solis and Wets' algorithm

This LS algorithm is a randomized 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 p. 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, p is increased to move more quickly. After several failures in a row, p is decreased to focus the search. Additionally, a bias term is included to put the search momentum in directions that yield success. See [20] for details. This is a simple LS that can adapt its size search very quickly.

B. Nelder-Mead simplex algorithm

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

Through a sequence of elementary geometric transformations, the initial simplex moves, expand or contracts. To select the appropriate transformation, the method only uses the values of the function to be optimised at the vertices of the simplex considered. After each transformation, a better vertex replaces the current worst one.

At the beginning of the algorithm, only the point of the simplex where the objective function is worst is replaced, and another point, the image of the worst one, is generated. This is the reflection operation. If the reflected point is better than all other points, the method expands the simplex in this direction, otherwise, if it at least better than the worst one, the algorithm performs again the reflection with the new worst point. The contraction step is performed when the worst point, in such a way that the simplex adapts itself to the function landscape and finally surrounds the optimum. If the worst point is better than the contracted point, the multi-contraction is performed. At each step we check that the generated point is not outside the allowed reduced solution space.

This method has the advance of initially creatimh the simplex composed by a movement in each direction. This is a very useful characteristic to explore high dimensionality spaces.

Please, refer to [19] for more details about this procedure.

C. LS Methods based in the MTS algorithm

The above LS methods are very classical LS methods that can obtain good results in large scale optimization problems, but they are not specifically designed to these type of algorithms. In the *CEC'2008 competition* different authors apply LS methods specially designed for this type of problems. One of them is the algorithm *Multiple Trajectory Search* (MTS) [26]. MTS was designed for multi-objective problems [25], but it has proved also to be a very good algorithm in large scale optimization, obtained the best results in the CEC'2008 competition [23].

The MTS consists of iterations of local searches over a population randomly initialized to create a simulated orthogonal array. The MTS apply to each individual I three local search, obtaining three new solutions I_{c1}, I_{c2}, I_{c3}. The original individual I is replaced by the best of these three new solutions.

The original paper calls these LS methods *LocalSearch1, LocalSearch2 and LocalSearch3*, we are going to name them *MTS-LS1, MTS-LS2 and MTS-LS3* to avoid any confusion with the previously described LS methods. We consider only the first two LS methods, because MTS-LS3 is more similar in structure to the Solis Wets' algorithm.

Both *MTS-LS1* and *MTS-LS2* are hill-climbing algorithms that explore one dimension each time.

In each step of these algorithms, the variable of the current dimension is decreased by a certain value (called SR in the paper) to check if the objective function value is improved. If the new solution improves the original one, the search proceeds to consider the next dimension. If it is worse, the original solution is restored and then the same variable is increased by 0.5*SR to see if the objective function value is improved again. In this case, the search proceeds to consider the next dimension. In other case, the solution is restored and the search proceeds to consider the next dimension. $LocalSearch1$ and $LocalSearch2$ are explained in more detail in [26].

MT-LS2 is very similar to *MT-LS1*. There are only two differences: First, the addition factor (SR) is multiply by -1 the 50% of times. They also differ in the number of genes improved: *MTS-LS1* explore all the genes, while *MT-LS2* considers only a quarter of them (randomly selected by each application of the LS).

III. MACOs Based on LS Chains

In this section, we describe a MACO approach proposed in [15] that employs continuous LS methods as LS operators. It is a steady-state MA model that employs the concept of *LS chain* to adjust the LS intensity assigned to the intense continuous LS method. In particular, this MACO handles LS chains, throughout the evolution, with the objective of allowing the continuous LS algorithm to act more intensely in the most promising areas represented in the EA population. In this way, the continuous LS method may adaptively fit its strategy parameters to the particular features of these zones.

In Section III-A, we introduce the foundations of steady-state MAs. In Section III-B, we explain the concept of LS

chain. Finally, in Section III-C, we give an overview of the MACO approach presented in [15], that handles LS chains with the objective of make good use of intense continuous LS methods as LS operators.

A. Steady-State MAs

In steady-state GAs [21] usually only or two offspring are produced in each generation. Parents are selected to produce offspring and then a decision is made to select which individuals in the population will be deleted in order to make room to new offspring. Steady-state GAs are *overlapping* systems because parents and offspring compete for survival. A widely used replacement strategy is to replace the worst individual only if the new individual is better. We will call this strategy the *standard replacement strategy*.

Although steady-state GAs are less common than generational GAs, Land [12] recommended their use for the design of *steady-state MAs* (steady-state GAs plus LS method) 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.

B. Local Search Chains

In steady-state MAs, individuals by the LS invocations may reside in the population during a long time. This circumstance allows these individuals to become starting points of subsequent LS invocations. In [15], Molina et al. propose to *chain* an LS algorithm invocation and the next one as follows:

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

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

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

- Every time the LS algorithm is applied to refine a particular chromosome, a fixed LS intensity should be considered for it, which will be called *LS intensity stretch* (I_{str}).
  In this way, 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 $n_{app} \cdot I_{str}$ fitness function evaluations.
- After the LS operation, the parameters that define the current state of the LS processing are stored along with the final individual reached (in the steady-state GA population). When this individual is selected to be improved, the initial values for the parameters of the LS algorithm will be directly available. For instance: In Solis Wets the count of successful and failed iterations is stored. In Simplex algorithm the simplex structure is stored, and in *MTS-LS1* and *MTS-LS2* algorithms, the following genes to be modified are stored.

C. A MACO Model that Handles LS Chains

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

1) It is a steady-state MA model.
2) It ensures that a fixed and predetermined local/global search ratio is always kept. With this policy, we easily stabilise this ratio, which has a strong influence on the final MACO behavior. 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 are showing promising fitness improvements in the best current search areas represented in the steady-state GA population. In addition, it encourages the activation of innovative LS chains with the aim of refining unexploited zones, whenever the current best ones may not offer profitability. The criterion to choose the individuals that should undergo LS is specifically designed to manage the LS chains in this way (Steps 3 and 4).

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

Fig. 1. Pseudocode algorithm for the proposed MACO model

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 equation $n_{frec} = I_{str}\frac{1-r_{L/G}}{r_{L/G}}$, where n_{str} is the LS intensity stretch (Section III-B), and $r_{L/G}$ is defined as the percentage of evaluations spent doing LS from the total assigned to the algorithm's run.

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

1) Build the set of individuals in the steady-state GA population, S_{LS} that fulfils:
   a) They have never been optimised by the intense continuous LS algorithm, or
   b) They previously underwent LS, obtaining a fitness function improvement greater than δ_{LS}^{min} (a parameter of our algorithm).
2) If $|S_{LS}| \neq 0$, then apply the continuous LS algorithm to the best individual in this set. On other case, the population of the steady-state MA is restarted randomly (keeping the best individual).

With this mechanism, when the steady-state GA finds a new best individual, it will be refined immediately. Furthermore, 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 a δ_{LS}^{min} threshold.

IV. Experiments

In this section, we present the experimental study carried out by different instances of the proposal. Each instance differs only in the LS method applied; each one for each LS method described in Section II. First, these instances are compared between them to identify which LS method give between results when the dimension is increased. Then, the best of them is compared with the algorithms proposed into the CEC'2008 competition.

This section is structured in the following way. In Section IV-A, we describe the instances of MACO used for the experiments. In Section IV-B, we detail the experimental setup and statistical methods that were used for this experimental study. In Section IV-C, we compare the different instances to identify the best one and, in IV-D, we compare the best instance with all the other algorithms proposed into the CEC'2008 competition.

A. Instances of MACO Based on LS Chains

In this section, we build different instances of the MACO model described in Figure 1, that only differ in the LS method applied, one instance for each LS method considered in Section II: The Solis and Wets' algorithm, the Nelder and Mead's simplex method, and the LS methods *MTS-LS1* and *MTS-LS2*.

Next, we list their main features:

Steady-state GA: It is a real-coded steady-state GA [21] specifically designed to promote high population diversity levels by means of the combination of the $BLX-\alpha$ crossover operator (see [5]) with a high value for its associated parameter ($\alpha = 0.5$) and the *negative assortative mating* strategy [7]. Diversity is favored as well by means of the BGA mutation operator (see [18]).

Continuous LS algorithms: The instances follow the MACO approach that handles LS chains, with the objective of tuning the intensity of each LS algorithm considered, which are employed with a high intensity to promising solutions. Each LS method 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 LS parameters (different for each LS method). Latter, when the LS method is applied to this inserted solution, these values will be recovered to proceed with a new LS application. When the LS method is carried out on a solution that do not belong to existing chains, the defaults values are assumed.

Parameter setting: For the experiments, we have use the same parameters that have give very good results in previous studies [15]. the MACO instances apply $BLX-\alpha$ with $\alpha = 0.5$. The population size is 60 individuals and the probability of updating a chromosome by mutation is 0.125. The n_{ass} parameter associated with the negative assortative mating is set to 3. They use $I_{str} = 500$ and $r_{L/G} = 0.5$. In this case, $\delta_{min}^{LS} = 0$ because in CEC'2008 functions there is no fitness threshold value.

B. Experimental Setup and Statistical Analysis

This experimental study has been carried by using the *CEC'2008 test suite*, that it is a set of benchmark functions specifically designed as large-scale problems, which were defined for *CEC'2008 Special Session on Large Scale Global Optimisation*. This test suite is used to study the behavior of the MACOs on problems with high dimensionality. It consists of 7 test functions with three different dimension values: D=100, 500 and 1000. In [24] the complete information and its source code can be obtained.

The experiments have been carried out following the instructions indicated in the documents associated to the Special Session, to be able to compare our proposal with the algorithms presented in this session. The main characteristics are:

1) Each algorithm is run 25 times for each test function, and the average of error of the best individual of the population is computed.
2) The study has been made with dimension D=100, 500 and 1000.
3) The maximum number of fitness evaluations is $5,000 \cdot D$. Each run stops when this maximum number of evaluations is achieved.

To analyse the results we have used *non-parametric tests*. These tests uses the mean ranking of each algorithm. We have applied them because several times the functions can not follow the requirement to apply the parametric test with security[8].

In particular, we have considered an alternative method based on non parametric tests to analyse the experimental results, that has been previously applied in comparisons of EAs [8], in which there are explained in detail:

- Application of *Iman and Davenport's* test, and *Holm*'s test as post-hoc procedure. The first one is used to know if between the algorithms there are statistically relevance differences. In that case, a post-hoc procedure, *Holm*'s test, is is employed to know which algorithms

are statistically worse than the algorithm with the best ranking.

C. Comparison the different local search methods

In this section, we compare the different instances described in section IV-A. Tables IV and V in the Appendix show the results. In the following, we are going to analyse them.

TABLE I
RESULTS OF THE IMAN-DAVENPORT'S TEST FOR EACH DIMENSION

Dimension	Iman-Davenport value	Critical value	Sig. differences?
100	0,062	3,29	No
500	5,760	3,29	Yes
1000	0,242	3,29	No

First, we applied Iman-Davenport test to check if there is a significant difference between them. Table I shows the results.From Table I we can see that there are few differences between the instances. Only for dimension 500 we may observe the existence of significant differences among the results (the statistical value is greater than the critical one, $3,29$). Thus, we compare for dimension 500 the different instances by Holm's test.

TABLE II
COMPARISON, USING HOLM'S TEST, OF THE INSTANCES VERSUS USING THE LS METHOD *MTL-S2*, WITH DIMENSION 500

$LSMethod$	z	p-value	α/i	Sig. differences?
Simplex	2,8983	0,00375	0,0166	Yes
Solis Wets	2,4842	0,01298	0,0250	Yes
MTS-LS1	1,2421	0,21419	0,0500	No

Table II shows the result of the comparison, obtaining than *MTS-LS2* is the best LS method, and statistically better than Simplex and Solis Wets method.

Figure 2 shows the average ranking for each instance for each dimension value. Clearly, *MTS-LS2* is the best LS method for dimension 500, and in the other dimensions there are no significant differences.

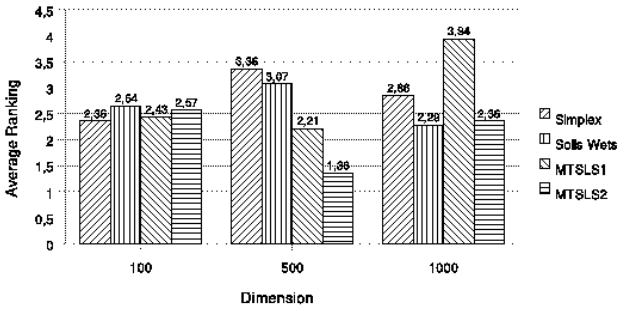

Fig. 2. Mean Ranking of each instance in function of the dimension

Thus, we use the LS method *MTS-LS2* in the following comparison in the next section.

D. Comparison with the algorithms of the CEC'2008 Competition

In this section we will test our proposal (called MA with LS Chains, *MA-LS-Chains*, in the following) , with the algorithms proposed into the CEC'2008 competition. The algorithms compared are: MLCC[29], EPUS-PSO[10], jDEdynNP-F[1], MTS[26], DEwSAcc[30], DMS-PSO[31], and LSEDA-gl[28]. UEP[13] is not used because it does not show all required results for the comparison. Curiously, although several algorithms are MAs, no one of them uses a GA to explore, but other EAs (Particle Swarm Optimizer or Differential Evolution, mainly).

The error values of all these algorithms have been obtained from the associated papers. Tables VII, VIII and IX show their error values.

First, we applied the Iman-Davenport test to see if there is a significant difference between them. Table III shows the results.

TABLE III
RESULTS OF THE IMAN-DAVENPORT'S TEST FOR EACH DIMENSION

$Dimension$	Iman-Davenport value	Critical value	Sig. differences?
100	4,7937	2,24	Yes
500	4,4115	2,24	Yes
1000	3,4548	2,24	Yes

From Table III we can see that there are significant differences for each dimension.

1) Dimension 100: in Figure 3 there are shown all the compared algorithms by its average ranking.

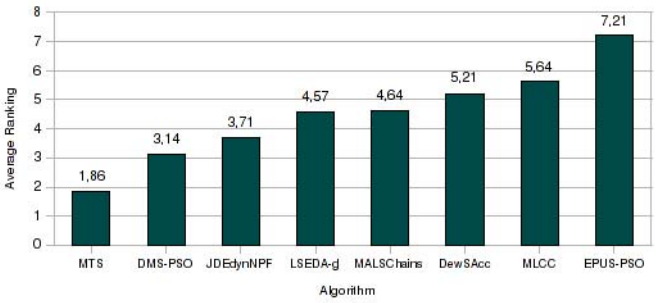

Fig. 3. Average Ranking for algorithms CEC'2008 and *MA-LS-Chains*, Dimension 100

We obtain that our algorithm is the 5th algorithm (but very close to the fourth algorithm, LSEDA-gl).

2) Dimension 500: Figure 4 shows the average ranking for dimension 500, with the same structure. In this case, we obtain that our proposal is the second best algorithm, only improved by MTS. This is very important, because with this dimension it is the second best algorithm, and the best MA in the comparison.

3) Dimension 1000: Figure 5 shows the average ranking for dimension 1000. In this case, we obtain that our proposal

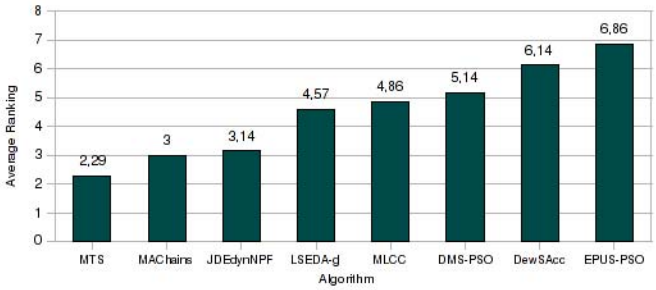

Fig. 4. Average Ranking for algorithms CEC'2008 and *MA-LS-Chains*, Dimension 500

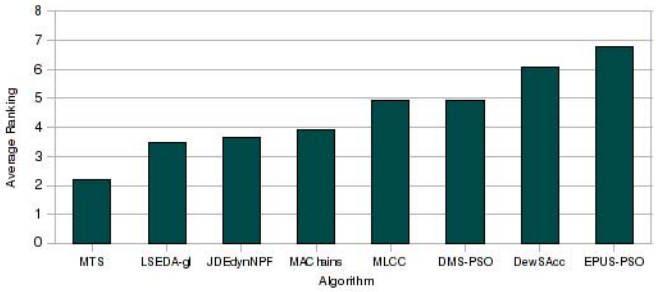

Fig. 5. Average Ranking for algorithms CEC'2008 and MALSChains, Dimension 1000

is the 4th best algorithm, but it is not statistically worse than any other.

The best algorithm is MTS that presents the most different structure (it uses only a combination of LS methods, without a component for exploration), as it has been shown in [23]. Our proposal achieves the second best results in dimension 500. In dimension 100 and 1000 our proposal is the 4th best algorithm, after JDEdynNP-F and LSEDA-gl. It is clear that the dimensionality value has a strong influence over the results of each algorithm (the DMS-PSO it is good with low intensity, but with a higher dimensionality it is clearly worse).

V. Conclusions

In this work, we have used a MA that gives very good results in medium dimensionality problems [15] and it is compared with other good algorithms in large scale global optimization problems. Our proposal is competitive considering the algorithms in the CEC'2008 competition. In particular, with dimension 500 our proposal it is the second best algorithm, only after MTS, being the best proposed MA.

These results suggests that our model is a promising algorithm for large scale problems. But the results with dimension 1000 show that we have to work in our algorithm to adapt it completely to these type of problems. One aspect to improve is the creation of an LS method specifically designed for large scale optimization. Comparing the different LS methods considered, we have obtained that *MTS-LS2* is the best one, thus the exploration of only a subset of genes can improve greatly the LS process. This idea will be explored to create a specific LS method for large scale optimization in future works.

This algorithm also shows that it can be created very good MAs with GAs. Large scale problems optimization is a field in which is very common to find other type of EAs, but GAs could be also a good option. In particular, our proposal is best with dimension higher than 100 than any proposed algorithm using the PSO scheme.

Acknowledgement

This work was supported by Project TIN2008-05854.

Appendix

In this section there are shown the different errors obtained by each instance, to allow to future algorithm to be compared with the algorithm presented in this work for large scale global continuous optimization.

D	LS Method	F1	F2	F3	F4
100	Simplex	5,68E-14	4,41E+00	2,69E+01	1,27E+00
	Solis Wets	5,68E-14	1,97E+01	1,96E+02	2,83E+00
	MTS-LS1	2,77E-17	2,58E+01	5,90E+02	4,14E+00
	MTS-LS2	2,88E-17	2,58E+01	4,75E+02	2,91E+00
500	Simplex	1,40E-16	7,15E+01	9,66E+02	5,82E+01
	Solis Wets	1,48E-16	7,34E+01	4,75E+02	2,96E+01
	MTS-LS1	5,70E-17	6,82E+01	9,44E+02	9,40E+00
	MTS-LS2	5,67E-17	2,98E+01	3,19E+02	1,20E+01
1000	Simplex	3,74E-13	1,03E+02	9,75E+02	1,95E+02
	Solis Wets	2,05E-17	9,73E+01	9,76E+02	1,95E+02
	MTS-LS1	3,25E-16	9,38E+01	1,83E+03	5,81E+02
	MTS-LS2	3,20E-16	9,40E+01	1,79E+03	5,39E+02

TABLE IV
MEAN ERRORS OBTAINED USING EACH MA-LS-CHAIN INSTANCE FOR FUNCTIONS F1-F4

D	LS Method	F5	F6	F7
100	Simplex	2,84E-14	1,00E-13	-1,47E+03
	Solis Wets	2,84E-14	8,24E-14	-1,46E+03
	MTS-LS1	1,38E-17	1,01E-14	-1,45E+03
	MTS-LS2	3,22E-17	1,02E-14	-1,43E+02
500	Simplex	7,88E-04	3,14E-14	-6,94E+03
	Solis Wets	1,31E-03	3,02E-14	-6,47E+03
	MTS-LS1	2,95E-04	3,09E-14	-6,52E+03
	MTS-LS2	1,55E-17	3,55E-15	-6,52E+03
1000	Simplex	8,85E-04	1,02E-07	-1,36E+04
	Solis Wets	1,58E-03	5,84E-14	-1,09E+04
	MTS-LS1	8,87E-04	3,62E-12	-1,27E+04
	MTS-LS2	5,71E-16	3,07E-12	-1,28E+04

TABLE V
MEAN ERRORS OBTAINED USING EACH LS MA-LS-CHAIN INSTANCE F5-F7

Tables IV and V show the results obtained with each instance.

Table VI shows the results obtained by our proposal for each dimension. For F1-F6 it is shown the mean error obtained by the 25 running. For F7, because there is unknown the optimum, it is shown the fitness value.

Dimension	F1	F2	F3	F4
100	2,88E-17	2,58E+01	4,75E+02	2,91E+00
500	5,67E-17	2,98E+01	3,19E+02	1,20E+01
1000	3,20E-16	9,40E+01	1,79E+03	5,39E+02
Dimension	F5	F6	F7	-
100	3,22E-17	1,02E-14	-1,43E+02	
500	1,55E-17	3,55E-15	-6,52E+03	
1000	5,71E-16	3,07E-12	-1,28E+04	

TABLE VI

MEAN ERRORS OBTAINED BY MA-LS-CHAINS (USING *MTS-LS2* METHOD) FOR EACH DIMENSION

Algorithm	F1	F2	F3	F4
DewSAcc	0,00E+00	8,25E+00	1,45E+02	4,38E+00
DMS-PSO	0,00E+00	3,64E+00	2,83E+01	1,83E+00
EPUS-PSO	5,67E-17	1,86E+01	4,99E+03	4,71E+02
JDEdynNP-F	9,32E-14	4,29E-01	1,12E+02	5,46E-14
LSEDA-gl	2,27E-13	2,21E-13	2,81E+02	1,31E+02
MA-LS-Chains	4,30E-13	2,68E+01	2,62E+00	2,62E+00
MLCC	2,09E-09	2,53E+01	1,49E+02	4,39E-13
MTS	8,45E+01	1,44E-11	5,17E-08	0,00E+00
Algorithm	F5	F6	F7	-
DewSAcc	3,07E-14	1,13E-13	-1,37E+03	
DMS-PSO	0,00E+00	0,00E+00	-1,14E+03	
EPUS-PSO	3,72E-01	2,06E+00	-8,55E+02	
JDEdynNP-F	2,84E-14	5,68E-14	-1,48E+03	
LSEDA-gl	2,84E-14	9,78E-14	-1,46E+03	
MA-LS-Chains	1,38E-17	1,03E-14	-1,42E+03	
MLCC	3,41E-14	1,11E-13	-1,54E+03	
MTS	0,00E+00	0,00E+00	-1,49E+03	

TABLE VII

COMPARISON RESULTS FOR DIMENSION 100

Algorithm	F1	F2	F3	F4
DewSAcc	2,09E-09	7,57E+01	1,81E+03	3,64E+02
DMS-PSO	0,00E+00	6,89E+01	4,67E+07	1,61E+03
EPUS-PSO	8,45E+01	4,35E+01	5,77E+04	3,49E+03
JDEdynNP-F	9,32E-14	8,46E+00	6,61E+02	1,47E-12
LSEDA-gl	2,27E-13	2,72E-10	8,67E+02	8,55E+02
MA-LS-Chains	5,67E-17	2,98E+01	3,19E+02	1,20E+01
MLCC	4,30E-13	6,67E+01	9,25E+02	1,79E-11
MTS	0,00E+00	7,32E-06	5,03E-03	0,00E+00
Algorithm	F5	F6	F7	-
DewSAcc	6,90E-04	4,80E-01	-5,75E+03	
DMS-PSO	0,00E+00	2,00E+00	-4,20E+03	
EPUS-PSO	1,64E+00	6,64E+00	-3,51E+03	
JDEdynNP-F	4,21E-14	1,49E-13	-6,88E+03	
LSEDA-gl	1,14E-13	3,13E-13	-4,83E+01	
MA-LS-Chains	1,55E-17	3,55E-15	-6,52E+03	
MLCC	2,13E-13	5,34E-13	-4,44E+03	
MTS	0,00E+00	6,18E-12	-7,08E+03	

TABLE VIII

COMPARISON RESULTS FOR DIMENSION 500

Algorithm	F1	F2	F3	F4
DewSAcc	8,79E-03	9,61E+01	9,15E+03	1,82E+03
DMS-PSO	0,00E+00	9,15E+01	8,98E+09	3,84E+03
EPUS-PSO	5,53E+02	4,66E+01	8,37E+05	7,58E+03
JDEdynNP-F	1,14E-13	1,95E+01	1,31E+03	2,17E-04
LSEDA-gl	3,22E-13	1,04E-05	1,73E+03	5,45E+02
MA-LS-Chains	3,20E-16	9,40E+01	1,79E+03	5,39E+02
MLCC	8,46E-13	1,09E+02	1,80E+03	1,37E-10
MTS	0,00E+00	4,72E-02	3,41E-04	0,00E+00
Algorithm	F5	F6	F7	-
DewSAcc	3,58E-03	2,30E+00	-1,06E+04	
DMS-PSO	0,00E+00	7,76E+00	-7,51E+03	
EPUS-PSO	5,89E+00	1,89E+01	-6,62E+03	
JDEdynNP-F	3,98E-14	1,47E-11	-1,35E+04	
LSEDA-gl	1,71E-13	4,26E-13	-1,35E+04	
MA-LS-Chains	5,71E-16	3,07E-12	-1,28E+04	
MLCC	4,18E-03	1,06E-12	-1,47E+04	
MTS	0,00E+00	1,24E-11	-1,40E+04	

TABLE IX

COMPARISON RESULTS FOR DIMENSION 1000

REFERENCES

[1] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In *2008 IEEE Congress on Evolutionary Computation*, pages 2032–2039, 2008.

[2] A. Caponio, G.L. Cascella, F. Neri, N. Salvatore, and M. Sumner. A fast adaptive memetic algorithm for off-line and on-line control design of pmsm drivers. *IEEE Transactions on Systems, Man and Cybernetics - Part B, Special Issue on Memetic Algorithms*, 37(1):28–41, 2007.

[3] A. Caponio, F. Neri, and V. Tirronen. Super-fit control adapation in memetic differential evolution frameworks. *Soft Computing-A Fusion of Foundations, Applications*, 2009.

[4] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[5] L. J. Eshelman and J. D. Schaffer. Real-coded Genetic Algorithms in Genetic Algorithms by Preventing Incest. *Foundation of Genetic Algorithms 2*, pages 187–202, 1993.

[6] W. Banzhaf et al., editor. *Optimizing global-local search hybrids*. Morgan Kaufmann, San Mateo, California, 1999.

[7] C. Fernandes and A. Rosa. A Study of non-Random Matching and Varying Population Size in Genetic Algorithm using a Royal Road Function. *Proc. of the 2001 Congress on Evolutionary Computation*, pages 60–66, 2001.

[8] S. García, D. Molina, M. Lozano, and F. Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics. In Press.*, 2008.

[9] W.E. Hart. *Adaptive Global Optimization With Local Search*. PhD thesis, Univ. California, San Diego, CA., 1994.

[10] S.T. Hsieh, T.Y. Sun, C.C. Liu, and S.J. Tsai. Solving large scale global optimization using improved particle swarm optimizer. In *2008 IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2008.

[11] N. Krasnogor and J. Smith. A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issue. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[12] M.W. Shannon Land. *Evolutionary Algorithms with Local Search for Combinational Optimization*. PhD thesis, Univ. California, San Diego, CA., 1998.

[13] C. MacNish and X. Yao. Direction matters in high-dimensional optimisation. In *2008 IEEE Congress on Evolutionary Computation*, pages 2377–2384, 2008.

[14] P. Merz. *Memetic Algorithms for Combinational Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Gesamthochschule Siegen, University of Siegen, Germany, 2000.

[15] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera. Memetic algorithms for continuous optimization based on local search chains. *Evolutionary Computation. In press*, 2009.

[16] P.A. Moscato. On evolution, search, optimization, genetic

algorithms and martial arts: Towards memetic algorithms. Technical report, Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California, 1989.

[17] P.A. Moscato. *Memetic algorithms: a short introduction*, pages 219–234. McGraw-Hill, London, 1999.

[18] H. Mülenbein and D. Schlierkamp-Voosen. Predictive Models for the Breeding Genetic Algorithm in Continuous Parameter Optimization. *Evolutionary Computation*, 1:25–49, 1993.

[19] J.A. Nelder and R. Mead. A simplex method for functions minimizations. *Computer Journal*, 7(4):308–313, 1965.

[20] F. J. Solis and R. J. Wets. Minimization by Random Search Techniques. *Mathematical Operations Research*, 6:19–30, 1981.

[21] G. Syswerda. Uniform Crossover in Genetic Algorithms. In J. David Schaffer, editor, *Proc. of the Thrid Int. Conf. on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers, San Mateo, 1989.

[22] E.G. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, 8:541–564, 2002.

[23] K. Tang. Summary of results on cec'08 competition on large scale global optimization. Technical report, Nature Inspired Computation and Application Lab (NICAL), 2008.

[24] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, and Z. Yang. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Application Laboratory, USTC, China, 2007. `http://nical.ustc.edu.cn/cec08ss.php`.

[25] L.Y. Tseng and C. Chen. Multiple trajectory search for multiobjective optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 3609–3616, 2007.

[26] L.Y. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation*, pages 3057–3064, 2008.

[27] F. van den Bergh and A.P. Engelbrencht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, pages 225–239, 2004.

[28] Y. Wang and B. Li. A restart univariate estimation of distribution algorithm: Sampling under mixed gaussian and lévy probability distribution. In *2008 IEEE Congress on Evolutionary Computation*, pages 3918–3925, 2008.

[29] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *2008 IEEE Congress on Evolutionary Computation*, pages 1663–1670, 2008.

[30] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In *2008 IEEE Congress on Evolutionary Computation*, pages 3719–3726, 2008.

[31] S.Z. Zhao, J.J. Liang, P.N. Suganthan, and M.F. Tasgetiren. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation*, pages 3846–3852, 2008.