ISCIES'11

Proceedings of ISCIES'11
2nd International Symposium on
Computational Intelligence for Engineering Systems

Edição Científica: Ana Madureira (ISEP), Cecília Reis (ISEP), Viriato Marques (ISEC)
Editor: ISEC (Instituto Superior de Engenharia de Coimbra)

ISBN: 978-989-8331-12-0

Instituto Superior de Engenharia de Coimbra
Coimbra, Portugal
16-18 November, 2011
http://www.iscies11.ipc.pt

Instituto Superior de Engenharia do Porto

gecad
Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

IEEE
PORTUGAL SECTION

IEEE Computational Intelligence Society
Portugal Chapter

Spreading Algorithm for Single-Objective Problems

E. J. Solteiro Pires*, Luís Mendes†, António M. Lopes‡,
P. B. de Moura Oliveira§, J. A. Tenreiro Machado¶,
*Centro de Investigação e de Tecnologias Agro-Ambientais e Biológicas
Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal
Email: epires@utad.pt
†Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Portugal
Instituto de Telecomunicações, Portugal
Email: lmendes@estg.ipleiria.pt
‡Instituto de Engenharia Mecânica
Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
Email: aml@fe.up.pt
§Centro de Investigação em Desporto, Saúde e Desenvolvimento Humano
Escola de Ciências e Tecnologia
Universidade de Trás-os-Montes e Alto Douro, Vila Real, Portugal
Email: oliveira@utad.pt
¶Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão,
Instituto Superior de Engenharia do Porto, Instituto Politécnico do Porto, Portugal
Email: jtm@isep.ipp.pt

***Abstract*—This paper addresses the problem of finding several different solutions with the same optimum performance in single objective real-world engineering problems. In this paper a parallel robot design was proposed. Thereby, this paper presents a genetic algorithm to optimize uni-objective problems with an infinite number of optimal solutions. The algorithm uses the maximin concept and ϵ-dominance to promote diversity over the admissible space. The performance of the proposed algorithm is analyzed with three well-known test functions and one function obtained from practical real-world engineering optimization problems. A spreading analysis is performed showing that the solutions drawn by the algorithm are well dispersed.**

***Keywords*-spreading technique, genetic algorithm**

I. Introduction

Some single-objective problems have several optimal solutions. A finite number of solutions can be found at different peaks of the objective function or an unlimited set of solutions exist along one or more regions in the parameter space. Therefore, achieving a well-spread and non-dominated parameter front is of paramount importance to the decision maker, since he can choose from a set of optimum solutions the one best suited to be developed or implemented in the problem context.

Evolutionary algorithms use different approaches to promote the solutions diversity, such as the sharing model, crowding and maximin techniques. The sharing model, originally suggested by Goldberg and Richardson [8], is used to obtain a set of optimal solutions distributed by several optimal peaks. To promote solutions in less crowded regions of the parameter space and, therefore, to *force* the population to be well distributed, the sharing model degrades the fitness values of similar solutions according to the distance of their closest neighbors. In this method, the number of optimum solutions under each peak is proportional to its value.

The crowding technique, introduced by De Jong [3], replaces offspring by *similar* solutions existing in the population in order to maintain its diversity. *Similarity* is measured, like in the sharing model, by evaluating the distance between solutions. In some crowding technique variants, the offspring compete against their parents, giving the winners a chance to breed in the next population. This method tends to spread solutions over the most prominent search space peaks, achieving a number of solutions in each peak proportional to its base size.

Deb and Tiwari [6], [7] proposed an algorithm to solve several types of optimization: single-objective uni-optimal problems, single-objective multi-optima problems, multi--objective uni-optimal and multi-objective multi-optima problems. The Deb and Tiwari algorithm is capable of solving the problem without knowing at the outset the type of optimization to perform. In single-objective multi-optima problems they [6], [7] use the ϵ-dominance and the crowding distance in the objective and the parameter space, respectively. These authors apply the algorithm in multi-modal problems which have several optimum solutions in different peaks.

Maximin is used in classic multi-attribute problems [17]

and in game theory [12], [14]. The maximin strategy aims to maximize the result from a set of values that were previously obtained, from other sets, using minimizing functions. In 2003, Balling [1] proposed a multi-objective optimization technique based on a fitness function derived from the maximin strategy [12] and Li [11] used the maximin fitness in a particle swarm multi-objective optimizer. Moreover, Pires *et al.* [16] used the maximin technique to find a well distributed non-dominated Pareto front in multi-objective optimization algorithms. The maximin strategy can be adopted by genetic algorithms (GAs) to obtain a set of solutions well distributed along the parameter space. The maximin technique is mainly used to obtain a set of points from a continuous front instead of finding the peaks of multi-modal functions.

The method proposed in this paper uses the maximin concept together with the ϵ-dominance [10] in order to find a front in the parameter space. This method presents a better performance than the one reported in the literature [13], which adopts the sharing scheme, because it reduces significantly the solutions dispersion along the optimum front and, consequently, leading a better defined front.

The paper is organized as follow. Section II describes the maximin spreading algorithm. This optimization method promotes solutions diversity in the parameters space. Section III presents the simulation results of five optimization problems. The first three problems follow two well-known and widely used functions: the Himmelblau's, squared sine and double pulse functions. These equations are used by the algorithm as the fitness function in three different optimizations. The fourth and fifth functions presented in section III are obtained from real-world engineering applications. One optimizes the design of radio frequency switched capacitor arrays and the other optimizes of the kinematics of parallel robots. Section 4 studies the spreading of the solutions drawn by the proposed algorithm. Finally, section V outlines the main conclusions.

II. Maximin Spreading Algorithm

In multi-objective problems there are several ways to find the Pareto front. One of them is based on the ϵ-dominance concept in which solutions with slightly inferior fitness values are allowed to remain in the population as non--dominated ones. This technique is used to get a set of solutions with good spread and diversity over the multi-objective space [10].

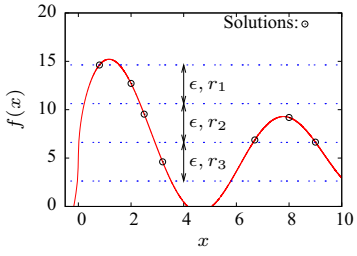
The proposed single-objective algorithm adopts the above--mention concept and the maximin scheme as the main techniques to achieve good diversity in the parameters space. Initially, the algorithm divides the objective space using virtual horizontal parallel hyperplanes (straight lines or planes for a 1-D or 2-D dimensional parameter space, respectively), that are separated from each other by a ϵ-distance (see Figure 1(a)). Two consecutive hyperplanes define a ϵ-rank, where all the solutions have the same preference, even if their objective values are different, as illustrated in Figure 1(a). After the division of the objective space, the algorithm selects the solutions, starting from the best ϵ-ranks (r_1), until a ϵ-rank is found with more solutions than the empty slots available in the new population. However, if the best ϵ-rank (rank r_1) has more solutions than the number of individuals of the new population, then the solutions of the other ϵ-ranks are not considered. In both cases, the best distributed solutions in the parameter space are selected according to the maximin scheme. The basic idea behind the maximin sorting scheme is the selection of the solutions that decrease the large gap areas where no solution exists in the already selected population. For example, consider the population solutions of one ϵ-rank depicted in Figure 1(b). In this case, two parameters $\{x_1, x_2\}$ are considered. Initially, two extreme solutions for each parameter are selected, $\{a, b\}$ and $\{d, c\}$ for x_1 and x_2, respectively. Through this selection the set $S \equiv \{a, b, c, d\}$ is initialized. Then, the solution e is selected because it has the larger distance to the set S. After that, solution f is selected for inclusion into the set $S \equiv \{a, b, c, d, e\}$, for the same reasons. The process is repeated until population S is completed. The maximin technique boosts its performance as the number of iterations increases, since all the solutions, which are already in the first ϵ-rank, tend to spread along a non-dominated front.

The maximin sorting scheme is depicted in Figure 2. In each generation new offspring (set D) are merged with their progenitors (set P), according with Figure 2, leading to the new set R (line 1). After that, the algorithm may select, for each one of the optimization parameters (n_{par}), the extreme solutions (getMin and getMax functions) from rank r_1 (lines 5-7) and introduces them into the final population (set S). Then, the individuals of lower rank are removed (getRankMin function) from the auxiliary population A and inserted into the set S until the number of solutions of the current rank surpass the allowed number of solutions of set S (lines 9-12). Next, the squared distance, c_{a_j} (1a), between each solution, a_j, and the solutions already selected, s_i, is evaluated (lines 13-15). After that, the solution a_j, whose squared distance to the set S is the larger (k solution), is selected (1b) (getMaxCi function, line 17). Each time a solution enters into the set S (line 18), the cost c_{a_l} of the set A is reevaluated (lines 19-21). This process ends when the set S is completed, requiring, at most, $O(pop_{dim}^2)$ computations, where pop_{dim} represents the number of population elements.

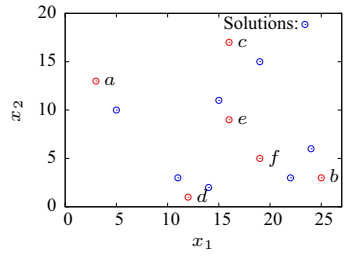
$$c_{a_j} = \min_{s_i \in S, a_j \in A} \| a_j - s_i \|^2 \tag{1a}$$

$$S = S \cup \{a_j : c_{a_j} = \max_{a_i \in A} c_{a_i}\} \tag{1b}$$

The proposed algorithm can be used with a static constant or a dynamic variable ϵ-parameter across all the iterations. In the dynamic technique, as the number of algorithm iterations



(a)



(b)

Figure 1. (a) Problem with one objective and one parameter $\{x\}$, where r_m represents the rank $m = \{1, ..., 3\}$ and ϵ is the rank length. (b) Solutions in a bi-dimensional parameter space $\{x_1, x_2\}$ represented by the dot-points. Solutions $\{a, b, ..., f\}$ are the most distant from each other.

increases, the ϵ-length is decreased so that the algorithm favors the search of new solutions in regions near the optimal front. Therefore, the dynamic ϵ-technique leads to better results than the static ϵ-method. In fact, a more well defined (*i.e.*, low dispersed) front is obtained at the end of the algorithm, while maintaining a good exploration at the beginning.

III. Simulation Results

This section presents the simulation results of the proposed algorithm for several problems. The first three functions, which have well-known optima values, are used to validate the presented algorithm. The Himmelblau's and the squared sine functions are selected directly from the literature and the third one, the double pulse function, is chosen because it has two continuous parameters fronts. The last two functions are obtained from two real-world engineering problems, namely, the design optimization of radio frequency circuits and the kinematic optimization of parallel robots. For each considered problems there is only one optimization function. However, each problem has several different optimums solutions differing between them by the parameters values. In all the optimization problems parameters are encoded directly into real-value vectors.

1: $R = P \cup D$
2: $S = \emptyset$
3: $A = \text{getRankMin}(R)$
4: **if** $\#A > pop_{\text{dim}}$ **then**
5: **for** $i = 1$ **to** n_{par} **do**
6: $S = S \cup \text{getMin}(A, i) \cup \text{getMax}(A, i)$
7: **end for**
8: **end if**
9: **while** $\#S + \#A \leq pop_{\text{dim}}$ **do**
10: $S = S \cup A$
11: $A = \text{getRankMin}(R)$
12: **end while**
13: **for** $j = 1$ **to** $\#A$ **do**
14: $c_{a_j} = \min_{s_i \in S} \{\|a_j - s_i\|^2\}$
15: **end for**
16: **while** $\#S < pop_{\text{dim}}$ **do**
17: $k = \text{getMaxCi}(A)$
18: $S = S \cup k$
19: **for** $l = 1$ **to** $\#A$ **do**
20: $c_{a_l} = \min\{\|a_l - k\|^2, c_{a_l}\}$
21: **end for**
22: **end while**

Figure 2. Maximin spreading algorithm pseudo-code.

For all the above mention optimization problems the successive generations of new solutions are reproduced based on a linear ranking scheme and simulated binary crossover [9], [5] (with $\eta_c = 20$). When mutation occurs the operator replaces the value of one design parameter according to a uniform distribution function. The probabilities of crossover and mutation are $p_c = 0.6$ and $p_m = 0.05$, respectively.

At the end of each GA cycle, it is used a $m\epsilon - (\lambda + \mu)$ strategy to select the solutions which survive for the next iteration. This means that the best solutions among parents (λ) and offspring (μ) are chosen based on ϵ rank sort. To fill the last gap of the population the maximin (m) technique is used.

For the first three optimization functions, it is used a dynamic ϵ that is decreased during the optimization. The ϵ begins with the value of 5 and stops when reaches the final value 0.01. The ϵ is decreased by 1% every 10 iterations or when all the solutions belongs to the same rank.

A. Himmelblau's Function

The first optimization function used is the well-known Himmelblau's function [4]

$$f_1(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \quad (2)$$

where $-5 \leq x_i \leq 5$, $i = \{1, 2\}$. It has two parameters and four isolated minima, all having the optimal function value of $f_1^* = 0$. In this experiment a set of 100 solutions is

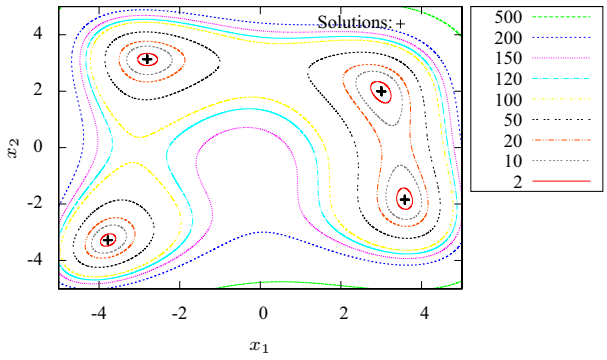


Figure 3. Solutions found for the Himmelblau's function.

Table I
SOLUTION DISTRIBUTION OVER THE OPTIMUMS OF $f_1(x_1, x_2)$

Optimal position $\{x_1, x_2\}$	Solution percentage
$\{+3.000, +2.000\}$	23%
$\{-2.805, +3.131\}$	18%
$\{-3.779, -3.283\}$	32%
$\{+3.584, -1.848\}$	27

used and the algorithm is executed during 1000 iterations. Figure 3 shows the obtained solutions for the Himmelblau's function.

It is clear that the algorithm converges and is able to find the four minima in only one simulation run. Moreover, the solutions are well distributed by all the 4 optimal as depicted by figure I.

B. Squared Sine Function

The second optimization function is a squared sine function:

$$f_2(x) = \sin^2(\pi x) \tag{3}$$

where $0 \leq x \leq 20$. It has one parameter and 21 isolated minima, all with the optimal value of $f_2^* = 0$. In this experiment a set of 100 solutions is used and the algorithm is executed during 1000 iterations. Figure 4 depicts the final solution set obtained for the sine function. It could be seen that the algorithm find all the isolated optimal.

At the end of the simulation, the solutions are well dispersed by all the optimal minimal. This phenomena can be observed examining the number of solutions in the neighborhood of each minima. The solutions percentage for each minimum peak can be seen in table II.

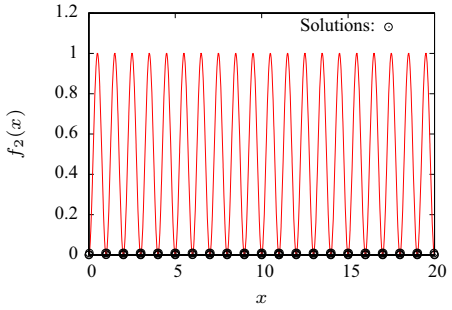


Figure 4. Solutions found for the sine function.

Table II
SOLUTION DISTRIBUTION OVER THE OPTIMUMS OF $f_2(x)$

Optimal position $\{x\}$	Solution percentage
$\{0\}$	3%
$\{1\}$	4%
$\{2\}$	5%
$\{3\}$	5%
$\{4\}$	5%
$\{5\}$	5%
$\{6\}$	4%
$\{7\}$	5%
$\{8\}$	5%
$\{9\}$	5%
$\{10\}$	5%
$\{11\}$	6%
$\{12\}$	5%
$\{13\}$	5%
$\{14\}$	5%
$\{15\}$	5%
$\{16\}$	5%
$\{17\}$	5%
$\{18\}$	5%
$\{19\}$	5%
$\{20\}$	3%

C. Double Pulse Function

In this section, a function is used with one parameter and two continuous optimal fronts in the parameter space:

$$f_3(x) = \begin{cases} 1, & 3.5 < x < 4.0 \text{ or } 16.0 < x < 16.5 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $0 \leq x \leq 20$.

Figure 5 shows that the algorithm finds a representative solution set of the non-dominated front in a single algorithm execution. Moreover, the final solutions are spread over the optimal parameter front. In this experiment, it is also considered a population with 100 individuals and 1000

Table III
SOLUTION DISTRIBUTION OVER THE OPTIMUMS OF $f_3(x)$

Optimal position $\{x\}$	$[3.5, 4.0]$	$[16.0, 16.5]$
Number of solutions	53%	47%

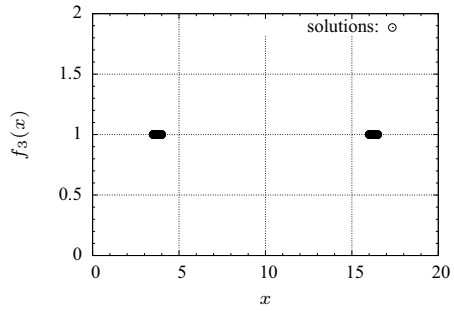


Figure 5. Solutions found for the double pulse function.

iterations. In this case the solutions found by the algorithm are divided between the pulses (see table III).

D. PR Function

The last function describes a real problem, where, the kinematic optimization of a Stewart-type parallel manipulator, based on maximum dexterity [2], is considered (Figure 6). The manipulator's mechanical structure comprises a fixed (base) platform and a moving platform, linked together by six independent, identical, open kinematic chains. Each chain comprises two links: the first link (linear actuator) is always normal to the base and has a variable length, l_i ($i = \{1, ..., 6\}$), with one of its ends fixed to the base and the other one attached, by a universal joint, to the second link; the second link (arm) has a fixed length, L, and is attached to the moving platform by a spherical joint.

The function to maximize is given by expression (5).

$$f_4(r_B, \phi_P, \phi_B, L) = \frac{\sigma_{\max}(\mathbf{J}_c)}{\sigma_{\min}(\mathbf{J}_c)} \tag{5}$$

where $\sigma_{\max}(\mathbf{J}_c)$ and $\sigma_{\min}(\mathbf{J}_c)$ represent the maximum and minimum singular values of the inverse kinematic jacobian matrix, $\mathbf{J}_C$ (6).

$$\mathbf{J}_C = \begin{bmatrix} \frac{(\mathbf{e}_1 - l_1\mathbf{z}_B)^T}{\mathbf{z}_B^T\mathbf{e}_1 - l_1} & \frac{({}^P\mathbf{p}_{1|_B} \times (\mathbf{e}_1 - l_1\mathbf{z}_B)^T)}{\mathbf{z}_B^T\mathbf{e}_1 - l_1} \\ \vdots & \vdots \\ \frac{(\mathbf{e}_6 - l_6\mathbf{z}_B)^T}{\mathbf{z}_B^T\mathbf{e}_6 - l_6} & \frac{({}^P\mathbf{p}_{6|_B} \times (\mathbf{e}_6 - l_6\mathbf{z}_B)^T)}{\mathbf{z}_B^T\mathbf{e}_6 - l_6} \end{bmatrix} \tag{6}$$

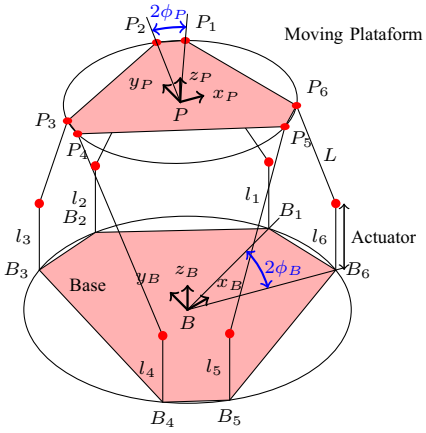


Figure 6. Schematic representation of the parallel manipulator structure.

This is a 6×6 matrix in which all vectors can be obtained using vector algebra applied to each parallel manipulator's kinematic chain. Vectors $\mathbf{e}_i$ are given by

$$\mathbf{e}_i = \mathbf{x}_{P(pos)} - \mathbf{b}_i + {}^P\mathbf{p}_{i|_B} \tag{7}$$

where $\mathbf{x}_{P(pos)}$ is the position of the moving platform expressed in the base frame, $\mathbf{b}_i$ represents the positions of the connecting points between kinematic chains and base and ${}^P\mathbf{p}_{i|B}$ are the positions of the points P_i connecting the kinematic chains to the moving platform, expressed in the base frame.

The scalars l_i ($i = \{1, \ldots, 6\}$) are the displacements of the actuators, given by

$$l_i = e_{iz} + \sqrt{L^2 - e_{ix}^2 - e_{iy}^2} \tag{8}$$

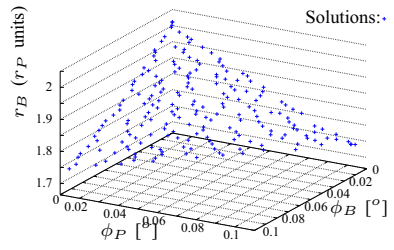
This work considers a particular manipulator pose (position and orientation), corresponding to the centre of the manipulator workspace *i.e.*, $[0\ 0\ 2\ 0\ 0\ 0]^T$ (units in r_P and degrees, respectively). Thus, for this pose, the jacobian matrix will be a function of the four kinematic parameters, that are sufficient to define the parallel manipulator's kinematic structure.

The parameters are codified by real values through the vector $p = [r_B\ \ \phi_P\ \ \phi_B\ \ L]^T$. The solutions are randomly initialized in the range $1.0 \leq r_B \leq 2.5$, $0^o \leq \phi_P, \phi_B \leq 25^o$ and $2.0 \leq L \leq 3.5$.

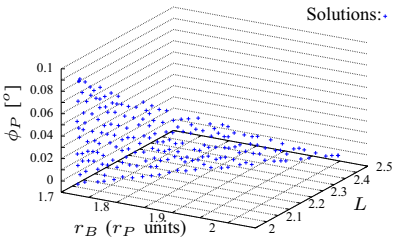
The ϵ is initialized with the value 20.0 and, during the evolution, its value is sucessively decreased until it reaches 0.03. This decrement is done at a rate of 10% every time the best 200 solutions are in the first rank and when ϵ has not changed in the last 100 generations.

The solutions are classified by the fitness function given by equation (5), in case the solution is admissible, otherwise it takes a very height value (1×10^{20}).

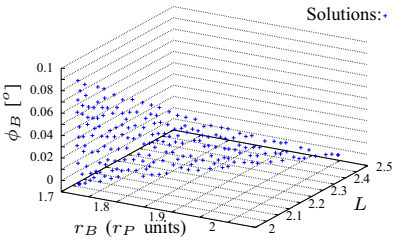
The global results (Figure 7 and Figure 8) show that there are multiple sets of solutions that are optimal. Moreover, the



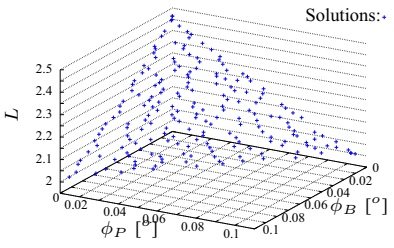
(a) Solutions projections in $\phi_P \times \phi_B \times r_B$ plan.



(b) Solutions projections in $r_B \times L \times \phi_P$ plan.



(c) Solutions projections in $r_B \times L \times \phi_B$ plan.



(d) Solutions projections in $\phi_P \times \phi_B \times L$ plan.

Figure 7. Optimal solutions set of kinematic parameters. (*cont.*)

algorithm draws a representative solution set of the optimal parameters front. As it can be seen in Figure 8, the final population set belongs to the best rank.

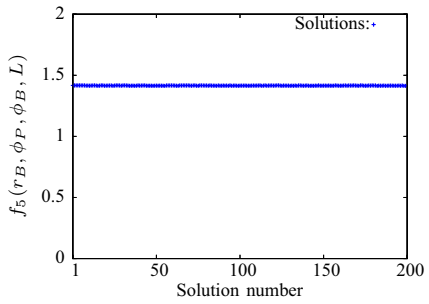


Figure 8. Fitness function values for all the optimal solutions.

IV. Performance Indices

To study the solution diversity the spacing index (SP) [15] and the minimal distance graph (MDG) [16] are employed. In this work, both functions uses the distances in parameter space instead of objective space as was initially proposed by their authors.

The spacing metric, which measures the range (distance) variance of neighboring vectors in the parameter front, is defined by (9a), where $\#S$ is the number of the considered solutions, $\overline{d}$ is the mean of all d_i distances and d_i, which is a measure according to the distance to its closest neighbor, is given by (9b). p is the number of problem parameters, and $s_{k,j}$ represents the parameter j of solution k.

$$SP(S) = \sqrt{\frac{1}{\#S-1}\sum_{i=1}^{\#S}(d_i - \overline{d})^2} \tag{9a}$$

$$d_i = \min_{s_k \in S \wedge s_k \neq s_i} \sum_{j=1}^{p} |s_{i,j} - s_{k,j}| \tag{9b}$$

The MDG performance index (10) is calculated based on the minimum weight spanning tree, where the weights are the minimum distances edges, d_i, that connect all the considered solutions of the population.

$$MDG(S) = \sqrt{\frac{1}{\#S-2}\sum_{i=1}^{\#S-1}(d_i - \overline{d})^2} \tag{10}$$

The resultant metric values, for the functions considered in section III are shown in table IV. The indices present small values indicating that the algorithm obtains a good solution spread. When the index MDG is used in the functions f_1, f_2 and f_3, the obtained values are greater than those evaluated with the SP index. This hapens since the MDG index uses, necessarily, the distances between the different niches.

Table V shows the median, the average, the standard desviation, the maximum, the minimum, the objective range and ϵ value for the considered functions. In all the optimization problems, the final solution set are spreaded in the best

Table IV
PERFORMANCE INDICES RESULTS.

Index	f_1	f_2	f_3	f_4
SP	0.001	0.002	0.003	0.005
MDG	0.948	0.371	1.205	0.003

Table V
ANALYSIS RESULTS.

Index	f_1	f_2	f_3	f_4
Median	0.007	0.003	1.000	1.416
Average	0.006	0.005	1.000	1.416
Std dev	0.004	0.004	0.000	0.001
Max	0.010	0.010	1.000	1.414
Min	0.000	0.000	1.000	1.417
Max - Min	0.010	0.010	0.000	0.003
ϵ	0.010	0.010	0.010	0.003

rank. This could be confirmed through the maximum and minimum values (Max - Min $\leq \epsilon$). For function f_3 all the solutions have the value 1 since it is a binary function and the rank value is 0.01. The solutions of the other functions are spreaded along the interval rank. The rank interval, ϵ, is fundamental to allow the movement of solutions along the front. Otherwise, they remain trapped in an optimal of the front.

The algorithm finds all the minimum for three functions (f_1, f_2, f_3) which have been commonly adopted as a benchmark in multi-objective evolutionary community.

For the practical problem, and since its optimums values are not known, is inferred that algorithm finds the function optimal.

V. CONCLUSIONS

A new spreading technique was developed with the purpose of solving a wide range of different single-objective optimization problems. To access the algorithm performance, it was firstly applied in benchmark functions. For these functions the algorihtm found a correct representative solution set. Then, it was used in a function obtained from a practical engineering problems, namely, in kinematic optimization of parallel robots. In all the test cases, the algorithm found in only one run several solutions with the same maximum performance but with different parameters values. The results show that the proposed algorithm solves different types of single-objective problems, maintaining a good spreading in parameter space.

REFERENCES

[1] R. Balling, "The maximin fitness function; multi-objective city and regional planning," in *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 8-11, 2003, Proceedings*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds., vol. 2632. Springer, 2003, pp. 1–15.

[2] M. R. Barbosa, E. J. Solteiro Pires, and A. M. Lopes, "Optimization of parallel manipulators using evolutionary algorithms," in *Soft Computing Models in Industrial and Environmental Applications, 5th International Workshop (SOCO 2010)*, Guimarães, Portugal, 16 -18 May 2010, pp. 79–86. [Online]. Available: http://www.springerlink.com/content/5429KX4R41Q435K7

[3] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, 1975.

[4] K. Deb, *Optimization for engineering design: Algorithms and examples*. New Delhi: Prentice Hall, 1995.

[5] ——, *Multi-Objective Optimization using Evolutionary Algorithms*, ser. Interscience Series in Systems and Optimization. John Wiley & Sons, 2001.

[6] K. Deb and S. Tiwari, "Omni-optimizer: A procedure for single and multi-objective optimization," in *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3410. Springer, 2005, pp. 47–61.

[7] ——, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4M4CN26-6/2/bdab4602ff1e930774e96c4e2955c179

[8] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multi-modal function optimisation," in *Proc of the 2nd Int. Conf. on Genetic Algorithms and Their Applications*, 1987, pp. 41–49.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison – Wesley, 1989.

[10] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Archiving with guaranteed convergence and diversity in multi-objective optimization," in *In Proceedings of the Genetic and Evolutionary Computation Conference Morgan Kaufmann Publishers*, 2002, pp. 439–447. [Online]. Available: citeseer.ist.psu.edu/laumanns02archiving.html

[11] X. Li, "Better spread and convergence: Particle swarm multi-objective optimization using the maximin fitness function," in *Proceeding of Genetic and Evolutionary Computation Conference 2004 (GECCO'04)*, ser. Lecture Notes in Computer Science, Springer-Verlag, K. e. a. Deb, Ed., vol. LNCS 3102, Seattle, USA, 26–30 June 2004, pp. 117–128.

[12] R. D. Luce and H. Raiffa, *Games and Decisions: Introduction and Critical Survey*. New York: John Wiley & Sons, Inc, 1957.

[13] L. M. Mendes, E. J. Solteiro Pires, J. Vaz, and M. J. Rosário, "Automated design of radio-frequency single-ended switched capacitor arrays using genetic algorithms," in *Proc Midwest Symp. on Circuits and Systems*. Montréal , Canada: IEEE Computer Society, August 2007, pp. 453–456.

[14] J. Rawls, *A Theory of Justice*. London: Oxford University Press, 1971.

[15] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, Massachusetts, May 1995.

[16] E. J. Solteiro Pires, P. B. de Moura Oliveira, and J. A. Tenreiro Machado, "Multi-objective maximin sorting scheme," in *Conference on Evolutionary Multi-criterion Optimization – EMO 2005*. Guanajuanto, México: Springer-Verlag, Lecture Notes in Computer Science Vol. 3410, Mar 2005, pp. 165–175.

[17] K. P. Yoon and C.-L. Hwang, *Multiple Attribute Decision Making: An Introduction*, ser. Quantitative Applications in the Social Sciences, S. Publications, Ed. SAGE Publications, 1995.