A Memetic Algorithm with Simplex Crossover for Solving Constrained Optimization Problems

Miriam Pescador Rojas and Carlos A. Coello Coello
Departamento de Computación (Evolutionary Computation Group),
CINVESTAV-IPN, Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07360
(email: pescador@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx)

***Abstract*—In this paper, we propose a new memetic algorithm (MA) for solving constrained optimization problems over continuous search spaces. Our MA is composed by a global search mechanism based on differential evolution (DE), a constraint-handling technique called stochastic ranking (SR) and a local search (LS) procedure which adopts a simplex crossover (SPX) operator. We show that the performance of our algorithm is improved by the influence of its LS mechanism. In order to avoid premature convergence, we adopt a diversity mechanism and a replacement strategy. Our proposal is validated using a set of standard test problems taken from the specialized literature. The results are compared with respect to those produced by three representative algorithms of the state-of-the-art in the area.**

I. Introduction

Memetic algorithms (MAs) are optimization techniques based on a strategic combination between global and local search (LS) mechanisms [1]. While a global mechanism explores all of the search space, the local mechanism exploits certain regions within it, aiming to refine the solutions previously obtained. Both schemes work in a cooperative way for achieving a trade-off between the exploration and the exploitation of the search space [2].

Several optimization problems are modeled using decision variables over a continuous domain. Such problems have been solved (with different degrees of success) by different types of evolutionary algorithms (EAs), including particle swarm optimization (PSO) [3], differential evolution (DE) [4], real-coded genetic algorithms (RCGA) [5], [6] and evolution strategies (ES) [7].

The design of a LS mechanism for continuous search spaces is associated with the use of neighborhood structures. In this regard, there exist proposals based on direct search methods [8], [9], [10], gradient search methods [11], approaches based on heuristics [12], [13], and neighborhood-based genetic operators [14], [15], [16].

The design of LS mechanisms over continuous search spaces is not trivial, since it depends on the type of problem to be solved (each type of search space requires a specific type of movements generator that allows making small and accurate steps). In summary, the good design of an LS engine requires of: a mechanism to build a neighborhood structure, defining a policy regarding the frequency of use of the LS engine, a mechanism to select the solutions on which the LS engine will work, a replacement strategy for the solutions generated by the LS engine and a mechanism that allows us to maintain diversity with the aim of avoiding premature convergence.

The remainder of this paper is organized as follows. In Section II, we present the basic concepts required for understanding the rest of this paper. In Section III, we provide a short review of the most relevant previous related work. In Sections IV and V, we briefly describe the simplex crossover operator and the stochastic ranking mechanism for handling constraints, since they are both incorporated into our proposed approach. In Section VI, we describe our proposed algorithm, and its validation is presented in Section VII. Finally, our conclusions and some paths for future research are presented in Section VIII.

II. Basic Concepts

We are interested in the general nonlinear programming problem:

To find $\vec{x} = [x_1, x_2, ..., x_n]^T$ which minimizes $f(\vec{x})$ subject to:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, m \tag{1}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{2}$$

where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_n]^T$, m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear). If we denote with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$. For an inequality constraint that satisfies $g_i(\vec{x}) = 0$, then we will say that is active at $\vec{x}$.

III. Previous Related Work

The DE algorithm has been found to be a very robust and effective search engine for single-objective optimization problems. This has motivated the development of a number of DE variants, including hybrid approaches that combine DE with some other optimization technique. Here, we will only focus on memetic algorithms (MAs) that are based on DE. In [18] a MA called DEahcSPX was introduced. This algorithm combines DE with a hill-climbing technique and a SPX operator, for solving single-objective optimization problems. DEahcSPX implements an iterative process that invokes the LS engine, in order to refine the best solution from the current population until no further improvement is possible. After

that, the global search mechanism is adopted to generate new offspring. Such a global search engine relies on DE in its rand/1/bin version [4]. This approach could successfully handle problems with high dimensionality.

Liu et al. [19] proposed a memetic co-evolutionary differential evolution (MCODE) approach for constrained optimization. This MA uses a method called cooperative co-evolution [20], which is based on the cooperative relationship of species in nature. This algorithm uses two cooperative populations, which are created by DE in its best/1/bin version; these two populations evolve independently. The first population aims to minimize the objective function without considering the constraints, while the second is responsible for minimizing the penalty functions, regardless of the objective function values. These two populations interact using a migration operator. This proposal uses a Gaussian mutation operator as its LS engine. LS is applied if the best solution does not change after several iterations. Additionally, MCODE adopts an adaptive scheme for transforming infeasible solutions into feasible ones.

Ali and Kajee-Bagdadi [21] proposed an approach called local exploration-based differential evolution (LEDE). This is an algorithm that combines a DE variant with a periodic local exploration technique, which consists of a modification to the pattern search (PS) method [24] called LPS. The LS engine is invoked at the end of the kth iteration of the global search engine which explores the vicinity of the best point in the current population. Two strategies are adopted to handle constraints in LEDE, the superiority of feasible points [22] and the penalty-free parameter approach [23] . This algorithm modifies the mutation operator of DE as follows:

$$\hat{x}_{i,k} = x_{b,k} + F_k(x_{\beta,k} - x_{\alpha,k}) \tag{3}$$

where $x_{b,k}$ is the best point in the population, while $x_{\beta,k}$ and $x_{\alpha,k}$ are points uniformly selected from the population. F_k is a scaling parameter which depends on the current iteration of the algorithm, and takes values uniformly in the interval $[0.4, 1]$.

IV. Simplex Crossover Operator

The simplex crossover operator (SPX) [15] is an interesting technique, which is adopted here as a LS engine. SPX creates y_k offspring uniformly distributed around the centroid of their parents, from an area defined by a k-simplex. SPX select k different points , for constructing a simplex. This simplex is expanded in each direction $(x_i^G - O)$ with $(\epsilon \geq 0)$, where O is the center of mass of the k parents, calculated as:

$$O = \frac{1}{k}\sum_{i=1}^{k} x^{(i)} \tag{4}$$

and

$$y_i = O + \epsilon(x_i^G - O), (i = 1, ..., k) \tag{5}$$

A variant of SPX given in [15] is called SPX-n-m-ε, where n is the number of directions in the search space, m is the number of parents and ε is a control parameter that defines the rate of expansion.

V. Stochastic Ranking

Stochastic Ranking (SR) [25] is a constraint-handling method which has been very popular in the specialized literature because of its simplicity and effectiveness. SR maintains a balance between the influence of the objective function and the degree of violation of the constraints when assigning the fitness value to an individual. SR does not require the definition of penalty factors. This mechanism sorts λ potential solutions, using a procedure similar to the bubble sort method. Instead of attempting to determine the optimum values for the penalty factors, SR defines a rate Pf to determine the balance between the objective function value and the constraint violation value. This means that comparisons are made in pairs. The search engine of SR is a multi-member evolution strategy (ES) which carries out mutations generated by a normal distribution of probabilities. It also uses a coefficient to adjust the step size for each decision variable in each individual.

VI. Our Proposed Approach

Our proposed approach is called SPX-MACO, and it includes a modified DE algorithm, SPX and a replacement strategy which aims to maintain diversity in the population. Each of these mechanisms is described next in more detail.

A. Global Search Mechanism

DE is one of the best currently available global optimizers. Besides being very effective, it usually converges very quickly to the optimum of a problem and, generally, with great consistency. Additionally, it requires few control parameters for its operation and its implementation is very simple.

The rand/1/bin model of DE has been found to have a good performance with a small number of fitness function evaluations. Therefore, SPX-MACO uses this version of DE as its global search mechanism. The selection process of DE is, however, modified in our approach, considering a binary tournament selection with the following rules:

1) If the two solutions compared are infeasible, we prefer the solution that violates less the constraints.
2) If the two solutions are infeasible and have the same constraint violation, we prefer the solution with better objective function value.
3) If the two solutions are feasible, we prefer the one with the best (i.e., lowest) objective function value.

DE is shown in Algorithm 1, and consists of the following stages:

Initialization. A population $P_{x,0}$ of N_p $D-$dimensional parameter vector $x_{1,i,0}, ..., x_{D,i,0}$, $i = 1, ..., N_p$ is randomly generated within the lower and upper bound of variables $b_L = [b_{1,L}, ..., b_{D,L}]$ and $b_U = b_{1,U}, ..., b_{D,U}$.

Trial vector generation. At the gth generation, a trial population $P_{u,g}$ consisting of N_p $D-$dimensional trial vectors

$u_{i,g} - [u_{1,i,g,...,D,i,g}]$ is generated via mutation and recombination operations applied to the current population $P_{x,g}$.

Differential Mutation. With respect to each vector $x_{i,g}$ in the current population, a mutant vector $u_{i,g}$ is generated by adding and scaling from three randomly vectors selected from the current population.

Algorithm 1: DE algorithm, which is the **global search mechanism** adopted by our proposed approach

Data: Cr (crossover rate), F (mutation rate), Sp (size of population), n (number of variables)
Result: New population

```
begin
    for i ← {1, ..., Sp} do
        Select randomly r1, r2, r3 in current population,
        where r1 ≠ r2 ≠ r3 ≠ i ;
        for j ← {1, ..., n} do
            if rand_j[0, 1] < Cr then
                u_{i,j} ← x_{r3,j} + F(x_{r1,j} − x_{r2,j}) ;
            else
                u_{i,j} ← x_{i,j};
            Apply selection rules and replace;
    return a new population;
```

B. Constraint Handling Mechanism

We adopted SR as our constraint-handling mechanism, because it is simple and quite effective. SR is shown in Algorithm 2.

Algorithm 2: Stochastic Ranking. Constraint-handling mechanism.

Data: Pf (rate of comparison between objective function value and penalty function value), Sp (size of population),
Result: Sorted population

```
begin
    for i ← {1, ..., Sp} do
        for j ← {1, ..., λ − 1} do
            if ψ(x_j) == ψ(x_{j+1}) == 0 || U(0, 1) < Pf
            then
                if f(x_j) > f(x_{j+1}) then
                    swap (x_j, x_{j+1}) ;
            else
                if ψ(x_j) > ψ(x_{j+1}) then
                    swap (x_j, x_{j+1}) ;
        if no swap done then
            break;
    return sorted population;
```

C. Assortative Mating and Replacement Strategy for the LS Engine

An important issue is how to define the selection of parents for applying LS. In our case, we obtained good results using positive assortative mating [6]. In this selection procedure, the individuals which are more similar among themselves are selected for building a neighborhood. We took advantage of our sorted population (this is done by SR), for selecting the first and last elements from the population, and then we apply LS to them. This means that our LS engine explores the neighborhood of both the best and the worst individuals in the population. Regardless of the outcome from applying LS, we add to the population the two individuals obtained from this exploration. This does not increase the total population size, since upon applying the selection mechanism of DE, we return to the maximum population size defined by the user.

D. Local Search Mechanism

In [18], the authors show that the SPX crossover operator has features that allow it to improve the performance of DE when solving unconstrained (single-objective) optimization problems. Based on this, we decided to extend this operator for dealing with constrained optimization problems. The SPX crossover operator performs the following steps:

Step 1. Select n_p parent vectors of the population $\vec{x}_i^G$, where $i = 1, \ldots, n_p$. The selected vectors define the search neighborhood by a k-simplex. In SPX-MACO, we establish n_p equal to 3 parents, for an n-dimensional space.

Step 2. Compute the centroid defined by $\vec{O}$ of the k-simplex. If n_p is equal to 3, the centroid for the 2-simplex is defined by:

$$\vec{O} = \frac{1}{3}\sum_{i=1}^{3} \vec{x}_i^{\,G} \tag{6}$$

where $\vec{x}_i$ is a solution vector in the population at generation G.

Step 3. Apply expansion parameter ϵ. Our SPX-MACO adopts an expansion factor $\epsilon = 1.3$

Step 4. Generate r_i random numbers using:

$$r_i = u^{\frac{1}{i+1}}, (i = 1, \ldots, n_{p-1}), \tag{7}$$

where $u \in [0, 1]$ is a random number in the range from 0 to 1. For the 2-simplex, these factors are $r_1 = u^{\frac{1}{2}}$ and $r_2 = u^{\frac{1}{3}}$.

Step 5. Compute the values for y_i and C_i according to equations (8) and (9).

$$y_i = O + \epsilon(x_i^G - O), (i = 1, ..., 3) \tag{8}$$

$$\vec{C}_i = \begin{cases} 0, & (i = 1) \\ \vec{r}_{i-1}(\vec{y}_{i-1} - \vec{y}_i + \vec{C}_{i-1}), & (i = 2, 3) \end{cases} \tag{9}$$

Step 6. Generate a descendant $\vec{C}$ from equation (10) and return a new solution.

$$\vec{C} = \vec{y}_3 + \vec{C}_3 \tag{10}$$

Algorithm 3 shows our complete LS procedure.

Algorithm 3: SPX crossover operator, **local search mechanism**

Data: n_p (number of solutions adopted for generating the simplex), ε (expansion rate for SPX), $parent_{spx}$ (parent solution for generating the simplex)
Result: New solution

begin
- Compute centroid $O = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Generate random numbers $r_i = u^{\frac{1}{i+1}}, (i = 1, \ldots, n-1)$
- Compute $y_i = O + \varepsilon(x_i - O), (i = 1, \ldots, n)$
- $C_i = \begin{cases} 0, & (i = 1) \\ r_{i-1}(y_{i-1} - y_i + C_{i-1}), & (i = 2, \ldots, n) \end{cases}$
- Generate an offspring $C = y_n + C_n$
- return offspring;

E. SPX-MACO

Our proposed SPX-MACO randomly creates an initial population using a uniform distribution. DE is applied as our global search mechanism for exploring all the search space. Our approach adopts the selection mechanism previously described, in order to incorporate the constraints of the problem into the search process. The new solutions generated by DE are sorted by the SR method. After that, the simplex crossover operator is applied to the best and to the worst solutions from the ranking in order to produce new individuals, which are added to the population. This process is repeated several times, until fulfilling the stopping criterion.

Algorithm 4 shows the full pseudo-code of our proposed SPX-MACO.

VII. Experimental Results

A. Test Problems

The test suite used for validating our proposed approach consists of 10 test problems taken from the set designed for a special session organized at the *2006 IEEE Congress on Evolutionary Computation* [26]. This set is composed by different types of objective functions: quadratic, nonlinear, polynomial and cubic. The problems also have different types of constraints (linear, nonlinear, equality and inequality). Among the features that make these problems difficult, we have: high dimensionality, multi-modality, and disjoint feasible regions.

All the test problems were transformed to the following format:

$$\text{Minimize } f(\vec{x}), \vec{x} = [x_1, x_2, ..., x_n] \tag{11}$$

subject to:

$$g_i(\vec{x}) \leq 0, i = 1, ..., q \tag{12}$$

Equality constraints were transformed into inequality constraints of the form:

$$|h_j(\vec{x})| - \epsilon \leq 0, \text{ with } j = q+1, ..., m \tag{13}$$

Algorithm 4: SPX-MACO

Data: Pop (initial population), G (number of generations), n_p (number of parents selected for SPX crossover operator)
Result: Best solution

begin
- Generate an initial population with uniform distribution;
- **while** *termination criteria is met* **do**
  - Apply **global search mechanism**;
  - Sort population using **stochastic ranking**;
  - Select the first n_p solutions from the sorted list ($best_{sol}[n_p]$);
  - Select the last n_p solutions from the sorted list ($worst_{sol}[n_p]$);
  - **for** $best_{sol}[n_p]$ *and* $worst_{sol}[n_p]$ **do**
    - Apply **local search mechanism** (SPX);
    - The new solution is added to the current population;
  - $G \leftarrow G + 1$;
- return the best solution;

where: ϵ is the allowable tolerance for the equality constraints. In the experiments reported here, we used $\epsilon = 0.0001$.

B. Experimental Methodology

The parameters adopted by our SPX-MACO were the following (these values were obtained after numerous experiments):

- For SPX: $\epsilon = 1.3$, $n_p = 3$
- Penalty function $= \sum_{j=1}^{m} max\{0, g_j(x)\}^2$
- For SR (our constraint-handling mechanism): $Pf = 0.45$
- For SPX-MACO: $G = 2500$
- DE-related parameters $Sp = 70$, for problems g01, g05, g07, g10, g14, g17, g19: $F = 0.9$ and $Cr = 0.9$ and problems g02, g03,g13: $F = 0.8$ and $Cr = 0.6$

For validating our proposed approach, we performed 100 independent runs for of the test problems considered, and we compared results with respect to the following approaches: stochastic ranking (SR), memetic co-evolutionary differential evolution (MCODE) and local exploration-based differential evolution (LEDE). In order to allow a fair comparison, all the algorithms performed 180,000 objective function evaluations. Our results are summarized in Table I.

The parameter values used for other algorithms are established by the authors in the original proposals.

- MCODE adopted a penalty function of the form: $\sum_{j=1}^{m} |g_j(x)|$, and LEDE adopted a penalty function of the form: $\sum_{j=1}^{m} max\{0, g_j(x)\}$.
- The mutation and crossover rates of MCODE were: $F = 0.8$, and $Cr = 0.8$, respectively. For LEDE, the mutation rate was adaptive, and the crossover rate was set as: $Cr = 0.9$.

C. Effect of Local Search

Knowing that DE is a very powerful search engine by itself, it was important for us to determine if the use of local search was actually beneficial in our proposed approach. Thus, we performed an additional experiment in which we compared the results obtained by our approach without LS (this version was called MACO) with respect to those obtained by our memetic approach (SPX-MACO). The results of this experiment are summarized in Table II. These results clearly indicate that there was an improvement when using LS in practically all cases.

VIII. Conclusions and Future Work

We have proposed here a new memetic algorithm called SPX-MACO, for solving constrained optimization problems. The proposed approach is based on the use of DE (in its rand/1/bin version) as a global search engine, and it also incorporates a variation of stochastic ranking and a simplex crossover operator. Local search, in our case, is applied on the neighborhood of the best and the worst solutions in the population.

The comparison of results of our SPX-MACO with respect to those generated by three other state-of-the-art approaches indicated that our proposal is competitive, while requiring a low number of objective function evaluations.

As part of our future work, we are interested in applying our approach to some real-world problems and in extending it to the solution of multi-objective optimization problems.

Acknowledgements

The second author acknowledges support from CONACyT project no. 103570.

References

[1] P. Moscato, *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms,* Technical Report C3P Report 826, 1989.

[2] N. Krasnogor and J. Smith, *A Memetic Algorithm With Self-Adaptive Local Search: TSP as a case study,* Genetic Evolutionary Computation Conference pp. 987-994, 2000.

[3] R. C. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory,* Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39-43, 1995.

[4] R. Storn and K. Price, *Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces,* Technical Report TR-95-012, Berkeley, CA, 1995.

[5] F. Herrera, M. Lozano and J. L. Verdegay, *Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis,* Artificial Intelligence Review, vol. 12, no. 4, pp. 265-319, 1998.

[6] M. Lozano, F. Herrera, N. Krasnogor and D. Molina, *Real-Coded Memetic Algorithms with Crossover Hill-Climbing,* Evolutionary Computation vol. 12, no. 3, pp. 273-302, 2004.

[7] D. B. Fogel, *Selected Readings on the History of Evolutionary Algorithms,* Evolutionary Computation. The Fossil Record, the Institute of Electrical and Electronic Engineers, New York, 1998.

[8] Robert Hooke and T. A. Jeeves, *Direct Search Solution of Numerical and Statistical Problems,* Journal of the ACM, vol. 8, no. 2, pp. 212-229, 1961.

[9] G. R. Hext, W. Spendley and F. R. Himsworth, *Sequencial application of simplex designs in optimization and evolutionary operation,* Technometrics, vol. 4, no. 4, pp. 441-461, 1962.

[10] J. A. Nelder and R. Mead, *A simplex method for function minimization,* The Computer Journal, vol. 7 pp. 308-313, 1965.

[11] D. P. Bertsekas, *Constrained Optimization and Lagrange multiplier methods,* Massachusetts Institute of Technology, Athena Scientific, Belmont , Massachusetts, Inc. 1982.

[12] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, *Optimization by Simulated Annealing,* Science. New Series 220 (4598), pp. 671-680, 1983.

[13] F. Glover, *Tabu Search,* ORSA Journal on Computing, vol. 1, no. 3, pp. 190-206, 1989.

[14] L. J. Eshelman, K. E. Mathias and J. D. Schaffer, *Crossover Operator Biases: Exploiting the Population Distribution,* Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 354-361, 1997.

[15] S. Tsutsui, M. Yamamura and T. Higuchi, *Multi-parent recombination with simplex crossover in real coded genetic algorithms,* Genetic Evolutionary Computation Conference (GECCO99), pp. 657-664, 1999.

[16] N. K. Bambha, S. S. Bhattacharyya, J. Teich and E. Zitzler, *Systematic integration of parameterized local search,* evolutionary algorithms, IEEE Transactions on Evolutionary Computation, vol. 8, no. 2, pp. 137-155, 2004.

[17] S. Das and P. N. Suganthan, *Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real-world optimization problems,* in Technical Report, 2010.

[18] N. Noman and H. Iba, *Accelerating Differential Evolution Using an Adaptive Local Search,* IEEE Trans. Evolutionary Computation, vol. 12, no. 1, pp. 107-125, 2008.

[19] B. Liu, H. Ma, X. Zhang and Y. Zhou, *A memetic co-evolutionary differential evolution algorithm for constrained optimization,* in Evolutionary Computation, CEC 2007. IEEE Congress on Issue, pp. 2996-3002, 2007.

[20] Y. Shi, H. Teng and Z. Li, *Cooperative Co-evolutionary Differential Evolution for Function Optimization,* ICNC (2), Lecture Notes in Computer Science, vol. 3611, pp. 1080-1088, 2005.

[21] M. M. Ali and Z. Kajee-Bagdadi, *A local exploration-based differential evolution algorithm for constrained global optimization,* Applied Mathematics and Computation, pp. 31-48, 2009.

[22] D. Powell and M. M. Skolnick, *Using genetic algorithms in engineering design optimization with non-linear constraints,* in Stephanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 424-431, 1993.

[23] K. Deb, *An Efficient Constraint Handling Method for Genetic Algorithms,* Computer Methods in Applied Mechanics and Engineering, vol. 2, no. 4, pp. 311-338, 2000.

[24] T.G. Kolda and R.M. Lewis and V. Torczon *Optimization by direct search: new perspective on classical and modern methods* SIAM Review vol. 45, no. 3 , pp. 385-482, 2003

[25] T. P. Runarsson and X. Yao *Stochastic ranking for constrained evolutionary optimization* IEEE Trans. Evolutionary Computation, vol. 4, no. 3, pp. 284-294, 2000

[26] E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, K. Liang and T. P. Runarsson *Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization.* in Technical report, 2006.

Problem	**g01**	**g02**	**g03**	**g05**	**g07**	**g10**	**g13**	**g14**	**g17**	**g19**
Optimum	-15.0	-0.803619	-1.0005	5126.4967	24.306209	7049.248020	0.0539415	-47.764888	8853.5338	32.655592
Stochastic Ranking (SR)										
best	**-15.0**	-0.766997	-0.999707	**5126.4967**	24.307578	7049.999279	0.053943	-47.747134	8875.9217	33.041103
median	**-15.0**	-0.712116	-0.997721	**5126.4967**	24.307578	7050.863179	0.438802	-47.747134	9034.0876	33.077049
mean	**-15.0**	-0.711341	-0.997962	**5126.6555**	24.313864	7059.034467	0.331740	**-47.742502**	9031.3350	33.173030
st. dev	**0.0**	**6.12E-03**	**6.27E-04**	**1.15**	0.017E-02	47.45	1.62E-01	**3.32E-02**	**22.38**	4.07E-01
worst	**-15.0**	**-0.706099**	**-0.997721**	**5137.9397**	24.462913	7525.392220	0.586825	**-47.463530**	9054.6249	37.153993
Memetic Co-evolutionary Differential Evolution (MCODE)										
best	**-15.0**	-0.803616	-1.000498	**5126.4967**	**24.306209**	7049.248030	**0.053941**	-47.764879	**8853.5338**	32.655726
median	**-15.0**	-0.743683	-0.380902	**5126.4967**	24.306311	7049.253333	0.438802	-47.650215	**8874.3750**	**32.665956**
mean	-14.9413	-0.733690	-0.374674	5127.3070	24.311192	7050.045833	0.271123	-47.486428	**8900.0818**	**32.720659**
st. dev	2.55E-01	5.51E-02	2.15E-01	8.04	4.41E-02	5.1321	1.99E-01	4.05E-01	48.62	**1.28E-01**
worst	-13.8281	-0.472336	-0.023192	5207.3343	24.750375	7091.226026	0.836580	-45.962903	9161.5502	33.409777
Local Exploration-based Differential Evolution (LEDE)										
best	**-15.0**	-0.803618	**-1.000500**	**5126.4967**	**24.306209**	**7049.248020**	0.135701	**-47.764888**	**8853.5338**	32.657763
median	**-15.0**	-0.606640	**-1.000500**	**5126.4967**	**24.306209**	**7049.248022**	0.438802	-47.742600	8929.3877	32.844874
mean	-14.3885	-0.606603	-0.996920	5210.4303	**24.306209**	**7049.248131**	0.365329	-47.629776	8956.8311	32.929617
st. dev	9.98E-01	1.05E-01	2.86E-02	146.89	**2.21E-06**	**4.36E-04**	1.9E-01	2.75E-01	102.36	3.26E-01
worst	-10.6562	0.331449	-0.722965	5706.7416	**24.306231**	7049.251059	0.999999	-45.890546	9232.4541	35.146084
SPX-MACO										
best	**-15.0**	**-0.803619**	**-1.000500**	**5126.4967**	**24.306209**	**7049.248020**	**0.053941**	**-47.764888**	**8853.5338**	**32.655706**
median	**-15.0**	**-0.770397**	-1.000492	**5126.4967**	24.306221	7049.248063	**0.053944**	**-47.763423**	8913.7372	32.743630
mean	**-15.0**	**-0.761270**	**-0.999010**	5126.9478	24.306293	7049.248279	**0.053966**	-47.632592	8912.5736	32.809258
st. dev	**0.0**	3.3E-02	7.04E-03	4.65	1.79E-04	6.15E-04	**9.81E-05**	3.55E-01	36.99	1.55E-01
worst	**-15.0**	-0.671374	-0.961216	5141.9895	24.306933	**7049.250759**	**0.054833**	-46.005550	8951.2457	**33.206513**

TABLE I

COMPARISON OF THE RESULTS OBTAINED BY OUR SPX-MACO AND THREE STATE-OF-THE-ART ALGORITHMS: SR, MCODE AND LEDE. THE BEST RESULTS FOUND IN EACH CASE ARE SHOWN IN **boldface**.

Problem	**Algorithm**	**best**	**median**	**mean**	**st. dev.**	**worst**
g01	MACO	-14.999946	-14.999805	-14.999823	9.91E-05	-14.999427
-15.0	SPX-MACO	**-15.0**	**-15.0**	**-15.0**	**0.0**	**-15.0**
g02	MACO	-0.716864	-0.630202	-0.623580	3.37E-02	-0.545416
-0.803619	SPX-MACO	**-0.803618**	**-0.770397**	**-0.761270**	**3.30E-02**	**-0.671374**
g03	MACO	-0.835070	-0.273257	-0.256595	0.122793	-0.083632
-1.0005	SPX-MACO	**-1.0005001**	**-1.000492**	**-0.999010**	**7.04E-03**	**-0.961216**
g05	MACO	5126.496714	5153.272313	5131.352670	47.776368	5312.815637
5126.4967	SPX-MACO	5126.496714	**5126.496714**	**5126.947831**	**4.658087**	**5141.989532**
g07	MACO	24.336617	24.364586	24.362005	1.44E-02	24.413607
24.306209	SPX-MACO	**24.306209**	**24.306221**	**24.306293**	**1.79E-04**	**24.306922**
g10	MACO	7050.313660	7052.712113	7052.632264	1.187864	7056.516780
7049.248020	SPX-MACO	**7049.248021**	**7049.248063**	**7049.248279**	**6.15E-04**	**7049.250759**
g13	MACO	0.346487	0.851239	0.912186	**0.153820**	0.999510
0.0539415	SPX-MACO	**0.053941**	**0.438802**	**0.331740**	0.162748	**0.5868255**
g14	MACO	-47.757573	-47.632699	-47.666634	**0.112765**	**-47.074625**
-47.764888	SPX-MACO	**-47.764888**	**-47.764845**	**-47.736939**	0.128721	-46.719956
g17	MACO	8854.615355	8949.925175	8950.020171	58.898262	9176.291456
8853.5396	SPX-MACO	**8853.533874**	**8913.737242**	**8899.960982**	**36.999861**	**8951.24573**
g19	MACO	35.864368	38.325524	38.312506	1.045887	41.130285
32.655592	SPX-MACO	**32.655706**	**32.743630**	**32.809258**	**0.155246**	**33.206513**

TABLE II

STATISTICAL RESULTS SHOWING THE INFLUENCE OF THE LOCAL SEARCH IN THE PERFORMANCE OF OUR APPROACH. HERE, $MACO$ REFERS TO THE ALGORITHM WITHOUT LOCAL SEARCH AND $SPX-MACO$ REFERS TO THE VERSION THAT USES LOCAL SEARCH. THE EXACT SAME INITIAL POPULATION WAS USED FOR BOTH APPROACHES IN ORDER TO AVOID ANY STATISTICAL BIAS IN THE RESULTS.