On-line Adaptation in Multi-Objective Particle Swarm Optimization

Margarita Reyes-Sierra and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Ingeniería Eléctrica, Sección Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07360, México
mreyes@computacion.cs.cinvestav.mx,ccoello@cs.cinvestav.mx

Abstract.- Tuning the parameters of any evolutionary algorithm is a difficult task. In this paper, we describe some experiments done in order to explore the impact of the main parameters of the particle swarm optimization algorithm, when using it for multi-objective optimization. These parameters are the inertia weight and the learning factors involved in the velocity update formula. Also, in our study, we included some parameters from our own multi-objective approach. As a result of our study, we propose three different mechanisms to adapt the values of those parameters that are found to be the most important for the performance of our approach. The mechanisms proposed are validated using seven different test functions taken from the specialized literature of multi-objective optimization. The obtained results show that it is possible to design on-line adaptation mechanisms able to maintain, and even improve, the quality of the obtained solutions, without increasing the computational cost.

1. INTRODUCTION

One of the main drawbacks of Evolutionary Algorithms (EAs) is that many of them have several parameters that have to be tuned each time we want to solve a different problem. In the majority of cases, the values of the parameters of a given EA affect the performance of the algorithm in a noticeable way. In the case of the Particle Swarm Optimization (PSO) algorithm, the main parameters are those involved in the velocity update formula: the inertia weight and the learning factors. Also, some PSO approaches incorporate a mutation operator in order to introduce some diversity into the swarm. In those cases, another parameter related with the mutation operator has to be tuned, too.

In this paper, we describe a study of the parameters of a Multi-Objective Particle Swarm Optimizer (MOPSO) approach, that is part of our previous work. In such study, we included the parameters involved in the velocity update formula, but also the parameters of our MOPSO approach. Since such study was performed in order to find the parameters that have the greatest impact on the performance of the MOPSO approach, as a result we propose three different mechanisms to adapt the values of the parameters that were found to be most important. The proposed mechanisms are tested using seven different functions taken from the specialized literature of evolutionary multi-objective optimization.

This paper is organized as follows. Section 2 provides a brief introduction to the PSO algorithm. In Section 3, we describe our MOPSO approach. Section 4 describes the study of the parameters done and presents the proposed approaches. The obtained results and the corresponding discussion are provided in Sections 5 and 6, respectively. Finally, we define our conclusions and future work in Section 7.

2. PARTICLE SWARM OPTIMIZATION

Kennedy and Eberhart [4] initially proposed the swarm algorithm for optimization. The particle swarm optimization (PSO) algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. In PSO, individuals, referred to as particles, are "flown" through search space. Changes to the position of the particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals. A swarm consists of a set of particles, where each particle represents a potential solution. The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{x}_i(t)$ denote the position of particle p_i, at time step t. The position of p_i is then changed by adding a velocity $\vec{v}_i(t)$ to the current position, i.e.:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \quad (1)$$

The velocity vector drives the optimization process and reflects the socially exchanged information. In the global best version (used here) of PSO, each particle uses its history of experiences in terms of its own best solution thus far (*pbest*) but, in addition, the particle uses the position of the best particle from the entire swarm (*gbest*). Thus, the velocity vector changes in the following way:

$$\vec{v}_i(t) = W\vec{v}_i(t-1) + C_1 r_1 (\vec{x}_{pbest_i} - \vec{x}_i(t)) + C_2 r_2 (\vec{x}_{gbest_i} - \vec{x}_i(t)) \quad (2)$$

where W is the inertia weight, C_1 and C_2 are the cognitive and social learning factors (usually defined as constants), and $r_1, r_2 \in [0, 1]$ are random values.

```
Begin
    Initialize swarm. Initialize leaders.
    Send leaders to ε-archive
    crowding(leaders), g = 0
    While g < gmax
        For each particle
            Select leader.
            Update position (flight).
            Mutation.
            Evaluation.
            Update pbest.
        EndFor
        Update leaders
        Send leaders to ε-archive
        crowding(leaders), g = g + 1
    EndWhile
    Report results in ε-archive
End
```

Figure 1. Pseudocode of our algorithm.

3. MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION

The first part of our work consists of the study of the parameters of a MOPSO approach that is part of our previous work. The first version of our MOPSO algorithm was presented in [9] and updated in [8]. The MOPSO studied in this paper is based on Pareto dominance, since it considers every nondominated solution as a new leader (a leader is used as the *gbest* solution in Equation (2)). Additionally, the approach also uses a crowding factor [2] as a second discrimination criterion which is also adopted to filter out the list of available leaders. For each particle, we select the leader in the following way: 97% of the time a leader is randomly selected, if and only if that leader dominates the current particle, and, the remaining 3% of the time, we select a leader by means of a binary tournament based on the crowding value of the available set of leaders. If the size of the set of leaders is greater than the maximum allowable size, only the best leaders are retained based on their crowding value. We also incorporated in the proposed approach the use of different mutation (or *turbulence*) operators which act on different subdivisions of the swarm. We proposed a scheme by which the swarm is subdivided in three parts (of equal size): the first sub-part has no mutation at all, the second sub-part uses uniform mutation and the third sub-part uses non-uniform mutation. The available set of leaders is the same for each of these sub-parts. Finally, the proposed approach also incorporates the ε-dominance concept [5] to fix the size of the set of final solutions produced by the algorithm. Figure 1 shows the pseudo-code of our proposed approach.

3.1. Parameters of our MOPSO approach

As we mentioned before, the selection of a leader in our MOPSO approach is done by means of a combination of two different techniques: (1) by dominance and (2) by crowding values. In this way, our approach chooses technique (1) with certain probability (called Ps), otherwise, it chooses technique (2). After an extensive series of experiments, we fixed the value of Ps to 0.97. However, Ps is itself a parameter of our approach. On the other hand, since we define a scheme by which two different mutation operators are used into the swarm, a mutation probability Pm is needed. However, in order to avoid the definition of extra parameters for the mutation operators, we adopt a rule of thumb normally used in the EA literature [1]: the mutation rate is defined as $1/codesize$, where $codesize$ refers to the total length of the string that encodes all the decision variables of the problem (the number of variables in our case).

In order to perform the flight of each particle, the changes to the velocity vector are done using Equations (1) and (2), corresponding to the global best version of PSO. However, in Equation (2) we use: $W = random(0.1, 0.5)$, $C_1, C_2 = random(1.5, 2.0)$, and $r_1, r_2 = random\ (0.0, 1.0)$. Note that most of the previous PSO proposals fix the values of W, C_1 and C_2 instead of using random values as in our case. The only exception that we know (in the specific case of MOPSOs) is some of our own previous work [10]. We adopted this scheme since we found it as a more convenient way of dealing with the difficulties of fine tuning the parameters W, C_1 and C_2 for each specific test function. However, W, C_1 and C_2 can be considered parameters of our approach.

In the first version of our MOPSO approach, for each problem, the value of ε was tuned based on the desired number of points in the final Pareto front. Nevertheless, the current version of our approach already includes a relatively simple mechanism to adapt the value of the parameter ε throughout the run. The mechanism implemented is based on the formula [5]: $PF \leq \left(\frac{\log K}{\log(1+\varepsilon)}\right)^{m-1}$, where PF is the number of points desired in the final Pareto front, K is an upper bound for all the objective functions (calculated over the current values) and m is the number of objective functions. Our approach starts the value of ε using the previous formula. Later on, the value of ε is adjusted based on the number of points that the current Pareto front contains. Since large values of ε produce Pareto fronts with few solutions, the value of ε increases or decreases depending on the current number of solutions in the Pareto front. The change in the value of ε is proportional to the difference between the desired and the current number of optimal solutions.

Thus, the parameters of our approach that are going to be considered in our study are: $swarmsize$ (size of the swarm), $gmax$ (number of iterations), Ps (selection probability) and W, C_1 and C_2 (velocity update formula).

4. DESCRIPTION OF OUR WORK

4.1. ANOVA

With the aim of exploring which parameters were the most important for the performance of our approach, we performed an Analysis of Variance (ANOVA). For such study, we considered 3 different levels for each one of the parameters of our approach:

$swarmsize$: 50, 100, 200.
$gmax$: 50, 100, 200.
Ps: 0.3, 0.6, 0.97.
W: 0.1, 0.5, *random*(0.1,0.5).
C_1, C_2: 1.5, 2.0, *random*(1.5,2.0).

In this way, the total number of combinations was: 729. We performed 30 runs for each combination and for each one of the seven following test functions (21870 runs per function, 153090 runs in total):

Test Function ZDT1 [13]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
$f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
$g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$, $h(f1, g) = 1 - \sqrt{f_1/g}$
where $m = 30$, and x_i ∈[0,1].

Test Function ZDT2 [13]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
$f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
$g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$,
$h(f1, g) = 1 - (f_1/g)^2$
where $m = 30$, and x_i ∈[0,1].

Test Function ZDT3 [13]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
$f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
$g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$,
$h(f1, g) = 1 - \sqrt{f_1/g} - (f_1/g)sin(10\pi f_1)$
where $m = 30$, and x_i ∈[0,1].

Test Function ZDT4 [13]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}))$
$f_1(\vec{x}) = x_1$, $f_2(\vec{x}) = g(\vec{x})h(f_1, g)$
$g(\vec{x}) = 1+10(m-1)+\sum_{i=2}^{m}(x_i^2-10cos(4\pi x_i))$, $h(f1, g) = 1 - \sqrt{f_1/g}$
where $m = 10$, x_1 ∈[0,1] and x_i ∈[-5,5], $i = 2, ..., m$.

Test Function DTLZ2 [3]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}))$
$f_1(\vec{x}) = (1 + g(\vec{x}))cos(x_1\pi/2)cos(x_2\pi/2)$
$f_2(\vec{x}) = (1 + g(\vec{x}))cos(x_1\pi/2)sin(x_2\pi/2)$
$f_3(\vec{x}) = (1 + g(\vec{x}))sin(x_1\pi/2)$
$g(\vec{x}) = \sum_{i=3}^{m}(x_i - 0.5)^2$ where $m = 12$ and x_i ∈[0,1].

Test Function DTLZ4 [3]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}))$
$f_1(\vec{x}) = (1 + g(\vec{x}))cos(x_1^\alpha\pi/2)cos(x_2^\alpha\pi/2)$
$f_2(\vec{x}) = (1 + g(\vec{x}))cos(x_1^\alpha\pi/2)sin(x_2^\alpha\pi/2)$
$f_3(\vec{x}) = (1 + g(\vec{x}))sin(x_1^\alpha\pi/2)$
$g(\vec{x}) = \sum_{i=3}^{m}(x_i-0.5)^2$ where α=100, $m = 12$ and x_i ∈[0,1].

Test Function DTLZ6 [3]:
Minimize$(f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}))$
$f_1(\vec{x}) = x_1, f_2(\vec{x}) = x_2$
$f_3(\vec{x}) = (1 + g(\vec{x}))h(f_1, f_2, g)$
$g(\vec{x}) = 1 + 9/(m-2)\sum_{i=3}^{m} x_i$,
$h(f_1, f_2, g) = 3 - \sum_{i=1}^{2}[\frac{f_i}{1+g}(1 + sin(3\pi f_i))]$
where $m = 22$ and x_i ∈[0,1].

In previous studies [8], we have used several measures of performance. However, for the study presented in this paper, we selected the measure that empirically has reflected the changes on quality in a better way:
Success Counting (SCC): This measure counts the number of vectors, in the current set of nondominated vectors available, that are members of the Pareto optimal set: $SCC = \sum_{i=1}^{n} s_i$, where n is the number of vectors in the current set of nondominated vectors available; s_i = 1 if vector i is a member of the Pareto optimal set, and $s_i = 0$ otherwise.

The ANOVA provided the following conclusions:

- $swarmsize$: large values give better results.
- $gmax$: as $swarmsize$, large values give better results, although at a lower rate.

In fact, form the ANOVA we were able to conclude that it is better to use a large swarm size than a high number of generations.

- Ps: except in function DTLZ4, large values give better results.
- W: except in function DTLZ2, large values give better results.
- C_1: this parameter is not important for our MOPSO approach, that is, its value doesn't affect the performance of the algorithm.
- C_2: except for function DTLZ2, large values give better results.

As we can see, the parameters Ps, W and C_2 are the most important for our approach, since their values affect the performance of the algorithm. In this way, we performed another ANOVA in which the value of the parameters $swarmsize$, $gmax$ and C_1 were fixed at: 200, 100 and *random*

(1.5,2.0), respectively. In this case, we considered the following levels for the parameters that were found to be the most important:
Ps: 0.0, 0.2, 0.4, 0.6, 0.8, 0.97, 1.0 (7 levels).
W: 0.1, 0.2, 0.3, 0,4, 0.5, *random* (6 levels).
C_2: 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, *random* (7 levels).

In this way, this time we had a total of 294 combinations. We performed 30 runs for each combination and for each one of the seven test functions previously defined (8820 runs per function, 61740 runs in total). As in the case of the previous ANOVA, we obtained very similar conclusions. However, in this case, we were able to obtain the values that gave the best performance of our MOPSO approach:

Ps: 0.2, 0.8, 0.97.
W: 0.2, 0.4, 0.5.
C_2: 1.5, 2.0.

4.2. Proposed Approaches

As we observed in the previous section, from the two ANOVAs performed we obtained a set of values for the most important parameters of our MOPSO approach, that give the best performance. Since we have a finite set of possible values for the parameters Ps, W and C_2, we will consider the problem of the selection of the best value for each parameter as a multi-armed bandit problem. The multi-armed bandit problem was originally described by Robbins [7]. A multi-armed bandit, also called K-armed bandit, is similar to a traditional slot machine (one-armed bandit) but, in general, has more than one lever. Initially, the gambler has no knowledge about the levers, but through repeated trials, he can focus on the most rewarding levers. There are two versions of the K-armed bandit, the *opaque* in which a unique reward is observed at each round, and the *transparent* in which all rewards are observed. The gambler plays iteratively one lever at each round and observes the associated reward. His objective is to maximize the sum of the collected rewards. The problem of determining the best strategy for the gambler is called the multi-armed bandit problem and many strategies or algorithms have been proposed as solutions to this problem. In our case, each lever corresponds to one of the possible values for the parameters that we want to adapt. Each time the MOPSO approach selects a specific value (a lever) for a parameter, that value (lever) receives a reward. We define two types of reward:
Reward 1: If the obtained particle is able to enter into the set of leaders, the lever receives a reward of 1. Otherwise, the lever receives a reward of 0.
Reward 2: If the obtained particle is able to enter into the set of leaders, the lever receives a reward of 0.5. Furthermore, if the obtained particle is able to dominate at least one member of the set of leaders, the lever receives an additional reward of 0.5. Otherwise, the lever receives a reward of 0.

All the levers, for the three parameters (Ps, W and C_2), start with a total reward of zero. The first generation (zero) of the MOPSO approach is used to obtain initial values for the mean reward for each lever, of each parameter. That is, any specific strategy is applied beginning with generation 1.

We use three different mechanisms to adapt the corresponding values of the studied parameters. The first is proposed by us, and the other two were chosen based on the work presented in [11] [1]:

Proportional. This strategy consists of a random choice according to the probability:$p_k = \frac{\mu_k}{\sum_{i=1}^{n} \mu_i}$ where μ_i is the estimated mean of the rewards brought by the lever i and n is the total number of levers.

The ϵ-Greedy Strategy. This is probably the simplest and the most widely used strategy to solve the bandit problem as it was described by Watkins [12]. The ϵ-greedy consist of choosing a random lever with ϵ-frequency, and otherwise choosing the lever with the highest estimated mean. ϵ must be in the open interval (0,1) and its choice is left to the user. The ϵ value controls the amount of exploration (the probability of executing actions other than the one with the highest estimated mean). In this way, the ϵ-greedy strategy is sub-optimal because asymptotically, the constant factor ϵ prevents the strategy from getting arbitrarily close to the optimal lever.

The Soft Max Strategy. This strategy, also called Boltzmann Exploration [6], consists of a random choice according to a Gibbs distribution. The lever k is chosen with probability $p_k = \frac{e^{\mu_k/\tau}}{\sum_{i=1}^{n} e^{\mu_i/\tau}}$ where μ_i is the estimated mean of the rewards brought by the lever i and $\tau \in \mathbb{R}^+$ is a parameter called the *temperature*. The choice of τ's value is left to the user. The parameter τ has an impact similar to ϵ. Small values of τ increase the tendence to choice the lever with the best estimated mean.

5. RESULTS

We have tested the proposed mechanisms using the seven test functions previously defined. For the ϵ-greedy and the soft-max approach, the allowable values of ϵ and τ are $\{0.05, 0.10, 0.15\}$, based on the work presented in [11] and given the impact of these parameters (briefly discussed in the previous section). We performed 30 runs for each function and each mechanism. The parameters adopted for the MOPSO algorithm were: 200 particles, 100 generations and 100 points in the final Pareto front. The obtained results are shown in Tables 1 to 7. For each test function, we present the best,

[1] In [11], Vermorel et al. provided the first preliminary empirical evaluation of several multi-armed bandit algorithms.

median, worst, mean and standard deviation of the SCC measure, for the MOPSO approach without adaptation and the MOPSO approach with adaptation, with the different mechanisms and the two types of reward.

6. DISCUSSION OF RESULTS

Function ZDT1. As we can see in Table 1, all the approaches obtained very good results with both types of reward. However, only the proportional strategy was able to improve the results of the approach without adaptation, by almost 10 particles (on average), using reward 2. It is very interesting to note that in all cases, the approaches with adaptation improved the worst results of the approach without adaptation (except for softmax, whose quality was the same when $\tau = 0.1$). In this way, the standard deviation of the approaches with adaptation was smaller than for the approach without adaptation. In general, the ϵ-greedy and softmax strategies, had a better performance (on average) using reward 1, being the softmax better than the ϵ-greedy strategy. Also, we can observe that the ϵ-greedy strategy lost quality when using reward 2.

Function ZDT2. As in function ZDT1, in this function the proportional approach (using reward 2) obtained the best results having the same quality than the results of the approach without adaptation (see Table 2). In this case, the softmax strategy had a better performance using reward 2, being better than the ϵ-greedy strategy (whose best results were obtained using reward 1). As in the previous function, we can observe again that the the ϵ-greedy strategy lost quality when using reward 2.

Function ZDT3. From Table 3, we can conclude that the proportional approach obtained again the best results using reward 2 and also slightly improved the results obtained by the approach without adaptation. In this case, the ϵ-greedy and the softmax strategies had a best performance using reward 1, being ϵ-greedy marginally better. Finally, we can observe that the ϵ-greedy strategy lost quality again when using reward 2.

Function ZDT4. As we can see in Table 4, in this function all the approaches had a very similar behavior. All the adaptation strategies improved the results obtained by the approach without adaptation or at least reached the same quality. The only exception was the ϵ-greedy strategy when ϵ=0.05 and it used reward 2. Also, this same case was the only that didn't improve the worst results obtained by the approach without adaptation. In this case, the best results were from the softmax strategy using reward 1. However, in this case there are no important differences in the performance of the approaches, when compared with respect to the type of reward used.

Function DTLZ2. In this function (see Table 5) all the adaptation strategies had again very similar behavior. However, only the softmax strategy (using reward 1) was able to improve the results obtained by the approach without adaptation. In this case, all the strategies had better results using reward 1, improving the worst results from the approach without adaptation, in all cases.

Function DTLZ4. As in function ZDT4, in this case all the adaptation strategies improved the results obtained by the approach without adaptation or at least reached the same quality (see Table 6). Also, all the approaches had their best performance using reward 1, being the best the ϵ-greedy strategy, in this case. It is very interesting to note that, in this function, all the adaptation strategies improved significantly the best result obtained by the approach without adaptation, specially in the case of the ϵ-greedy strategy (almost 70 particles, when ϵ=0.05 and using reward 1). However, since the median and worst values were not improved, the standard deviations were greater in this case.

Function DTLZ6. As we can see in Table 7, in this function it is again reward 1 the one that provided the best results, for all the adaptation strategies. The best results, in this case, were obtained by the softmax strategy. In this function, the softmax strategy improved the best and median results obtained by the approach without adaptation. However, as in function DTLZ4, since the worst values were not improved, the standard deviations were greater in this case. As we observe, the proportional strategy obtained very good results in three of the seven test functions, using reward 2: ZDT1, ZDT2 and ZDT3. However, this behavior was not consistent. In general, reward 1 provided better results than reward 2, specially in the case of the ϵ-greedy strategy. The obtained results seem to indicate that the knowledge about the ability of a set of parameters to provide a nondominated particle is enough information to reward it. In general, the softmax strategy had better performance than the proportional and the ϵ-greedy strategies. In fact, unlike the case of the two other approaches (specially the ϵ-greedy), the results of the softmax strategy were not significantly affected by the use of different rewards. Also, the softmax strategy was able to improve the performance (on average) of the approach without adaptation in five of the seven test functions, and to maintain the quality of the obtained solutions in one more test function. Finally, for all the test functions, at least one of the three proposed strategies was able to improve the performance (on average) of the approach without adaptation.

7. CONCLUSIONS AND FUTURE WORK

When using evolutionary algorithms to solve optimization problems, one of the main difficulties is to define the values of the corresponding parameters, that provide the best performance. In this paper, we proposed three different strategies to adapt the values of the parameters of a multi-objective particle swarm optimizer previously proposed by the authors.

Table 1. OBTAINED RESULTS FOR FUNCTION ZDT1

Function ZDT1								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	$\epsilon = 0.05$	$\epsilon = 0.10$	$\epsilon = 0.15$	$\tau = 0.05$	$\tau = 0.10$	$\tau = 0.15$
best	100	99	99	98	99	99	100	97
median	86	82	82	81	85	84	83	85
worst	15	51	30	18	20	48	58	62
mean	84	81	78	80	82	82	82	84
st.dev.	17.5	9.1	17.7	15.0	15.3	11.9	9.9	9.5
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	$\epsilon = 0.05$	$\epsilon = 0.10$	$\epsilon =0.15$	$\tau =0.05$	$\tau =0.10$	$\tau =0.15$
best	100	100	98	93	93	97	98	96
median	86	95	76	72	77	82	81	83
worst	15	54	19	27	29	49	15	60
mean	84	92	70	68	72	82	80	82
st.dev.	17.5	9.8	19.6	17.2	17.3	9.9	15.8	9.7

Table 2. OBTAINED RESULTS FOR FUNCTION ZDT2

Function ZDT2								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	$\epsilon =0.05$	$\epsilon = 0.10$	$\epsilon =0.15$	$\tau =0.05$	$\tau =0.10$	$\tau =0.15$
best	100	100	100	100	100	100	100	100
median	96	93	93	94	95	90	93	92
worst	60	16	23	58	45	34	30	0
mean	94	89	84	89	89	83	86	87
st.dev.	7.8	15.9	18.4	10.6	13.4	15.4	16.9	18.1
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	$\epsilon = 0.05$	$\epsilon =0.10$	$\epsilon =0.15$	$\tau =0.05$	$\tau =0.10$	$\tau =0.15$
best	100	100	100	100	100	100	100	100
median	96	98	82	92	91	93	94	95
worst	60	54	12	0	22	24	44	1
mean	94	94	76	77	81	90	87	88
st.dev.	7.8	9.5	23.7	28.2	20.2	13.9	15.5	18.6

Table 3. OBTAINED RESULTS FOR FUNCTION ZDT3

Function ZDT3								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	$\epsilon =0.05$	$\epsilon =0.10$	$\epsilon =0.15$	$\tau =0.05$	$\tau =0.10$	$\tau =0.15$
best	95	92	90	88	92	93	91	91
median	83	74	74	78	76	74	78	74
worst	63	9	41	51	25	30	16	39
mean	81	69	74	75	72	74	74	72
st.dev.	9.8	19.5	10.8	9.7	14.8	16.5	15.7	10.6
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	$\epsilon =0.05$	$\epsilon =0.10$	$\epsilon =0.15$	$\tau =0.05$	$\tau =0.10$	$\tau =0.15$
best	95	97	83	81	83	92	91	93
median	83	86	63	60	60	75	76	72
worst	63	46	13	22	21	32	33	25
mean	81	83	59	56	55	73	73	68
st.dev.	9.8	12.5	15.7	14.5	17.5	11.5	14.8	18.9

Table 4. OBTAINED RESULTS FOR FUNCTION ZDT4

Function ZDT4								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	99	100	100	100	100	100	99	99
median	97	97	97	97	95	99	97	97
worst	73	94	81	88	85	86	89	89
mean	95	97	95	96	95	98	97	96
st.dev.	5.2	1.5	4.4	3.4	3.7	2.7	2.1	2.3
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	99	98	100	100	100	100	100	100
median	97	95	97	98	98	97	97	97
worst	73	84	0	89	91	84	88	88
mean	95	94	92	97	97	96	96	96
st.dev.	5.2	2.8	17.9	2.7	2.6	3.2	2.9	2.6

Table 5. OBTAINED RESULTS FOR FUNCTION DTLZ2.

Function DTLZ2								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ = 0.15	τ =0.05	τ =0.10	τ =0.15
best	50	41	38	48	38	51	40	47
median	28	29	28	26	27	32	27	30
worst	9	13	12	11	16	14	15	10
mean	30	28	27	26	26	32	27	28
st.dev.	9.6	7.4	6.9	8.1	5.9	9.5	6.9	8.4
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	50	34	31	35	37	47	42	43
median	28	22	20	17	18	28	28	30
worst	9	8	8	5	5	7	13	8
mean	30	21	20	19	19	28	29	28
st.dev.	9.6	6.2	6.7	8.6	6.8	9.3	7.8	7.7

Table 6. OBTAINED RESULTS FOR FUNCTION DTLZ4.

Function DTLZ4								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	17	58	86	51	77	44	63	27
median	9	9	10	10	12	10	10	12
worst	1	1	1	1	2	1	1	2
mean	9	11	18	11	17	11	14	12
st.dev.	4.7	10.9	22	11.3	17.4	8.3	13.5	4.4
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	17	19	49	75	83	26	61	58
median	9	9	8	7	9	8	9	11
worst	1	2	1	0	0	0	0	1
mean	9	8	11	9	11	9	11	13
st.dev.	4.7	3.9	12.3	13.9	15.8	6.1	13.7	10.2

Table 7. OBTAINED RESULTS FOR FUNCTION DTLZ6.

Function DTLZ6								
SCC	without	with adaptation and reward 1						
measure	adaptation	Prop.	ϵ =0.05	ϵ = 0.10	ϵ =0.15	τ =0.05	τ =0.10	τ =0.15
best	97	96	100	92	100	98	100	100
median	64	62	66	56	59	68	71	75
worst	43	18	38	23	29	33	38	27
mean	67	62	68	57	60	69	73	71
st.dev.	14.9	14.9	16.6	18.5	16.1	21.1	17.1	21.5
SCC	without	with adaptation and reward 2						
measure	adaptation	Prop.	ϵ =0.05	ϵ =0.10	ϵ =0.15	τ =0.05	τ =0.10	τ = 0.15
best	97	94	96	96	96	100	100	97
median	64	68	54	49	46	72	73	76
worst	43	27	9	2	6	18	21	14
mean	67	65	52	48	51	69	67	69
st.dev.	14.9	17.7	21.9	25.8	22.2	22.6	23.2	21.9

First of all, we described the study performed to explore which of the parameters of our multi-objective approach were the most important. From such study, we obtained a finite set of values for each parameter, that provided the best results. In this way, we proposed three different mechanisms inspired in the multi-armed bandit problem, to adapt the value of the parameters of our approach. We used seven different test functions taken from the specialized multi- objective optimization literature to validate our proposals. For all the test functions, at least one of the three proposed strategies was able to improve the performance of the approach without adaptation. In fact, one of the proposed strategies (Soft Max) was able to improve the performance of the approach without adaptation in five of the seven test functions, and was able to maintain the quality of the obtained solutions in one more. In this way, we may conclude that it is possible to design on-line adaptation mechanisms able to maintain (and even improve) the quality of the solutions, with the same computational cost. As part of our future work, we plan to design better mechanisms to adapt the values of the parameters of our multi-objective approach and to test such strategies on different evolutionary algorithms.

Acknowledgments. The first author acknowledges support from CONACyT for granting her a scholarship to pursue graduate studies at CINVESTAV-IPN. The second author acknowledges support from CONACyT project number 42435-Y.

8. REFERENCES

[1] Thomas Bäck, editor. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[2] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[3] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *Congress on Evolutionary Computation*, volume 1, pages 825–830. IEEE Service Center, 2002.

[4] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *1995 IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Service Center, 1995.

[5] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.

[6] D. Luce. *Individual Choice Behavior*. Wiley, 1959.

[7] H. Robbins. Some aspects of the sequencial design of experiments. *Bulletin of the American Mathematical Society*, 55:527–535, 1952.

[8] Margarita Reyes Sierra and Carlos A. Coello Coello. Fitness inheritance in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 116–123. IEEE Service Center, 2005.

[9] Margarita Reyes Sierra and Carlos A. Coello Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In *EMO'2005*, pages 505–519. LNCS 3410, Springer-Verlag, 2005.

[10] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using clustering techniques to improve the performance of a particle swarm optimizer. In *Proceedings of GECCO'2004*, pages 225–237. Springer-Verlag, LNCS Vol. 3102, 2004.

[11] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European Conference on Machine Learning*, pages 437–448. LNAI 3720, Springer-Verlag, 2005.

[12] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[13] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.