Memetic Algorithm with Local Search Chaining for Continuous Optimization Problems: A Scalability Test

Daniel Molina
Department of Computer Languages and Systems
University of Cadiz
Cadiz, Spain
daniel.molina@uca.es

Manuel Lozano and Francisco Herrera
Department of Computer Science and Artificial Inteligence
University of Granada
Granada, Spain
lozano@decsai.ugr.es, herrera@decsai.ugr.es

Abstract

Memetic algorithms arise as very effective algorithms to obtain reliable and high accurate solutions for complex continuous optimization problems. Nowadays, higher dimensional optimization problems are an interesting field of research, that introduces new problems for the optimization process, making recommendable to test the scalable capacities of optimization algorithms. In particular, in memetic algorithms, a higher dimensionality increases the domain space around each solution, requiring that the local search method must be applied with a high intensity.

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 has obtained good results in continuous optimization problems and we study whether, using this intensity adaptation mechanism with the scalable LS method MTS-LS2*, the algorithm is scalable enough for being a good algorithm for medium and high-dimensional problems. Experiments are carried out to test the ability of being scalable, and results obtained show that the proposal is scalable in many of the functions, scalable and non-scalable, of the benchmark used.*

1 Introduction

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

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 [6, 14, 7].

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

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 [8]. 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. This MA using LS chaining obtains very good results for continuous optimization problems [8].

Unfortunately, the original proposal was not scalable because the original LS method used, C-MA-ES [5], which is not able to tackle effectively problems when their dimensionality is increased [9]. There are other LS methods, like *MTS-LS2* [16], that explores the neighborhood of a solution making changes to a subset of variables. These LS methods are more scalables when the dimensionality is increased, but sometimes they are not adequated for tackle non-separable problems. However, that disadvantage could be reduced when they are used in MAs. Previously, we have applied *MTS-LS2* as our LS method, obtaining good results

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.143

in problems with high dimensionality values [9].

In this work, we want to extend this study to test the scalability capacities of the algorithm using the LS method *MTS-LS2*, and check if the proposed algorithm is scalable and obtains good results. Experiments are carried out to test the ability of our algorithm of being scalable for separable and non-separable problems.

This work is set up as follows. In Section 2, we describe *MTS-LS2*, the LS method used. In Section 3, we describe the proposed MA. In Section 4, we present the empirical study. Finally, conclusions and future work are presented in Section 5.

2 MTS-LS algorithm

Several classical LS methods are not scalable, thus they are not able to obtain good results in large scale optimization problems [13]. One option to solve this problem is using a specifically designed LS method to these type of algorithms. In a previous work, we have compared several classical LS methods and other specifically designed to tackle the scalability problem [9]. In that work, one of the LS methods, *MTS-LS2* [16], was clearly more scalable than the other ones, thus we use it in this paper.

MTS-LS2 is a hill-climbing algorithm that explores changing a quartier of dimensions each time.

In each step of this algorithm, each variable of the individual to be changed is increased by a certain value (called SR in the paper) to check if the objective function value is improved. Only a quarter of them (randomly selected by each application of the LS) are modified.

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.

The addition factor (SR) is multiplied by (-1) the 50% of times to avoid focussing always the search to the same direction.

Although it could be expected that an LS method that explores dimension by dimension is only adequate to separable problems, in combination with an EA that explore all variables at the same time, this problem is reduced, like it is observed in Section 4.

3 MACOs Based on LS Chains

In this section, we describe a MACO approach proposed in [8] that employs continuous LS methods as LS operators. It is a steady-state MA model that employs the concept of *LS chain* to adjust the LS intensity assigned to the intense continuous LS method. In particular, this MACO handles LS chains, throughout the evolution, with the objective of allowing the continuous LS algorithm to act more intensely in the most promising areas represented in the EA population. In this way, the continuous LS method may adaptively fit its strategy parameters to the particular features of these zones.

In Section 3.1, we introduce the foundations of steady-state MAs. In Section 3.2, we explain the concept of LS *chain*. Finally, in Section 3.3, we give an overview of the MACO approach presented in [8], that handles LS chains with the objective of applying the LS method with an intensity value in function of how promising is each solution.

3.1 Steady-State MAs

In steady-state GAs [17] 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, *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 [2] with a high value for its associated parameter ($\alpha = 0.5$) and the *negative assortative mating* strategy [3]. Diversity is favored as well by means of the BGA mutation operator [12].

3.2 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 [8], 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 *MTS-LS2* algorithm, the SR parameter value is stored.

3.3 A MACO Model that Handles LS Chains

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

1. It is a steady-state MA model.

2. It ensures that a fixed and predetermined local/global search ratio is always kept. With this policy, we easily stabilise this ratio, which has a strong influence on the final MACO 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).

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

```
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 MTS-LS2 method 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.
```

Figure 1. Pseudocode algorithm for the proposed MACO model

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

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

   (a) They have never been optimized by the intense continuous LS algorithm, or

   (b) They previously underwent LS, obtaining a fitness function improvement greater than δ^{min}_{LS} (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 δ_{LS}^{min} threshold.

Parameter setting For the experiments, we have used the same parameter values applied in [8] which obtained good results. 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 $r_{L/G} = 0.8$. In this case, $\delta_{min}^{LS} = 0$ because in the functions there is no fitness threshold value.

4 Experiments

The proposal has been tested on 11 scalable optimization problems, defined for the organizers of the Workshop *Evolutionary Algorithms and Other Metaheuristics for Continuous Optimization Problems — A Scalability Test*, of the *9th International Conference on Intelligence Systems Design and Applications, ISDA'2009*.

Functions $f_1, \ldots, f_6$ have been defined in [15], and functions $f_7, \ldots, f_{11}$ have been defined in [4]. Table 1 shows their names, bounds, and optimum values. In the following, we describe several properties of the functions that we consider interesting. For more detail, the above references can be it could be consulted the .

- f_1 and f_2 are unimodals, and $f_3 \ldots f_{11}$ are multimodals. Thus, the majority of functions have many optima.
- Functions f_1, f_4 and f_6 are strictly separable. In other words, only 3 of the eleven functions are strictly separable. That is specially interesting to analyse if our proposal obtains good results in non-separable functions.

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 the other algorithms presented in the workshop. 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=50, 100, 200 and 500.
3. The maximum number of fitness evaluations is $5,000 \cdot D$. Each run stops when this maximum number of evaluations is achieved.

Table 1. Benchmark's Test problems

	Name	*Intervals*	f^*
f_1	*Shifted Sphere*	[-100, 100]	-450
f_2	Shifted Schwefel 2.21	[-100, 100]	-450
f_3	Shifted Rosenbrock	[-100, 100]	390
f_4	*Shifted Rastrigin*	[-5, 5]	-330
f_5	Shifted Griewank	[-600, 600]	-180
f_6	*Shifted Ackley*	[-32, 32]	-140
f_7	Schwefel 2.22	[-10, 10]	0
f_8	Schwefel 1.2	[-65.536, 65.536]	0
f_9	Extended f_{10}	[-100, 100]	0
f_{10}	Bohachevsky	[-15, 15]	0
f_{11}	Schaffer	[-100, 100]	0

Table 2. Mean Error for function and dimension

Fun	Dim=50	Dim=100	Dim=200	Dim=500
f_1	5,23E-14	5,68E-14	1,14E-13	2,84E-13
f_2	9,87E-01	1,16E+01	3,26E+01	7,00E+01
f_3	1,06E+02	8,88E+02	3,47E+02	7,86E+02
f_4	5,46E-14	6,59E-14	7,96E-02	1,07E+00
f_5	2,27E-14	3,94E-04	2,74E-03	1,44E-13
f_6	5,12E-14	7,84E-14	1,46E-13	3,33E-13
f_7	1,07E-75	6,11E-80	1,17E-75	1,27E-58
f_8	1,24E+00	2,86E+02	2,97E+03	6,84E+04
f_9	5,13E-02	1,77E-01	2,23E-01	3,47E+00
f_{10}	2,45E-99	1,24E-51	3,01E-65	7,04E-45
f_{11}	1,60E-01	4,11E-01	9,95E-01	1,61E+01

Table 2 shows the average mean for each function and dimension obtained by our proposal. We are going to analyse the results:

- In the half of functions, our algorithm achieves an average error lower than 10^{-12} in the majority of dimensions, and in general our proposal achieve very good results.
- In separable functions, f_1, f_4, and f_6, we obtain very good results, as it was expected. But as it is said in the previous line, the good results are not limited to this type of functions, several non-separable functions are improved meaningfully.
- Results of our algorithm are not good enough for three functions, f_2, f_3 and f_8. To know the reason, we have to see their mathematical expressions. Function f_2 is the maximum of the absolute value of the variables, thus an exploitation that not consider many variables at the same time will have problems to optimize this function. In function f_3, there is a very narrow valley from local optimum to global optimum, and that appears to be a disadvantage to be optimized with our proposal. Function f_8, on the other side, is a function that is rather special, because the influence of variable is increased in function of its position, higher for lower position. This behaviour makes more difficult the optimization using our algorithm, because the variables in lower positions are not always consider in the exploitation.
- Considering the scalability question, we can observe that our algorithm tackle successfully the scalability issue for the majority of functions. In many functions, although results are getting worse when dimensionality is increased, they have the same order of magnitude during several dimension values. The functions with the worse behaviour are function f_8, that is not improved enough by the algorithm for any dimension value, and f_4.

5 Conclusions

In this work, we have shown the adaptation of MACO proposed in [8], that apply the idea of LS chaining, with the scalable LS method *MTS-LS2*. We have carried out an empirical study to analyse how scalable is the proposal for large scale problems. Experiments show that the algorithm obtain good results in the majority of functions. Our proposal, nevertheless, have several problems to optimize several extremely non-separable functions. In future works, we are going to modify the algorithm to obtain good results also with this type of functions.

Acknowledgement

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

References

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

[2] 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.

[3] 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.

[4] C. Garcia, M. Lozano, and D. Molina. Parallel Problem Solving From Nature - PPSN IX. In *Proceedings of the 9th Internationational Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science 4193*, pages 192–201. Springer Berlin / Heidelberg, 2006.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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. Digital Proceeding*, 2009.

[10] P. Moscato and C. Cotta. *Handbook of Metaheuristics*, chapter A Gentle Introduction to Memetic Algorithms, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.

[11] 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.

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

[13] H. P. Schewefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.

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

[15] 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`.

[16] 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.

[17] 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.