Use of Cooperative Coevolution for Solving Large Scale Multiobjective Optimization Problems

Luis Miguel Antonio and Carlos A. Coello Coello
Computer Science Department
CINVESTAV-IPN (Evolutionary Computation Group)
Av. IPN No. 2508, Col. San Pedro Zacatenco, Mexico City .07300, Mexico.
lmiguel@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

***Abstract*—Many real-world multi-objective optimization problems have hundreds or even thousands of decision variables, which contrast with the current practice of multi-objective metaheuristics whose performance is typically assessed using benchmark problems with a relatively low number of decision variables (normally, no more than 30). In this paper, we propose a cooperative coevolution framework that is capable of optimizing large scale (in decision variable space) multi-objective optimization problems. We adopt a benchmark that is scalable in the number of decision variables (the ZDT test suite) and compare our proposed algorithm with respect to two state-of-the-art multi-objective evolutionary algorithms (GDE3 and NSGA-II) when using a large number of decision variables (from 200 up to 5000). The results clearly indicate that our proposed approach is effective as well as efficient for solving large scale multi-objective optimization problems.**

I. Introduction

In the real word there are many problems that require the optimization of two or more objective functions at the same time. These are known as multi-objective optimization problems (MOPs), and their solution involves finding the best possible trade-offs among the objective functions being optimized. This set of solutions is called the *Pareto optimal set*, and their corresponding objective function values form the so-called *Pareto front*.

MOPs have been solved during many years, using mathematical programming techniques [1]. However, the fact that a wide variety of MOPs in real-world applications tend to be nonlinear, and perhaps even non-differentiable, has contributed to the remarkable increase that the use of metaheuristics has experienced, mainly in the last 10 years. From the many metaheuristics in current use, Evolutionary Algorithms (EAs) are the most popular in the specialized literature. Multi-objective evolutionary algorithms (MOEAs) have the advantage of being population-based, which allows them to generate several elements of the Pareto optimal set in a single run, whereas mathematical programming techniques tend to produce a single element per run.

The motivation of this work is that many real-world problems have hundreds or even thousands of decision variables, which constrasts with the current practice of multi-objective metaheuristics that validate their performance using benchmark problems such as the Zitzler-Deb-Thiele (ZDT) [2], the Deb-Thiele-Laumanns-Zitzler (DTLZ) [3], and the Walking-Fish-Group (WFG) test problems [4], which are normally adopted with a relatively low number of decision variables (usually, adopting a maximum of up to 30 variables). In fact, scalability in decision variable space is a topic that has been only scarcely studied in the context of multi-objective optimization using metaheuristics.

To the best of the authors' knowledge, no MOEA has been designed so far, with the explicit goal of being able to deal with a very large number of decision variables. This is perhaps motivated by the fact, that most researchers assume that the currently available MOEAs should be able to work properly with a large number of decision variables. However, there is empirical evidence that indicates that most of the currently available multi-objective metaheuristics significantly decrease their efficacy as the number of decision variables of the MOP increases [5], [6]. Here, we propose a cooperative coevolutionary framework that allows a MOEA to deal with a large number of decision variables. The reason for using such a framework, is because there is evidence that indicates that cooperative coevolutionary schemes have been found to be effective for solving large scale global optimization problems [7]. Interestingly, cooperative coevolutionary MOEAs have been proposed before (see for example [8], [9]), but none of them has been designed with the explicit purpose of solving large scale MOPs, as we do in this paper.

II. Basic Concepts

In the following definitions we are assuming, without loss of generality, the minimization of all the objectives.

We are interested in solving problems of the type:

$$\text{minimize} \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \quad (1)$$

subject to:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, \ldots, m \quad (2)$$

$$h_i(\vec{x}) = 0 \qquad i = 1, 2, \ldots, p \quad (3)$$

where k is the number of objective functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $gi, hj : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, ..., m, j = 1, ..., p$ are the constraint functions of the problem and $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ the vector of decision variables. We thus wish to determine from the set Ω (where Ω is the feasible region) of all the vectors that satisfy (2) and (3) to the vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ that are *Pareto*

optimal. To describe the concept of optimality that we will adopt, we need to introduce a few additional definitions.

Pareto Optimality: We say that a vector of decision variables $\vec{x^*} \in \Omega$ (where Ω is the feasible region) is Pareto-optimal if $\forall \vec{x} \in \Omega \wedge \forall i \in \{1, \ldots, k\}$:

$$f_i(\vec{x}) = f_i(\vec{x^*}) \vee \nexists i \in \{1, \ldots, k\} : f_i(\vec{x}) < f_i(\vec{x^*}) \quad (4)$$

Pareto Dominance: A vector $\vec{u} = [u_1, \ldots, u_k]^T$ is said to dominate another vector $\vec{v} = [v_1, \ldots, v_k]^T$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, i.e.,:

$$\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i \quad (5)$$

Pareto Optimal Set: For a given MOP $\vec{f}(\vec{x})$, the Pareto Optimal Set P^* is defined by:

$$P^* := \{\vec{x} \in \Omega | \nexists \vec{x'} \in \Omega, \vec{f}(\vec{x'}) \preceq \vec{f}(\vec{x})\} \quad (6)$$

Pareto Front: For a given MOP $\vec{f}(\vec{x})$ and its Pareto optimal set P^*, the Pareto front PF^* is defined by:

$$PF^* := \{\vec{f}(\vec{x}), \vec{x} \in P^*\} \quad (7)$$

III. Previous Related Work

In the multi-objective research community, parameter scalability has been rarely considered before, in contrast with objective function scalability (the so-called *many-objective optimization*) which has been a hot research topic in recent years [10], [11]. The only work in this direction that we have found in the specialized literature is the small study presented in [12] in which ZDT1 is solved with up to 100 decision variables.

Regarding studies on parameter scalability in MOEAs, the most significant ones that we are aware of are those reported in [5], [6], in which eight multi-objective metaheuristics are analyzed. Such metaheuristics include three genetic algorithms (GAs) (NSGA-II, SPEA2, and PESA-II), an evolution strategy (PAES), a PSO (OMOPSO), a cellular GA (MOCell), a Differential Evolution algorithm (GDE3) and a Scatter Search algorithm (AbYSS). All of these approaches are representative of the state-of-the-art in evolutionary multi-objective optimization, and were studied when solving a benchmark of parameter-wise scalable problems (the ZDT test suite). The authors analyzed the behavior of these eight multi-objective metaheuristics when using a number of decision variables that ranged from 8 up to 2048. The hypervolume indicator was adopted to define a stopping criterion, and the study paid particular attention to the computational effort required by each algorithm for reaching the true Pareto front of each problem. These papers provided empirical evidence of the decrease in efficacy that multi-objective metaheuristics have when increasing the number of decision variables. These defficiencies are precisely the main motivation of the work, in which our aim is to present a framework that allows to use a MOEA for large-scale multi-objective optimization.

Although parameter scalability is a topic that has been only scarcely studied in the literature on evolutionary multi-objective optimization, large-scale optimization has been the focus of an important amount of research in global (single-objective) optimization using evolutionary algorithms, and we relied on such studies to develop our proposal. The currently available approaches for large-scale global optimization can be roughly divided in two groups: (1) those that decompose a high-dimensional objective vector into small subcomponents which can then be handled by conventional EAs, and (2) those that approach the problem by disturbing the population of the EA or by combining different evolutionary methods [7]. From these methods, cooperative coevolution has been one of the most successful approaches for solving large and complex problems, through the use of problem decomposition. There is plenty of evidence of the success of this sort of approach in large scale global optimization [13], [14].

A. Cooperative Coevolution

In nature, coevolution is the process of reciprocal genetic change in one species, or group, in response to another. The original framework of cooperative coevolution (CC) utilized within evolutionary algorithms was originally introduced by Potter and De Jong [15]. This framework uses a divide-and-conquer approach to split the decision variables into subpopulations of smaller size, so that each of these subpopulations is optimized with a separate EA. The original CC framework for high-dimensional optimization can be summarized as follows:

1) Decompose an objective vector into m low dimensional subcomponents.
2) Set $j = 1$ to start a new cycle.
3) Optimize the j-th subcomponent with a certain EA for a predefined number of fitness evaluations (FEs).
4) If $j < m$ then $j++$, and go to Step 3.
5) Stop if the stopping criteria are satisfied; otherwise go to Step 2 for the next cycle.

Here, a cycle consists of one complete evolution of all subcomponents and the main idea is to decompose a high-dimensional problem into several low-dimensional subcomponents and evolve these subcomponents cooperatively for a predefined number of cycles. CC has shown to be a good framework for solving large scale problems [13], [14]. Since the cooperative coevolutionary framework can be extended in a relatively easy way to multi-objective optimization, a number of approaches have been proposed which incorporate it to improve the performance of multi-objective EAs. This is evidenced by MOCCGA [16], which integrates cooperative coevolution with Fonseca and Fleming's MOGA [17]. MOCCGA uses a dominance rank for individuals, in which a count of the number of individuals dominating others is the fitness criterion. In MOCCGA, the objectives are evaluated twice for each individual both with the best ranked individuals from each subpopulation, as well as with randomly selected individuals. This follows the approach described by Potter and De Jong, which aims to decrease the premature convergence observed on some test problems adopted with the original CC framework. In MOCCGA, the subcomponents are ranked only within the same subpopulation. It is important to mention

that the number of evaluations adopted by the authors is not reported anywhere in the paper.

Another approach is presented in [9], where a cooperative coevolutionary algorithm for multi-objective optimization is presented. This algorithm subdivides the decision variable space and determines which portions of the decision variables intervals are being used and discards portions of the intervals that it deems that are not being used by the search process. It also subdivides intervals so that separate sub-populations can operate on the portions of these intervals which contribute to the search. Sub-populations which are not making contributions are eliminated from the search. There exist more examples of the use of CC as a framework, but none of them focuses on the solution of MOPs with a high number of decision variables, which is the main motivation in this work.

IV. Our Proposed Approach

Our proposed approach is based on the previous work done in large scale global optimization. The main idea of our proposed approach is to make use of the divide-and-conquer technique applied in the CC framework (such as in [15]), but transfering this concept to multi-objective optimization. We also adopt some additional concepts taken from other CC based frameworks found in the specialized literature [7]. Our aim was to combine the best of this previous work, and enhance it with our own ideas that are focused on the specific features found in evolutionary multi-objective optimization.

In the large scale global optimization literature, it has been reported that approaches based on the use of CC as a framework, have a poor performance in nonseparable functions, because the vector of decision variables in this kind of problems is composed by elements that interact with each other and are not independent. Because of this problem, it has been found that dividing the problem into random groups provides better results than applying a deterministic division scheme, when dealing with nonseparable functions [13], [14]. Motivated by this previous work, our proposed approach divides the vector of decision variables into S subpopulations, each one representing a subset of all the decision variables at a time rather than taking only one variable at a time. Our scheme assigns each decision variable to its corresponding group in a random way, since this will increase the chance of optimizing some interacting variables together. Our proposed approach is described next.

A. Description of our proposed approach

Our proposed approach works as follows: at the beginning, it divides the vector of decision variables $\vec{x}$ of dimension $D \in \mathbb{N}$ into $S \in \mathbb{N}$ subcomponents of equal size. Each subcomponent is created from a random grouping of decision variables in order to increase the probability of grouping interacting variables in non-separable problems. At the same time, S subpoputaions (species) are created, each one with NP individuals, and these S subpopulations are assigned their corresponding decision variables in a random way. This means that to each subpopulations, it corresponds a subcomponent from the S which have been already done. Thus, every subpopulation will have a total of m decision variables. This is graphically depicted in Figure 1.

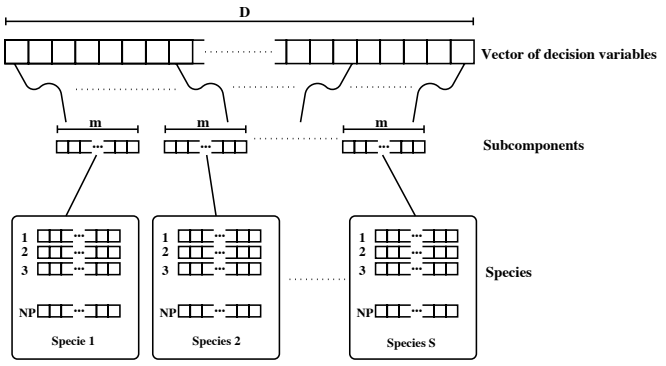

Fig. 1. Graphical representation of the subcomponents (species) creation. Here, we assume a vector of decision variables of dimension D which is divided into S subcomponents of dimension m, created in a random way from the original vector of decision variables and assigned to the S existing species, where $D = m * S$.

Once the subpopulations are created, the algorithm does a random initialization of all the individuals across all subpopulations. Then, the algorithm performs the *cycles* in which the evolution of each of the subpopulations is done for a given number of *generations*. This will continue until the stop condition is reached, and at the end, the solutions that are globally nondominated (i.e., with respect to all the subpopulations), constitute the outcome of the algorithm. The collaboration among the subpopulations takes place in the next way: in the first generation, random collaborations are formed and evaluated, obtaining a random individual from each subpopulation and forming a complete set of solutions to be evaluated in their objective functions. Then, the results from the evaluation are assigned back to the individual under evaluation. After the first generation, the resulting child subpopulations Q_1 to Q_S will be evaluated by forming collaborations with randomly selected components from the *best* non-dominated levels in the subpopulations, P_1 to P_S, of the previous generation. This is shown in Figure 2. The algorithm iterates until some termination condition is fulfilled (usually when a certain predefined number of cycles is reached). At the end, we apply a fast non-dominated sorting procedure as in the NSGA-II [18] to the best non-dominated levels of each subpopulations in order to obtain a final set of solutions for the problem being solved. A summary of the way in which our approach works is presented in Figure 3 and in Algorithm 1.

B. GDE3

Since our proposed approach adopts the GDE3 algorithm as its basic multi-objective optimizer, we provide next a brief description of this technique.

GDE3 [19] is the third version of the so-called Generalized Differential Evolution (GDE) algorithm [20], which is able to deal with multiple objectives. It starts with a population of

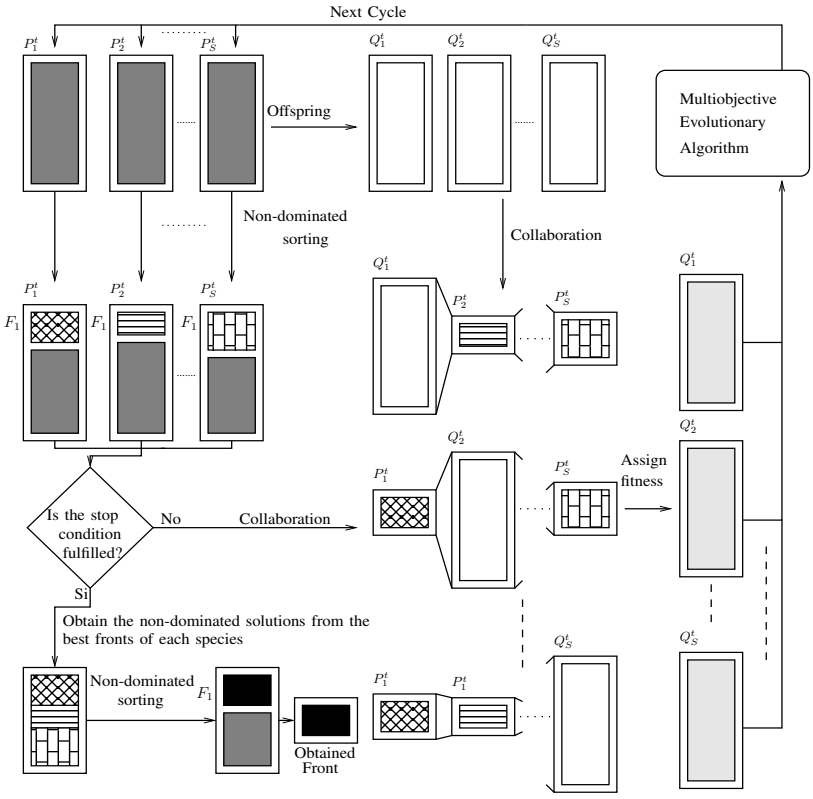

Fig. 3. General scheme of the proposed technique. Q_i^t are the offspring of each species and P_i^t are the parents of those descendants in the t-th cycle. There are S subpopulations, where S is the number of species, inside of which their corresponding decision variables have been assigned, with the method described in Figure 1.

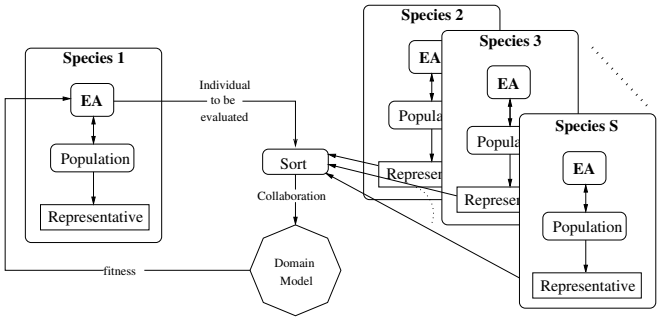

Fig. 2. Cooperative coevolutionary collaboration architecture from the perspective of species number 1. Here, we assume that we have S species, where the representative of each species for the collaboration is taken randomly from its best level of non-dominated solutions.

Algorithm 1 Cooperative Coevolutionary Framework

Input: NP, $Cycles$, $Gmax$, $NumEsp$
Output: $SolutionSet$

```
Pobs ← Populations(NP, NumEsp)
InitializeSpecies(Pobs)
for j ← 1 to Cycles do
  for i ← 1 to NumEsp do
    for k ← 1 to Gmax do
      MOEA(Pobs[i])
    end for
  end for
end for
SolutionSet ← ObtainNonDominatedSet(Pobs)
return SolutionSet
```

random solutions, which becomes the current population. At each generation, an offspring population is created using the differential evolution operators; then, the current population for the next generation is updated using the solutions of both, the offspring and the parent populations. Before proceeding to the next generation, the size of the population is reduced using non-dominated sorting and a pruning technique aimed at diversity preservation, in a similar way as NSGA-II, although the pruning used in GDE3 modifies the crowding distance of NSGA-II in order to solve some of its drawbacks when dealing with problems having more than two objectives.

C. Experimental Studies

For the purposes of this study, we adopted a benchmark that is scalable in the number of decision variables: Zitzler-Deb-Thiele test suite [2]. We selected four problems from this suite: ZDT1, ZDT2, ZDT3 and ZDT6. The main characteristics of these problems are described next. ZDT1 has a convex Pareto front which is continuous and uniformly distributed. ZDT2 has a non-convex Pareto front. ZDT3 has 5 discontinuous

non-convex fronts. ZDT6 has a non-uniform mapping between objective function space and decision variable space and has a non-connected Pareto front.

We compare our proposed approach with respect to two MOEAs: GDE3 [19] and NSGA-II [18]. In our experiments, we use a large number of decision variables that ranges from 200 up to 5000. Our approach will be using GDE3 as its basic multi-objective optimizer, so we decided to call our proposed approach: Cooperative Coevolutionary GDE3 (CCGDE3).

D. Methodology

Since the main objective of this work is to evaluate the behavior of our approach when solving MOPs with a large number of decision variables, we will analyze its convergence rate with respect to that of the other two MOEAs with respect to which it is compared. For this sake, we adopt the hypervolume performance indicator [21].

The hypervolume is obtained by computing the volume (in objective function space) of the non-dominated set of solutions Q that minimize a MOP. For every solution $i \in Q$, a hypercube v_i is generated with a reference point W and the solution i as its diagonal corner of the hypercube. The reference point W can be generated by building a vector of worst possible objective function values. Then, the hypervolume (HV) is computed as a union of all the found hypercubes as follows:

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{8}$$

Since the true Pareto fronts of the ZDT problems are known, we will run each of the MOEAs being analyzed, until they obtain an approximation of the Pareto front that has a hypervolume of 95% with respect to the true Pareto front. This represents a reasonable approximation to the true Pareto fronts in terms of convergence and spread of solutions. The aim of this study is to identify which of the MOEAs being compared is able to reach faster the true Pareto front. Since it is possible that some of the MOEAs being compared never achieve the desired convergence, we decided to use as an alternative stopping condition a maximum number of function evaluations. We set the maximum number of evaluations to ten million. In our experiments, we check the stopping condition at every 100 evaluations for GDE3 and NSGA-II, which means that the condition is checked at each iteration, and every cycle in CCGDE3, when we measure the hypervolume of the non-dominated solutions found so far. We performed 25 independent runs for each algorithm and each problem instance using a number of decision variables that ranges from 200 up to 5000. Since we are dealing with stochastic algorithms, we need to perform a statistical analysis of the obtained results to compare them with a certain level of (statistical) confidence. For assuring this we have performed a bootstrapping test [22]. We have done the calculations based on 1000 bootstrap replicates, from our original sets of 25 independent runs for each algorithm and for each problem instance. This was done in order to get the mean and standard error from the bootstrap distribution and to calculate confidence intervals, through the adjusted bootstrap percentile (BCa) interval, with a confidence level of 95% for the mean.

TABLE V
RUNNING TIME FOR ZDT1

D.V.	NSGA II	GDE3	CCGDE3
200	0.1412 mins	0.0488 mins	**0.0230 mins**
500	0.5174 mins	0.2865 mins	**0.0707 mins**
1000	*14.8514 mins*	2.9855 mins	**0.2060 mins**
2000	*89.0180 mins*	*11.8222 mins*	**0.8269 mins**
3000	*92.7289 mins*	*23.7238 mins*	**1.9190 mins**
4000	*159.6966 mins*	*59.8936 mins*	**2.7693 mins**
5000	*174.4521 mins*	*78.7402 mins*	**7.1859 mins**

E. Parameterization

The parameters of each MOEA used in our study were chosen in such a way that we could do a fair comparison among them. Thus, for NSGA-II and GDE3, we used an internal population size equal to 100. For CCGDE3, we used a populations size of 40 individuals for each subpopulation (species), since the number of species is set to two. In the case of NSGA-II, the distribution indexes for the SBX and polynomial-based mutation operators [18], were set as: $\eta_c = 20$ and $\eta_m = 20$, respectively. The crossover probability is $p_c = 0.7$ and the mutation probability is $p_m = 1/L$, where L is the number of decision variables. In GDE3 and CCGDE3, the value for both F and CR [19] was set to 0.5. In the particular case of CCGDE3, we used 2 species each of which had a size of 40 individuals for their internal populations. We used just one generation for each species per cycle. Finally, the test problems were used with the following numbers of decision variables: 200, 500, 1000, 2000, 3000, 4000 and 5000.

F. Analysis of results

In our experiments, we obtained the mean number of evaluations needed by GDE3, NSGA-II and CCGDE3 in order to generate an approximation of the Parero front with a hypervolume value of 95% of the true Pareto front. We also report the median of the time taken by each algorithm to solve each problem, averaged over the 25 independents runs performed. Tables I, II, III and IV show the mean, standard error and confidence intervals of the number of evaluations obtained from the bootstrapping test and Tables V, VI, VII and VIII show the average of the time, in minutes, needed to get an approximation with a hypervolume of 95% of the true Pareto front. We show in such tables, the results for ZDT1, ZDT2, ZDT3 and ZDT6, respectively. When a value of 10,000,000 appears in the table for any of the algorithms, this means that it was not able to obtain an acceptable approximation for the problem in the 25 independent runs performed. In this case, the average time reported appears in *italics*, because, in these cases, more time would be required in order to obtain the desirable convergence.

TABLE I
EVALUATIONS FOR ZDT1

	NSGA II			GDE3			CCGDE3		
	Mean	SE	CI	Mean	SE	CI	Mean	SE	CI
200	119556.66	1189.41	(117276,121879)	54055.30	258.13	(53525,54544)	24323.79	210.59	(23895,24710)
500	253903.03	2190.67	(249937,258559)	239592.64	1225.13	(237008,241923)	64723.36	303.95	(64059,65283)
1000	10000000	0	_	2212089.16	120619.60	(2026548,2517188)	155135.41	1197.78	(152405,157114)
2000	10000000	0	_	10000000	0	_	387793.43	3733.05	(380450,395144)
3000	10000000	0	_	10000000	0	_	627913.55	5982.13	(615268,638379)
4000	10000000	0	_	10000000	0	_	862118.13	6104.23	(851877,876888)
5000	10000000	0	_	10000000	0	_	1183792.22	12822.72	(1160979,1210149)

TABLE II
EVALUATIONS FOR ZDT2

	NSGA II			GDE3			CCGDE3		
	Mean	SE	CI	Mean	SE	CI	Mean	SE	CI
200	139511.64	926.74	(137609,141263)	74300.78	383.90	(73589,75152)	31766.35	236.42	(31331,32251)
500	287201.97	1439.25	(284480,289913)	506293.02	11795.19	(486781,533661)	87862.07	620.00	(86613,88959)
1000	10000000	0	_	10000000	0	_	221829.77	6897.47	(214008,254278)
2000	10000000	0	_	10000000	0	_	508944.26	5491.12	(498824,520553)
3000	10000000	0	_	10000000	0	_	797833.51	14679.82	(772387,830117)
4000	10000000	0	_	10000000	0	_	1088563.11	21295.44	(1052706,1140226)
5000	10000000	0	_	10000000	0	_	1544704.96	56609.77	(1450560,1685531)

TABLE III
EVALUATIONS FOR ZDT3

	NSGA II			GDE3			CCGDE3		
	Mean	SE	CI	Mean	SE	CI	Mean	SE	CI
200	119416.07	1240.83	(116495,121591)	69935.36	228.51	(69448,70378)	24525.18	188.76	(24112,24886)
500	245161.60	1639.91	(242088,248536)	316581.42	1484.31	(313621,319528)	63566.92	463.12	(62674,64525)
1000	10000000	0	_	1408341.00	8834.41	(1390794,1425559)	145131.07	900.46	(143306,146744)
2000	10000000	0	_	5492656.00	33479.38	(5419861,5552752)	345831.12	2234.16	(341108,349741)
3000	10000000	0	_	9632953.07	54638.04	(9495504,9718228)	562077.37	4226.78	(553467,570352)
4000	10000000	0	_	10000000	0	_	777169.50	6922.90	(764062,791999)
5000	10000000	0	_	10000000	0	_	1006945.83	7946.20	(995261,1027586)

TABLE IV
EVALUATIONS FOR ZDT6

	NSGA II			GDE3			CCGDE3		
	Mean	SE	CI	Mean	SE	CI	Mean	SE	CI
200	584371.092	1666.53	(581109,587773)	461942.38	2534.61	(457304,467485)	157878.21	783.76	(156356,159437)
500	1150362.232	2223.03	(1146032,1154919)	10000000	0	_	481930.17	3064.22	(475202,487641)
1000	10000000	0	_	10000000	0	_	1283136.32	7555.97	(1266427,1296133)
2000	10000000	0	_	10000000	0	_	2810482.71	16512.01	(2778592,2843568)
3000	10000000	0	_	10000000	0	_	3892644.27	35764.36	(3805150,3947065)
4000	10000000	0	_	10000000	0	_	4859792.40	45586.92	(4785931,4964634)
5000	10000000	0	_	10000000	0	_	5768502.88	19085.49	(5723900,5800079)

Now, we will pay attention to the convergence rate, i.e., the number of function evaluations needed by the algorithms to find an approximation of the true Pareto front according to our success condition. In Figures 4, 5, 6 and 7, we plot the results of the median of the number of evaluations needed by GDE3, NSGA-II and CCGDE3 to obtain an approximation of the true Pareto front that has a hypervolume value of 95%. These plots are shown for ZDT1, ZDT2, ZDT3 and ZDT6. We have connected with a line the results of the algorithms for each number of decision variables under consideration. Thus, we can observe that CCGDE3 is the fastest algorithm, which means that it scales better than the other two MOEAs

TABLE VI
RUNNING TIME FOR ZDT2

D.V.	NSGA II	GDE3	CCGDE3
200	0.1964 mins	0.6402 mins	**0.0281 mins**
500	0.6001 mins	0.7082 mins	**0.0997 mins**
1000	*6.6401 mins*	*1.0876 mins*	**0.2968 mins**
2000	*14.7021 mins*	*17.4891 mins*	**1.2929 mins**
3000	*89.6921 mins*	*40.4873 mins*	**2.3151 mins**
4000	*159.5900 mins*	*59.7938 mins*	**3.5489 mins**
5000	*175.0103 mins*	*78.4983 mins*	**7.8487 mins**

TABLE VII
RUNNING TIME FOR ZDT3

D.V.	NSGA II	GDE3	CCGDE3
200	0.1495 mins	0.3203 mins	**0.0241 mins**
500	0.4471 mins	0.3448 mins	**0.0654 mins**
1000	*10.3974 mins*	2.0606 mins	**0.2886 mins**
2000	*15.7690 mins*	12.3773 mins	**0.8323 mins**
3000	*77.6378 mins*	29.1153 mins	**1.3393 mins**
4000	*159.5612 mins*	*59.2153 mins*	**2.2788 mins**
5000	*174.8563 mins*	*78.1015 mins*	**6.2366 mins**

TABLE VIII
RUNNING TIME FOR ZDT6

D.V.	NSGA II	GDE3	CCGDE3
200	0.7987 mins	14.0197 mins	**0.1507 mins**
500	2.2383 mins	*15.4108 mins*	**0.5872 mins**
1000	*7.1561 mins*	*17.2080 mins*	**1.9982 mins**
2000	*42.4543 mins*	*33.3090 mins*	**7.0168 mins**
3000	*102.4043 mins*	*34.0594 mins*	**9.6578 mins**
4000	*159.2021 mins*	*58.7624 mins*	**31.1226 mins**
5000	*174.6141 mins*	*77.6254 mins*	**39.1256 mins**

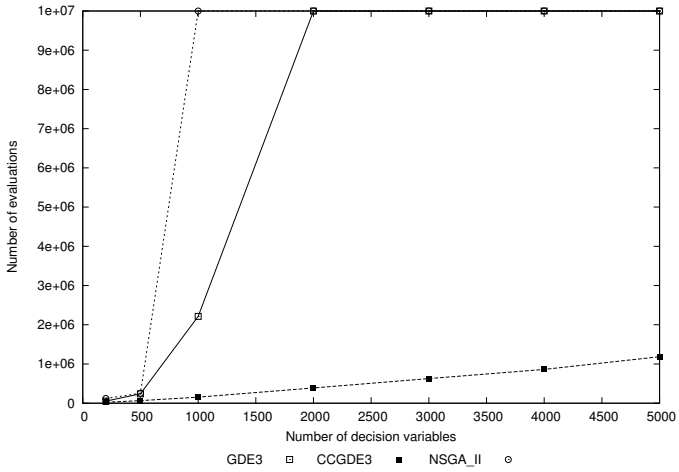

Fig. 4. Plot of the number of evaluations needed by each algorithm to obtain an approximation with 95% of the hypervolume of the true Pareto front, for ZDT1.

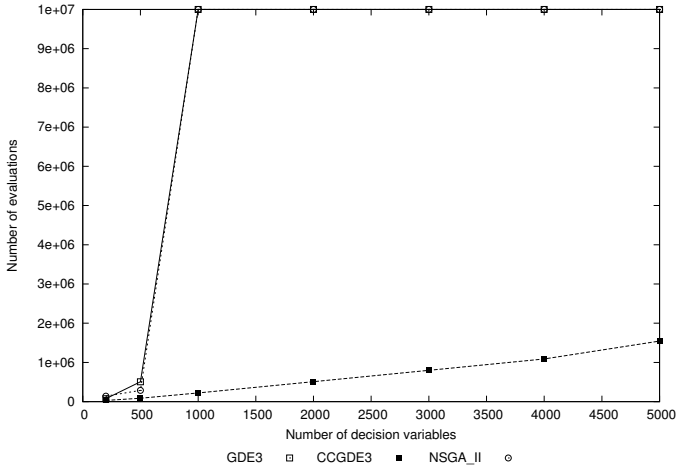

Fig. 5. Plot of the number of evaluations needed by each algorithm to obtain an approximation with 95% of the hypervolume of the true Pareto front, for ZDT2.

when the number of variables is large. The lines clearly show that CCGDE3 tends to be faster than the other techniques as the number of decision variables increases, while NSGA-II and GDE3 experiment the opposite behavior. It is clear that CCGDE3 is much faster than NSGA-II and GDE3, not only on terms of number of evaluations, but also in terms of time, since the time needed for CCGDE3 to get a front with a hypervolume of 95%, for all the test problems adopted, is much less than the one needed by NSGA-II and GDE3, as shown by the results reported in Tables V, VI, VII and VIII. Also, Tables I, II, III and IV, show narrowed CI values, relative to the values of the means. Therefore, the results are a very reliable estimation of the average behavior of the algorithms.

V. CONCLUSIONS AND FUTURE WORK

This paper proposes a new cooperative coevolutionary framework for solving MOPs with a large number of decision variables. Combined with GDE3, we presented a novel cooperative coevolutionary MOEA, called CCGDE3, which was shown to be able to successfully deal with a large number of decision variables (up to 5,000). Studies were carried out to evaluate the performance of CCGDE3 on the ZDT benchmark functions. We have estudied the convergence rate of our proposed CCGDE3 with respect to that of NSGA-II and GDE3 when solving these problems. The results confirmed that our proposed approach is very effective and efficient in tackling large scale MOPs. Although we used GDE3 as the basic subcomponent optimizer, it should be relatively easy to incorporate any other MOEA into our framework. Moreover, more than one kind of MOEA could also be employed as a subcomponent optimizer, in order to combine different search biases. Thus, as part of our future work, we aim to focus on more techniques that can be incorporated into our framework and we also intend to find a way of simplifying the parameters setting procedure required by our algorithm by adapting (online or in a self-adaptive way) the number of species needed as well as the size of each subpopulation.

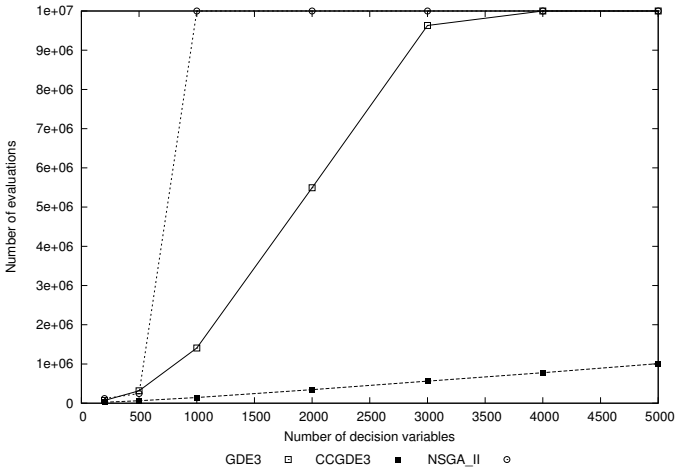

Fig. 6. Plot of the number of evaluations needed by each algorithm to obtain an approximation with 95% of the hypervolume of the true Pareto front, for ZDT3.

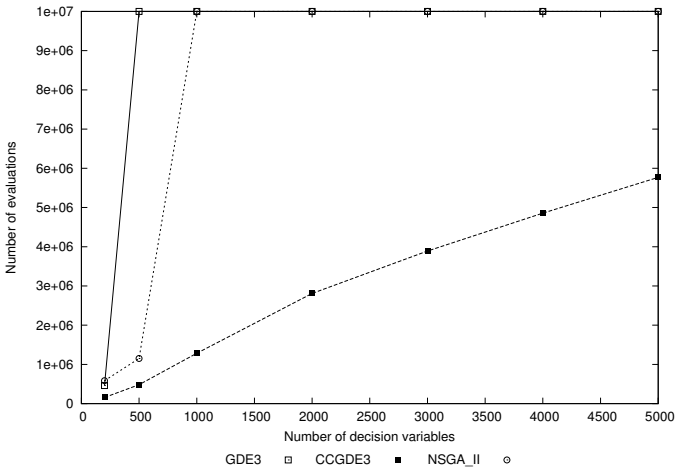

Fig. 7. Plot of the number of evaluations needed by each algorithm to obtain an approximation with 95% of the hypervolume of the true Pareto front, for ZDT6.

ACKNOWLEDGMENTS

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at Computer Science Department of CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT project no. 103570.

REFERENCES

[1] W. Cook, *Mathematical Programming Computation*, mathematical optimization society ed. Springer, 2009.

[2] E. Zitzler, K. Deb, and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, Tech. Rep. 70, December 1999.

[3] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Tech. Rep., 2001.

[4] S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, October 2006.

[5] J. Durillo, A. Nebro, C. Coello Coello, J. Garcia-Nieto, F. Luna, and E. Alba, "A Study of Multiobjective Metaheuristics When Solving Parameter Scalable Problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 618–635, August 2010.

[6] J. J. Durillo, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, "A Comparative Study of the Effect of Parameter Scalability in Multi-Objective Metaheuristics," in *2008 Congress on Evolutionary Computation (CEC'2008)*. Hong Kong: IEEE Service Center, June 2008, pp. 1893–1900.

[7] Y. R. Garca, "Optimizacin evolutiva a gran escala con y sin restricciones," Master's thesis, Centro de Investigacin y de Estudios Avanzados del Instituto Politcnico Nacional, dic 2011.

[8] K. Tan, Y. Yang, and C. Goh, "A Distributed Cooperative Coevolutionary Algorithm for Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 527–549, October 2006.

[9] C. A. Coello Coello and M. Reyes Sierra, "A Coevolutionary Multi-Objective Evolutionary Algorithm," in *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, vol. 1. Canberra, Australia: IEEE Press, December 2003, pp. 482–489.

[10] J. Bader and E. Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, Spring, 2011.

[11] A. B. de Carvalho and A. Pozo, "Measuring the convergence and diversity of CDAS Multi-Objective Particle Swarm Optimization Algorithms: A study of many-objective problems," *Neurocomputing*, vol. 75, no. 1, pp. 43–51, January 1 2012.

[12] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, December 2007.

[13] X. Y. Zhenyu Yang, Ke Tang, "Differential evolution for high-dimensional function optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, vol. 1, sept. 2007.

[14] X. Y. Mohammad Nabi, Zhenyu Yang, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, vol. 1, sept. 2010, pp. 1– 8 Vol.1.

[15] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, ser. PPSN III. London, UK, UK: Springer-Verlag, 1994, pp. 249–257.

[16] N. Keerativuttiumrong, N. Chaiyaratana, and V. Varavithya, "Multi-objective Co-operative Co-evolutionary Genetic Algorithm," in *Parallel Problem Solving from Nature—PPSN VII*, J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. F.-V. nas, and H.-P. Schwefel, Eds. Granada, Spain: Springer-Verlag. Lecture Notes in Computer Science No. 2439, September 2002, pp. 288–297.

[17] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest, Ed., University of Illinois at Urbana-Champaign. San Mateo, California: Morgan Kauffman Publishers, 1993, pp. 416–423.

[18] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," Indian Institute of Technology, Kanpur, India, KanGAL report 200001, 2000.

[19] S. Kukkonen and J. Lampinen, "GDE3: The third Evolution Step of Generalized Differential Evolution," in *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, vol. 1. Edinburgh, Scotland: IEEE Service Center, September 2005, pp. 443–450.

[20] J. Lampinen, "DE's selection rule for multiobjective optimization," Lappeenranta University of Technology, Department of Information Technology, Tech. Rep., 2001.

[21] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, November 1999.

[22] G. P. Mccabe and D. S. Moore, *Introduction to the Practice of Statistics*. W.H. Freeman & Company, 2009.