WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain CEC IEEE

MA-SW-Chains: Memetic Algorithm Based on Local Search Chains for Large Scale Continuous Global Optimization

Daniel Molina, Manuel Lozano, and Francisco Herrera

***Abstract*— Memetic algorithms are effective algorithms to obtain reliable and 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, requiring more scalable algorithms that, at the same time, could explore better the higher domain space around each solution.**

In this work, we proposed a memetic algorithm, MA-SW-Chains, for large scale global optimization. This algorithm assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. MA-SW-Chains is an adaptation to large scale optimization of a previous algorithm, MA-CMA-Chains, to improve its performance on high-dimensional problems.

Finally, we present the results obtained by our proposal using the benchmark problems defined in the Special Session of *Large Scale Global Optimization* on the IEEE Congress on Evolutionary Computation in 2010.

I. INTRODUCTION

It is well known that the hybridization of *evolutionary algorithms* (EAs) with other techniques can improve the search efficiency [2], [4]. EAs that have been hybridized with local search (LS) techniques are often called *memetic algorithms* (MAs) [11], [15], [16]. 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 [1], [9], [10], [18], [19].

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 [21]. Thus, the ability of being scalable for high-dimensional problems becomes an essential requirement for modern optimization algorithm approaches.

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, and a high dimension increases the domain search and the region to explore. Thus, this large area to explore suggests the convenience to search around them with high intensity.

In a previous work, we have defined a MA for continuous optimization, MA-CMA-Chains, specifically designed to adapt the LS intensity, exploiting with higher intensity the most promising individuals [12]. 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. MA-CMA-Chains achieved very good results for continuous optimization for medium scale problems [12].

The original proposal was not applied for large scale optimization because the original LS method used, CMA-ES [8], is not able to tackle effectively problems when their dimensionality is increased [7], [14]. However, the algorithm is scalable when a different LS method is applied [13].

In this work, we present an algorithm, MA-SW-Chains, that combines the ideas of MA-CMA-ES with a scalable LS algorithm, the classic *Solis Wets'* algorithm. To show its behavior with large scale problems, we have applied this algorithm to the specific test suite proposed in the *Special Session on Large Scale Continuous Global Optimization* in the 2010 IEEE Congress on Evolutionary Computation.

This paper is set up as follows. In Section II, we describe MA-SW-Chains, the proposed MA. In Section III, we detail the test suite and the followed experimental conditions. In Section IV, we present the empirical study, comparing also the results of MA-SW-Chains with other algorithms. Finally, conclusions and future work are presented in Section V.

II. MACOS BASED ON LS CHAINS

In this section, we propose a MACO, MA-SW-Chains, that uses the model proposed in [12], using the Solis Wets' algorithm as its LS method. MA-SW-Chains is a steady-state MA model that employs the concept of *LS chain* to adjust the LS intensity assigned to the 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

Daniel Molina Cabrera is with the Department of Computer Languages and Systems, University of Cadiz, Cadiz, Spain (email: daniel.molina@uca.es).

M. Lozano and F. Herrera are with Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain (e-mails:{lozano,herrera}@decsai.ugr.es.).

may adaptively fit its strategy parameters to the particular features of these zones.

In Section A, we introduce the foundations of steady-state MAs. In Section B, we introduce the *Solis Wets'* method. In Section C, we explain the concept of LS *chain*. Finally, in Section D, we give an overview of the proposed MA-SW-Chains.

A. Steady-State MAs

In steady-state genetic algorithms (SSGAs) [22] usually only one 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 Genetic Algorithms (GAs), *steady-state MAs* (steady-state GAs plus LS method) 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.

The SSGA applied was specifically designed to promote high population diversity levels by means of the combination of the $BLX - \alpha$ crossover operator [3] with a high value for its associated parameter ($\alpha = 0.5$) and the *negative assortative mating* strategy [5]. Diversity is favoured as well by means of the BGA mutation operator [17].

B. Solis and Wets' Algorithm

The classic *Solis and Wets'* algorithm [20] 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 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 (*maxSuccesses*) in a row, ρ is increased to move quicker. After several failures (*maxFailures*) in a row, ρ is decreased to focus the search. Additionally, a bias term is included to put the search momentum in directions that yield success. The size search is defined by its strategy parameter ρ, and it can be adapted very quickly. In our experiments we have applied *maxSuccesses*=5, *maxFailures*=3, and initial $\rho = 0.2$. More details about this procedure may be found in [20].

C. Local Search Chains

In steady-state MAs, individuals may reside in the population during a long time. This circumstance allows these individuals to become starting points of subsequent LS invocations. In [12], 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. In the case of *Solis Wets'* algorithm, the following parameters are stored: success and failure numbers, and the parameters *bias* and ρ.

D. MA-SW-Chains: A MACO that Handles LS Chains

In this section, we introduce MA-SW-Chains, which handles LS chains (see Figure 1). It has the following main features:

1) It is a steady-state MA model.
2) It applies the *Solis Wets'* algorithm as its LS method.
3) Results could differ greatly in function of the ratio of evaluations number invested in each component (EA for exploration and LS method to exploitation). We define $r_{L/G}$ as the ratio of evaluations used by the LS method and the total evaluations number. Our algorithm ensures than a fixed and predetermined $r_{L/G}$ is always kept, which has strong influence on the final behavior. Without this strategy, the application of the LS method may induce MA-SW-Chains to prefer super exploitation.
4) 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).

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*

```
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.     Initialize the LS operator with the LS state stored together with $c_{LS}$.
7. Else
8.     Initialize the LS operator with the default LS state.
9. Apply Solis Wets' 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

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 $r_{L/G}$, n_{frec} may be calculated using the equation $n_{frec} = I_{str}\frac{1-r_{L/G}}{r_{L/G}}$, where I_{str} is the LS intensity stretch (previous subsection), 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 optimized by the LS algorithm, or
    b) They previously underwent LS, obtaining a fitness function improvement greater than δ^{min}_{LS} (a parameter of our algorithm).
2) If $|S_{LS}| \neq 0$, then apply the *Solis Wets'* 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 δ^{min}_{LS} threshold.

Parameter setting: For the experiments, we have used the same parameter values applied in [12]. The SSGA 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. We have considered $I_{str} = 500$ and $r_{L/G} = 0.5$. In this case, $\delta^{LS}_{min} = 0$ because in the functions there is no fitness threshold value.

III. Experiments

The proposal has been tested on 20 scalable optimization problems, defined for the organizers of the Special Session on *Large Scale in Global Optimization*, presented in the *IEEE Congress on Evolutionary Computation 2010.*

This test suite is composed by four types of high–dimensional problems:

1) Separable functions:
    a) F_1 : Shifted Elliptic Function.
    b) F_2 : Shifted Rastrigins Function.
    c) F_3 : Shifted Ackleys Function.
2) Partially-separable functions, in which a small number of variables are dependent while all the remaining ones are independent (m=50):
    a) F_4 :Single-group Shifted and m-rotated Elliptic Function.
    b) F_5 : Single-group Shifted and m-rotated Rastrigins Function.
    c) F_6 : Single-group Shifted and m-rotated Ackleys Function.
    d) F_7 : Single-group Shifted and m-dimensional Schwefels Problem 1.2.
    e) F_8 : Single-group Shifted and m-dimensional Rosenbrocks Function.
3) Partially-separable functions that consist of multiple independent subcomponents, each of which is m-non-separable (m=50):
    a) F_9 : $\frac{D}{2m}$ -group Shifted and m-rotated Elliptic Function.
    b) F_{10} : $\frac{D}{2m}$ -group Shifted and m-rotated Rastrigins Function.
    c) F_{11} : $\frac{D}{2m}$ -group Shifted and m-rotated Ackleys Function.
    d) F_{12} : $\frac{D}{2m}$ -group Shifted and m-dimensional Schwefels Problem 1.2.
    e) F_{13} : $\frac{D}{2m}$ -group Shifted and m-dimensional Rosenbrocks Function.
    f) F_{14} : $\frac{D}{m}$ -group Shifted and m-rotated Elliptic Function.
    g) F_{15} : $\frac{D}{m}$ -group Shifted and m-rotated Rastrigin's Function.
    h) F_{16} : $\frac{D}{m}$ -group Shifted and m-rotated Ackley's Function.
    i) F_{17} : $\frac{D}{m}$ -group Shifted and m-rotated Schwefel's Problem 1.2 Function.
    j) F_{18} : $\frac{D}{m}$ -group Shifted and m-rotated Rosenbrock's Function.
4) Fully-nonseparable functions:
    a) F_{19} : Shifted Schwefels Problem 1.2.
    b) F_{20} : Shifted Rosenbrocks Function.

We are going to analyze the results of our proposal in each one of these categories.

The experiments have been carried out following the instructions indicated in the documents associated to the benchmark functions, to be able to compare our proposal with other algorithms presented in the same special 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=1000.
3) The maximum number of fitness evaluations (FEs) is $3 \cdot 10^6$. Each run stops when this maximum number of evaluations is achieved.
4) The error values have been recorded with FEs=$1.2 \cdot 10^5$, $6 \cdot 10^5$, and $3 \cdot 10^6$.

IV. Main Results

In this section, we analyze the results of our algorithm on the test suite described in the previous section.

First, in section A, we observe the convergence curves of MA-SW-Chains with several functions. In Section B, we describe the behavior of MA-SW-Chains for each function category. Finally, in section C, we compare MA-SW-Chains with the results of other algorithms presented by the organizers: DECC-CG [23], and MLCC [24], showing the convenience of our proposal.

A. Convergence Results

In this section, we are going to provide convergence curves of MA-SW-Chains on the following eight selected functions: F_2, F_5, F_8, F_{10}, F_{13}, F_{15}, F_{18}, and F_{20}. For each function, the single convergence curve has been plotted using the average results over all 25 runs.

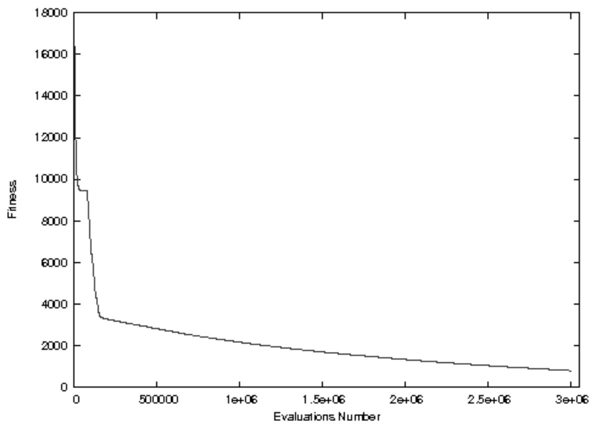

Fig. 2. Mean results for function F_2

Figures 2-9 show the convergence curve of functions F_2-F_{20}, respectively. We can observe the following:

1) In function F_2, we observe than MA-SW-Chains, after a quickly convergence ratio, continues with a smaller ratio, but the convergence has not finished, it is not at a standstill.
2) In function F_5, we observe than the best fitness is improved by steps.
3) In function F_8, we observe regions with different convergence speed, and from evaluation 10^6 until $3 \cdot 10^6$ there are improvements, but the convergence curve is almost horizontal.
4) In function F_{10}, we observe that there are improvements until the maximum FEs is reached.
5) In function F_{13}, there is a good convergence curve until

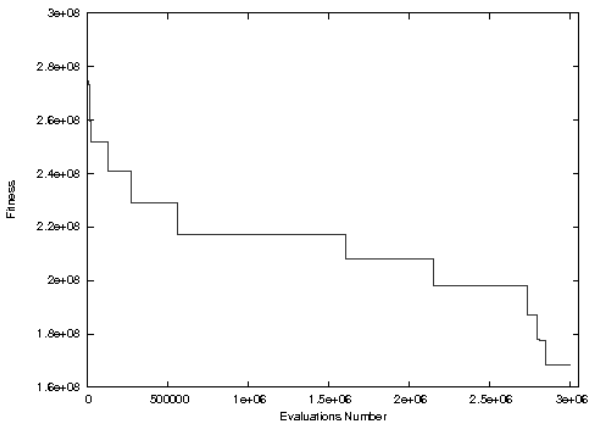

Fig. 3. Mean results for function F_5

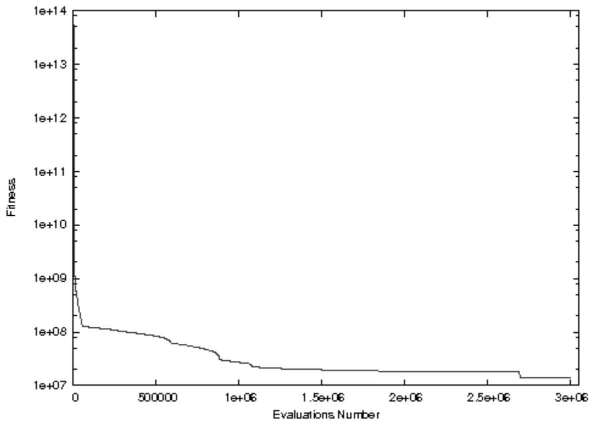

Fig. 4. Mean results for function F_8

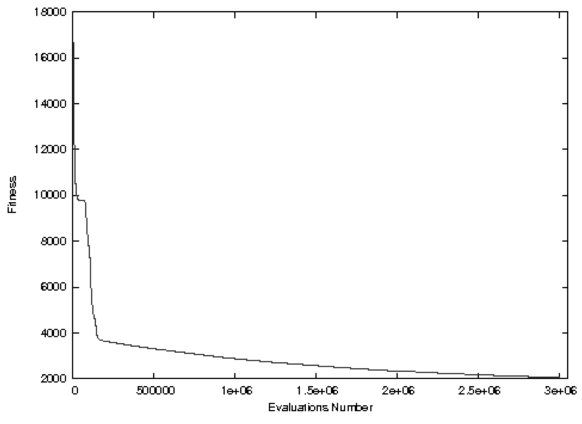

Fig. 5. Mean results for function F_{10}

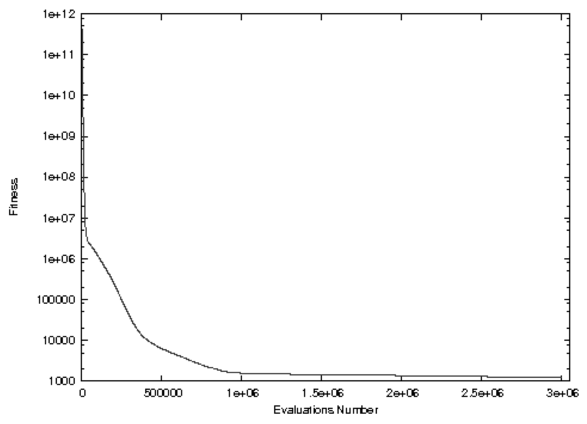

Fig. 6. Mean results for function F_{13}

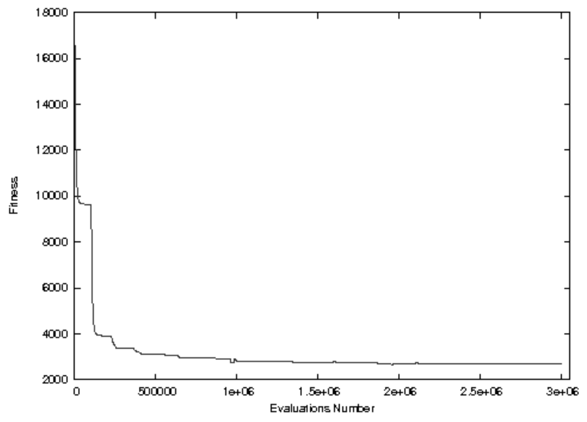

Fig. 7. Mean results for function F_{15}

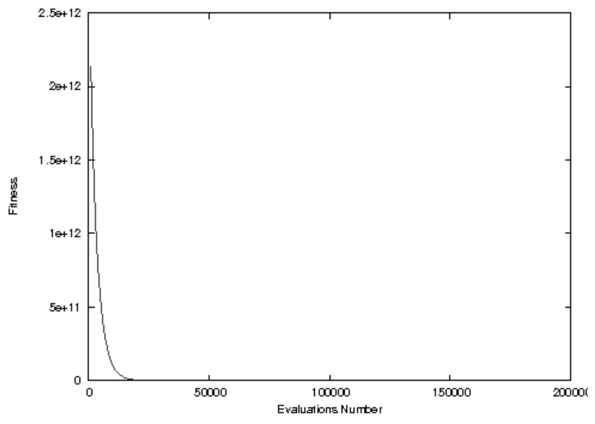

Fig. 8. Mean results for function F_{18}

the evaluation 10^6, and then there are only very few improvements.

6) In functions F_{15}, after a quick improvement, the fitness value is almost stabilized during the majority of the evaluations.
7) For functions F_{18} and F_{20}, the convergence curve has a very steep slope.

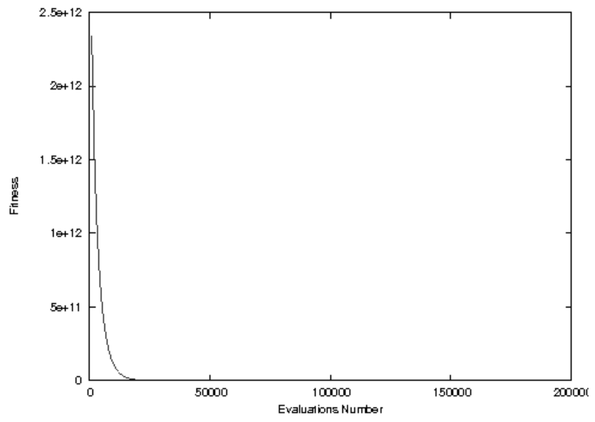

Fig. 9. Mean results for function F_{20}

B. Results of our algorithm

Table I contains the results obtained by MA-SW-Chains. We can observe the following characteristics:

- In the majority of functions, differences between mean and median are small. That implies the MA-SW-Chains is a rather robust algorithm.
- The results of functions that differs only in the number of variables with dependences, are very similar, showing than MA-SW-Chains is robust against dependences between variables.
- In many functions, results with $FEs = 3 \cdot 10^6$ are significantly better than with $FEs = 6 \cdot 10^5$. Thus, MA-SW-Chains achieves improvements until the maximum FEs is achieved.
- Results in non-separable functions are very good, and when the number of variable dependences is increased, as F_9 and F_{14}, or F_{13} and F_{18}, results are worse, but not in a significant way.

C. Comparisons with other algorithms

In this section we compare the mean results obtained MA-SW-Chains with the ones obtained by DECC-CG [23], and MLCC [24].

First, we compare them using the Wilcoxon's test, described in [6]. Table II shows the results of Wilcoxon's test, using p-value=0.05.

TABLE II
MA-SW-CHAINS VERSUS DECC-CG AND MLCC (WILCOXON'S TEST WITH p-VALUE = 0.05)

Algorithm	$R+$	$R-$	**Critical value**	**Sig. differences?**
DECC-CG	203	**7**	52	Yes
MLCC	204	**4**	52	Yes

From Table II we can observe than our proposal achieves the best results and the difference is statistically relevant.

TABLE I
EXPERIMENTAL RESULTS WITH MA-SW-CHAINS

		F_1	F_2	F_3	F_4	F_5	F_6	F_7
FEs = 1.2e5	Best	2.15e+07	3.32e+03	1.13e+01	1.22e+12	9.35e+07	2.02e+01	4.54e+06
	Median	2.76e+07	3.75e+03	1.15e+01	2.04e+12	2.64e+08	2.08e+01	4.91e+06
	Worst	3.51e+07	1.00e+04	1.22e+01	3.35e+12	3.42e+08	1.16e+06	5.71e+06
	Mean	2.83e+07	5.09e+03	1.16e+01	2.12e+12	2.52e+08	8.14e+04	4.90e+06
	Std	3.06e+06	2.38e+03	2.68e-01	6.21e+11	6.49e+07	2.84e+05	2.59e+05
FEs = 6.0e5	Best	8.52e+02	2.36e+03	3.44e+00	4.29e+11	3.68e+07	3.61e+00	6.33e+04
	Median	1.55e+03	2.68e+03	3.83e+00	5.75e+11	2.59e+08	1.78e+01	7.78e+05
	Worst	7.28e+03	2.97e+03	4.60e+00	7.42e+11	3.24e+08	1.16e+06	4.61e+06
	Mean	2.24e+03	2.67e+03	3.84e+00	5.79e+11	2.17e+08	8.14e+04	8.35e+05
	Std	1.71e+03	1.63e+02	2.13e-01	6.46e+10	8.56e+07	2.84e+05	9.08e+05
FEs = 3.0e6	Best	3.18e-15	7.04e+02	3.34e-13	3.04e+11	2.89e+07	8.13e-07	3.35e-03
	Median	1.50e-14	7.90e+02	6.11e-13	3.54e+11	2.31e+08	1.60e+00	9.04e+01
	Worst	8.15e-14	9.37e+02	1.58e-12	3.97e+11	2.90e+08	1.16e+06	2.68e+02
	Mean	2.10e-14	8.10e+02	7.28e-13	3.53e+11	1.68e+08	8.14e+04	1.03e+02
	Std	1.99e-14	5.88e+01	3.40e-13	3.12e+10	1.04e+08	2.84e+05	8.70e+01
		F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
FEs = 1.2e5	Best	3.30e+07	4.48e+08	3.62e+03	5.01e+01	2.20e+05	7.64e+05	8.16e+08
	Median	4.17e+07	5.60e+08	4.15e+03	6.41e+01	2.40e+05	9.04e+05	8.81e+08
	Worst	8.55e+08	6.45e+08	1.00e+04	7.13e+01	2.62e+05	1.11e+06	1.04e+09
	Mean	1.21e+08	5.54e+08	5.12e+03	6.31e+01	2.40e+05	9.13e+05	8.95e+08
	Std	2.11e+08	5.20e+07	2.20e+03	5.53e+00	1.26e+04	8.09e+04	6.60e+07
FEs = 6.0e5	Best	3.42e+06	6.93e+07	2.79e+03	2.77e+01	1.39e+03	1.08e+03	1.51e+08
	Median	1.90e+07	8.08e+07	3.25e+03	3.79e+01	1.64e+03	3.06e+03	1.70e+08
	Worst	6.11e+08	1.00e+08	3.54e+03	5.15e+01	1.91e+03	1.07e+04	1.95e+08
	Mean	6.13e+07	8.18e+07	3.22e+03	3.83e+01	1.63e+03	4.34e+03	1.69e+08
	Std	1.27e+08	8.36e+06	1.85e+02	7.23e+00	1.53e+02	3.21e+03	1.17e+07
FEs = 3.0e6	Best	1.54e+06	1.19e+07	1.81e+03	2.74e+01	2.65e-06	3.86e+02	2.79e+07
	Median	3.43e+06	1.40e+07	2.07e+03	3.75e+01	3.50e-06	1.07e+03	3.09e+07
	Worst	1.80e+08	1.62e+07	2.28e+03	5.11e+01	4.98e-06	2.92e+03	3.67e+07
	Mean	1.41e+07	1.41e+07	2.07e+03	3.80e+01	3.62e-06	1.25e+03	3.11e+07
	Std	3.68e+07	1.15e+06	1.44e+02	7.35e+00	5.92e-07	5.72e+02	1.93e+06
		F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	-
FEs = 1.2e5	Best	3.94e+03	2.01e+02	5.80e+05	2.22e+04	3.23e+06	2.01e+03	
	Median	4.29e+03	2.12e+02	6.78e+05	5.18e+04	3.63e+06	2.22e+03	
	Worst	9.63e+03	2.31e+02	7.50e+05	8.33e+04	4.05e+06	4.69e+03	
	Mean	4.83e+03	2.13e+02	6.78e+05	5.14e+04	3.63e+06	2.43e+03	
	Std	1.51e+03	9.19e+00	3.52e+04	1.64e+04	1.94e+05	5.43e+02	
FEs = 6.0e5	Best	2.95e+03	8.51e+01	3.59e+04	1.80e+03	1.29e+06	1.02e+03	
	Median	3.19e+03	9.71e+01	4.29e+04	3.89e+03	1.42e+06	1.18e+03	
	Worst	3.45e+03	1.26e+02	5.01e+04	1.61e+04	1.58e+06	1.65e+03	
	Mean	3.19e+03	1.02e+02	4.31e+04	5.53e+03	1.41e+06	1.21e+03	
	Std	1.46e+02	1.42e+01	3.42e+03	3.94e+03	7.44e+04	1.42e+02	
FEs = 3.0e6	Best	2.56e+03	8.51e+01	1.04e+00	7.83e+02	2.49e+05	9.25e+02	
	Median	2.72e+03	9.44e+01	1.26e+00	1.19e+03	2.85e+05	1.06e+03	
	Worst	2.96e+03	1.24e+02	1.63e+00	2.55e+03	3.32e+05	1.21e+03	
	Mean	2.74e+03	9.98e+01	1.24e+00	1.30e+03	2.85e+05	1.07e+03	
	Std	1.22e+02	1.40e+01	1.25e-01	4.36e+02	1.78e+04	7.29e+01	

TABLE III
EXPERIMENTAL RESULTS WITH DIFFERENT ALGORITHMS, FES=3E6

		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	Best	1.63e-07	1.25e+03	1.20e+00	7.78e+12	1.50e+08	3.89e+06	4.26e+07
	Median	2.86e-07	1.31e+03	1.39e+00	1.51e+13	2.38e+08	4.80e+06	1.07e+08
DECC-CG	Worst	4.84e-07	1.40e+03	1.68e+00	2.65e+13	4.12e+08	7.73e+06	6.23e+08
	Mean	2.93e-07	1.31e+03	1.39e+00	1.70e+13	2.63e+08	4.96e+06	1.63e+08
	Std	8.62e-08	3.26e+01	9.73e-02	5.37e+12	8.44e+07	8.02e+05	1.37e+08
	Best	0.00e+00	1.73e-11	1.28e-13	4.27e+12	2.15e+08	5.85e+06	4.16e+04
	Median	**0.00e+00**	**6.43e-11**	**1.46e-13**	1.03e+13	3.92e+08	1.95e+07	5.15e+05
MLCC	Worst	3 .83e-26	1.09e+01	1.86e-11	1.62e+13	4.87e+08	1.98e+07	2.78e+06
	Mean	**1.53e-27**	**5.57e-01**	9.88e-13	9.61e+12	3.84e+08	1.62e+07	6.89e+05
	Std	7.66e-27	2.21e+00	3.70e-12	3.43e+12	6.93e+07	4.97e+06	7.37e+05
	Best	3.18e-15	7.04e+02	3.34e-13	3.04e+11	2.89e+07	8.13e-07	3.35e-03
	Median	1.50e-14	7.90e+02	6.11e-13	**3.54e+11**	**2.31e+08**	**1.60e+00**	**9.04e+01**
MA-SW-Chains	Worst	8.15e-14	9.37e+02	1.58e-12	3.97e+11	2.90e+08	1.16e+06	2.68e+02
	Mean	2.10e-14	8.10e+02	**7.28e-13**	**3.53e+11**	**1.68e+08**	**8.14e+04**	**1.03e+02**
	Std	1.99e-14	5.88e+01	3.40e-13	3.12e+10	1.04e+08	2.84e+05	8.70e+01
		F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
	Best	6.37e+06	2.66e+08	1.03e+04	2.06e+01	7.78e+04	1.78e+03	6.96e+08
	Median	6.70e+07	3.18e+08	1.07e+04	**2.33e+01**	8.87e+04	3.00e+03	8.07e+08
DECC-CG	Worst	9.22e+07	3.87e+08	1.17e+04	2.79e+01	1.07e+05	1.66e+04	9.06e+08
	Mean	6.44e+07	3.21e+08	1.06e+04	**2.34e+01**	8.93e+04	5.12e+03	8.08e+08
	Std	2.89e+07	3.38e+07	2.95e+02	1.78e+00	6.87e+03	3.95e+03	6.07e+07
	Best	4.51e+04	8.96e+07	2.52e+03	1.96e+02	2.42e+04	1.01e+03	2.62e+08
	Median	4.67e+07	1.24e+08	3.16e+03	1.98e+02	3.47e+04	1.91e+03	3.16e+08
MLCC	Worst	9.06e+07	1.46e+08	5.90e+03	1.98e+02	4.25e+04	3.47e+03	3.77e+08
	Mean	4.38e+07	1.23e+08	3.43e+03	1.98e+02	3.49e+04	2.08e+03	3.16e+08
	Std	3.45e+07	1.33e+07	8.72e+02	6.98e-01	4.92e+03	7.27e+02	2.77e+07
	Best	1.54e+06	1.19e+07	1.81e+03	2.74e+01	2.65e-06	3.86e+02	2.79e+07
	Median	**3.43e+06**	**1.40e+07**	**2.07e+03**	3.75e+01	**3.50e-06**	**1.07e+03**	**3.09e+07**
MA-SW-Chains	Worst	1.80e+08	1.62e+07	2.28e+03	5.11e+01	4.98e-06	2.92e+03	3.67e+07
	Mean	**1.41e+07**	**1.41e+07**	**2.07e+03**	3.80e+01	**3.62e-06**	**1.25e+03**	**3.11e+07**
	Std	3.68e+07	1.15e+06	1.44e+02	7.35e+00	5.92e-07	5.72e+02	1.93e+06
		F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	-
	Best	1.09e+04	5.97e+01	2.50e+05	5.61e+03	1.02e+06	3.59e+03	
	Median	1.18e+04	**7.51e+01**	2.89e+05	2.30e+04	1.11e+06	3.98e+03	
DECC-CG	Worst	1.39e+04	9.24e+01	3.26e+05	4.71e+04	1.20e+06	5.32e+03	
	Mean	1.22e+04	**7.66e+01**	2.87e+05	2.46e+04	1.11e+06	4.06e+03	
	Std	8.97e+02	8.14e+00	1.98e+04	1.05e+04	5.15e+04	3.66e+02	
	Best	5.30e+03	2.08e+02	1.38e+05	2.51e+03	1.21e+06	1.70e+03	
	Median	6.89e+03	3.95e+02	1.59e+05	4.17e+03	1.36e+06	2.04e+03	
MLCC	Worst	1.04e+04	3.97e+02	1.86e+05	1.62e+04	1.54e+06	2.34e+03	
	Mean	7.11e+03	3.76e+02	1.59e+05	7.09e+03	1.36e+06	2.05e+03	
	Std	1.34e+03	4.71e+01	1.43e+04	4.77e+03	7.35e+04	1.80e+02	
	Best	2.56e+03	8.51e+01	1.04e+00	7.83e+02	2.49e+05	9.25e+02	
	Median	**2.72e+03**	9.44e+01	**1.26e+00**	**1.19e+03**	**2.85e+05**	**1.06e+03**	
MA-SW-Chains	Worst	2.96e+03	1.24e+02	1.63e+00	2.55e+03	3.32e+05	1.21e+03	
	Mean	**2.74e+03**	9.98e+01	**1.24e+00**	**1.30e+03**	**2.85e+05**	**1.07e+03**	
	Std	1.22e+02	1.40e+01	1.25e-01	4.36e+02	1.78e+04	7.29e+01	

To analyze the main differences for functions, we are going to compare directly the results obtained by each algorithm, because the number of functions for each category is not high enough to apply a statistical test.

Table III shows the final results of each algorithm. For remarking the best algorithm, we have put in bold the best median and mean values for each function. From Table III we have obtained the following conclusions:

- MA-SW-Chains achieves the best mean in 16 functions. Only for F_1, F_2, F_{11}, F_{16} our proposal does not achieve the best result; F_1 and F_2, because MLCC is specialized in separable functions like them; and F_{11} and F_{16}, because DECC-CG is better in the Ackleys' function.
- MA-SW-Chains obtains the best results for all the categories of functions, except for the easier group, separable functions. This is very important, because it is clearly the most adequate algorithm for problems with a non-separable group of functions.
- The great differences of results in Functions F_7 and F_{12} between MA-SW-Chains and the others is clearly remarkable.
- For many functions, the worst result obtained by MA-SW-Chains is better than the best result obtained by the other algorithms.

V. Conclusions

In this work, we have presented MA-SW-Chains. It uses the idea of the MACO proposed in [12], applying the idea of LS chaining, with the scalable LS method *Solis Wets*. We have carried out an empirical study to analyze how scalable is the proposal for large scale problems, following the test suite proposed by the organizers of the Special Session of *Large Scale Global Optimization*, on the IEEE Congress on Evolutionary Computation in 2010. Experiments show that the proposal, MA-SW-Chains, obtain good results in the majority of functions, in particular, on the more complex functions, combining separable and non-separable variables.

Acknowledgment

This work was supported by Research Projects TIN2008-05854 and P08-TIC-4173.

References

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

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

[3] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithm and interval schemata. *Foundation of Genetic Algorithms*, pages 187–202, 1993.

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

[5] 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.

[6] 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*, 15:617–644, 2009.

[7] N. Hansen. Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pages 2389–2396, 2009.

[8] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceeding of the IEEE International Conference on Evolutionary Computation (ICEC '96)*, pages 312–317, 1996.

[9] 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.

[10] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(2):273–302, 2004.

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

[12] D. Molina, M. Lozano, C. García-Martínez, and F. Herrera. Memetic algorithms for continuous optimization based on local search chains. *Evolutionary Computation. In press*, 2010.

[13] D. Molina, M. Lozano, and F. Herrera. Memetic algorithm with local search chaining for continuous optimization problems: A scalability test. In *ISDA '09: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 1068–1073, 2009.

[14] D. Molina, M. Lozano, and F. Herrera. Memetic algorithm with local search chaining for large scale continuous optimization problems. In *Proc. of the 2009 IEEE Congress on Evolutionary Computation*, pages 830–837, 2009.

[15] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms. Technical report, Technical Report Caltech Concurrent Computation Program Report 826, Caltech, Pasadena, California, 1989.

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

[17] H. Mülenbein and D. Schlierkamp-Voosen. Predictive models for the breeding genetic algorithm in continuous parameter optimization. *Evolutionary Computation*, 1:25–49, 1993.

[18] Q. H. Nguyen, Y. S. Ong, and M. H. Lim. A probabilistic memetic framework. *IEEE Transactions on Evolutionary Computation*, 13(3):604–623, 2009.

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

[20] F. J. Solis and R. J. Wets. Minimization by random search techniques. *Mathematical Operations Research*, 6:19–30, 1981.

[21] F. van den Bergh and A. Engelbrencht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, (3):225–239, 2004.

[22] D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 116–121, 1989.

[23] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008.

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