ISSN 2007-9737

A Hybrid Approach for Solving Dynamic Bi-level Optimization Problems

Eduardo Samaniego[1], Pavel Novoa-Hernández[1,2]

[1] Universidad Técnica Estatal de Quevedo, Los Ríos, Ecuador

[2] Guest Lecturer at Universidad Estatal de Milagro, Guayas, Ecuador

{esamaniego, pnovoa}@uteq.edu.ec

Abstract. Several real-life decision scenarios are hierarchical, which are commonly modeled as bi-level optimization problems (BOPs). As other decision scenarios, these problems can be dynamic, that is, some elements of their mathematical model can change over time. This kind of uncertainty imposes an extra level of complexity on the model, since the algorithm needs to find the best bi-level solution over time. Despite the importance of studying these problems, the literature reflects just a few works on dynamic bi-level optimization problems (DBOPs). In this context, this work addresses the solution of DBOPs from the viewpoint of metaheuristic methods. Our hypothesis is that, by hybridizing successful solving approaches from both bi-level and dynamic optimization fields, an effective method for DBOPs can be obtained. In this regard, we propose a hybrid method that combines a coevolutionary approach and a self-adaptive, multipopulation algorithm. Experimental results assert our hypothesis, specially for certain information exchange mechanisms.

Keywords. Dynamic Bi-level Optimization, Coevolutionary algorithms, Differential Evolution, Self-adaptation, Hybrid metaheuristics

1 Introduction

A bilevel optimization problem has two levels of single or multi-objective optimization problems such that the optimal solution of the lower level problem determines the feasible space of the upper level optimization problem. In general, the lower level problem is associated with a variable vector x_l and a fixed vector x_u.

However, the upper level problem usually involves all variables $X = (x_u, x_l)$ [26].

From the economic viewpoint they can be seen as decision making scenarios where an upper-level *leader* is optimizing a strategic (main) model, while a lower-level *follower* reacts to the *leader* decisions by optimizing a related subproblem. In other words, they are "*mathematical programs with optimization problems in the constraints*" [7]. BOPs have been extensively studied in the past (e.g. location routing problems [19], relief operations after a disaster [5], Stackelberg games [27], engineering problems [14], among others). Even in more simple cases (e.g. models with linear objective functions and constraints) BOPs are hard to solve by traditional optimization techniques [11, 12].

Nowadays, the use of metaheuristic methods to deal with bilevel optimization problems is gaining increasing attention [30]. One of the reasons behind this interest is their ability to obtain near optimal solutions in a reasonable amount of time [3]. Moreover, as these methods are derivative-free, they could be applied to solve non-differentiable problems.

As in other optimization scenarios, uncertainty may exist, being one source of it, the dynamic nature of data involved in the mathematical model. Examples of these dynamic BOPs (DBOPs), could be location routing problems with variable number of depots or clients over time, or aid distribution in recurrent disasters, among others. This feature increases the complexity of BOPs, since now the

goal of any solving strategy is to find the optimal bi-level solution (or the best possible achievable solution with the available resources), at every time step. Despite the importance of studying these special scenarios, the literature reflects just a few works addressing this topic.

For example, in [29] a genetic algorithm is used to solve a dynamic traffic signal problem. The authors were focused on modeling the decision scenario of dynamic traffic signal optimization in networks with time-dependent demand and stochastic route choice. Here, the upper-level problem represented the decision-making behavior (signal control), of the system manager, while the user travel behavior is represented at the lower level.

[18] proposed a general model for dynamic bi-level multi-objective problems. The authors analyzed the interaction between the upper and lower levels over time and described the benefits of bi-level dynamic multiobjective optimization through the examination of an industrial case in which the design of a paper mill (upper level) and the mill operation (lower level), are optimized. As solution method, the authors considered a differential evolution algorithm.

Another study on dynamic bi-level optimization was conducted by [6]. Authors addressed the modelization and solution of a multi-period portfolio selection problem in stochastic markets with bankruptcy risk control. They assumed that the investor wants to find an investment strategy to maximize his terminal wealth, while the bankruptcy risk in each period needs to be controlled. Essentially, a bi-level programming algorithm is employed for deriving analytical solutions for the each period-wise optimization problem.

In this context, we propose to deal with these complex scenarios by exploiting the current advances on metaheuristics methods from the fields of bi-level [30] and evolutionary dynamic optimization [1]. To the best of our knowledge, there are no research works related to this topic.

More specifically, in this paper we address the solution of dynamic bi-level optimization problems by hybrid metaheuristics. Our hypothesis is that, by hybridizing successful solving approaches from both bi-level and dynamic optimization fields, an effective method for solving DBOPs can be obtained. We will focus on coevolutionary and multipopulation methods, which are successful strategies for tackling bi-level and dynamic problems, respectively [30, 16].

The rest of the paper is organized as follows: Section 2, gives the necessary background on DBOPs. Section 3 describes the proposed method for solving DOPs, which is validated through computational experiments in Section 4. Finally, some concluding remarks and future works are outlined in Section 5.

2 Background and Related Works

This section is devoted to the fundamentals of dynamic bi-level optimization problems. In order to better understand the formulation of DBOPs, we start by defining bi-level and dynamic optimization problems. Furthermore, we summarize some available metaheuristic based solution approaches. The section ends with the definition of the dynamic bi-level optimization problems.

2.1 Bi-level Optimization Problems

A bi-level optimization problem is defined as follows:

$$BOP := \begin{cases} \max\limits_{x \in \Omega, y \in \Gamma} F(x, y), \\ \text{subject to} \\ \qquad \max\limits_{y \in \Gamma} f(x, y), \end{cases} \tag{1}$$

where $\Omega \subseteq \mathbb{R}^n$ and $\Gamma \subseteq \mathbb{R}^m$ are the feasible search spaces for x (upper-level) and y (lower-level) decision variables, respectively. Besides, $F, f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ are the objective functions for the upper-level and lower-level sub-problems, respectively. One important feature of BOPs is the relationship between the upper and the lower models.

For example, for a given value of x in the upper model it could be possible to obtain a different optimization problem at the lower level, which in turn must be solved for y. As a consequence, it may happen that in the upper level model the y

part of the optimal solution is not the same as the one obtained at the lower level. So, independently solving both models does not necessarily lead to the true optimal solution for the bi-level problem. In other words, the optimal reactions of the lower decision maker could not be the same as the one expected for the upper counterpart.

We will now illustrate a real-life BOP model through an example. We consider the Stackelberg competition model described by [26] in the context of game theory.

Example 1 (Stackelberg competition) Two firms (l and f) compete in order to maximize their profits according to the following model:

$$ExBOP := \begin{cases} \max\limits_{q_l, q_f, Q \in \mathbb{R}^+} \prod_l = P(Q)q_l - C(q_l), \\ \text{subject to} \\ \qquad \max\limits_{q_f \in \mathbb{R}^+} \prod_f = P(Q)q_f - C(q_f), \\ \qquad q_l + q_f \geq Q, \end{cases} \tag{2}$$

where q_l y q_f are the production levels and Q is the required quantity (demand). Functions P and C represent the price of the goods sold and the production cost of each firm, respectively. Besides, it is worth noting that the model has just one functional constraint ensuring that all demand is satisfied.

Solving this model implies for the *leader* firm l, finding the so-called Stackelberg equilibrium. In other words, the optimal solution corresponds to the best production level that firm l must to achieve, taking into account the optimal reaction of the *follower* firm f.

Now suppose that both firms sell homogeneous goods and their corresponding price functions P have a linear form as inverse of the demand Q:

$$P(Q) = \alpha - \beta Q, \tag{3}$$

where α and β are two positive constants. Furthermore, we assume that the cost functions for both firms are given by convex quadratic expressions:

$$C(q_l) = \delta_l q_l^2 + \gamma_l q_l + c_l, \tag{4}$$

$$C(q_f) = \delta_f q_f^2 + \gamma_f q_f + c_f, \tag{5}$$

where $\gamma_l, \gamma_f, \delta_l, \delta_f$ are positive constants and c_l, c_f are fixed costs.

[26] shows that it is possible to analytically compute the optimal solution, which is given by:

$$q_l* = \frac{2(\beta + \delta_f)(\alpha - \gamma_l) - \beta(\alpha - \gamma_f)}{4(\beta + \delta_f)(\beta + \delta_l) - 2\beta^2}, \tag{6}$$

$$q_f* = \frac{\alpha - \gamma_f}{2(\beta + \delta_f)} - \frac{\beta(\alpha - \gamma_l) - \frac{\beta^2(\alpha - \gamma_f)}{2(\beta + \delta_f)}}{4(\beta + \delta_f)(\beta + \delta_l) - 2\beta^2}, \tag{7}$$

$$Q^* = q_l^* + q_f^*. \tag{8}$$

We can interpret these values as the optimal strategies of the leader and follower at Stackelberg equilibrium.

From the viewpoint of metaheuristic methods, BOPs can be solved using the following approaches [30]: (1) nested sequential, (2) single-level transformation, (3) multi-objective, and (4) coevolutionary.

The first one is the most intuitive but the most computationally expensive, since for every single evaluation of the upper-level objective function, the lower-level problem needs to be solved. To cope with such complexity, other authors transform the bi-level problem into suitable models that can be solved by other metaheuristics (e.g. genetic algorithms, multi-objective evolutionary algorithms, etc.), which is the case in the second and third approaches.

However, note that these approaches can be applied under the following conditions: 1) an explicit model of the problem is known and 2) there exist some proof that such transformations lead to the true (or near), optimal solution of the problem.

Finally, the more general approach is to use coevolutionary algorithms (CoEAs). Here, the algorithm evolves two populations (sets) of solutions for both, the upper-level and the lower-level problems. In addition, CoEAs must implement some exchange mechanisms for guiding the search process. Defining such mechanisms is a key issue in the algorithm performance on BOPs. We will return to this topic in Sec. 3.

2.2 Dynamic Optimization Problems

A dynamic optimization problem (DOP), is formally defined as:

$$DOP := \left\{ \max_{x \in \Omega} F(x, \phi, t), \right. \tag{9}$$

where $\Omega \subseteq \mathbb{R}^n$ is the search space, $t \in \mathbb{N}_0$ is the real-world time, $\phi \in \Phi$ are the system control parameters and $F : \mathbb{R}^n \times \Phi \times \mathbb{N}_0 \to \mathbb{R}$, is the objective function. These system control parameters determine the solutions' distribution in the landscape, for instance, the objective function parameters, the search space dimension, etc. So, model's dynamism comes from a change in ϕ after a time period. Hence, the algorithm is faced with different environments during the run.

In this dynamic context, the main goal of a metaheuristic is to find the best solution at every time step. Between changes, the problem is "static" thus allowing the algorithm to perform the optimization process. So, keeping a suitable level of diversity in the population is a challenge when using population-based metaheuristics in dynamic environments. A proper management of diversity allows for avoiding premature convergence in previous environments and for tracking the new optima after the change.

Regarding of how this challenge has been addressed in the past, [13] and [8] observed four strategies: 1) enhancing diversity after the change, 2) maintaining diversity during run, 3) memory-based approaches and 4) multi-population approaches. More recently [21] pointed out that another alternative is to implement self-adaptive strategies [20] to cope with changes. This latter approach provides the algorithm, the ability to intelligently react to environment variations, as was shown by [22, 24].

2.3 Dynamic Bi-level Optimization Problems

From the previous definitions (1) and (9), it is straightforward to derive a general formulation for dynamic bi-level optimization problems:

$$DBOP := \begin{cases} \max\limits_{x \in \Omega, y \in \Gamma} F(x, y, \phi_u, t), \\ \text{subject to} \\ \qquad \max\limits_{y \in \Gamma} f(x, y, \phi_l, t), \end{cases} \tag{10}$$

Here, note that there are two sets of system control parameters, ϕ_u and ϕ_l, corresponding to the upper level and lower level models, respectively. It means that different dynamics could be present in both models, including the case in which one of models is not dynamic. In other words, according to the presence or not of dynamism at the upper and/or lower level objective functions, we have the four cases depicted in Fig 1. They are:

1. BOPs where both subproblems are stationary,
2. DBOPs with dynamism only in the *upper*-level problem,
3. DBOPs with dynamism only in the *lower*-level problem and
4. DBOPs with dynamism at *both* levels.

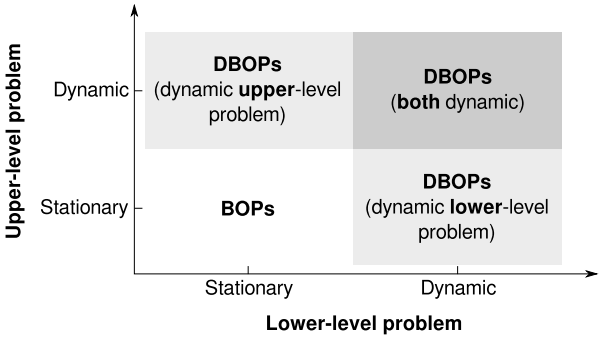

Fig. 1. Possible bi-level optimization problems according the type (stationary or dynamic), of the lower-level and upper-level problems

The most difficult scenario arises when both subproblems are dynamic.

In order to illustrate a DBOP, we derive the dynamic version of the problem described in *Example 1* from Sec. 2.1.

Example 2 (Dynamic Stackelberg competition) From model (2), suppose that firm l has time-varying reactions as a consequence of the market conditions related to the production costs. So, by considering that the positive constants of cost function C are dynamic we have the following model:

$$E_{DBOP} := \begin{cases} \max\limits_{q_l, q_f, Q \in \mathbb{R}^+} \prod_l = P(Q)q_l - C(q_l), \\ \text{subject to} \\ \quad \max\limits_{q_f \in \mathbb{R}^+} \prod_f = P(Q)q_f - C(q_f, \phi_C, t), \\ \quad q_l + q_f \geq Q, \end{cases} \tag{11}$$

where $\phi_C = (\delta_f, \gamma_f, c_f)^T$ are the system control parameters, which are subject to change. Suppose that they change according to the following transition rule:

$$\phi_C(t+1) = \phi_C(t) + r \cdot sev, \tag{12}$$

where $sev \geq 0$ is a vector composed by the change severity of each parameter and r represents a vector of random numbers drawn from the standard normal distribution.

One important question here is how the optimal solution of the model is affected when these system control parameters change over time. By using the general expressions (6 - 8), for the optimal solutions, we can have an idea of how it can be affected. In that sense, Figure 2, shows what happens in a hypothetical scenario, in which the model changes 10 times and the other parameters takes fixed values as follows: $\alpha = 82.51$, $\beta = 10.74$, $\delta_l = 9.33$, $\gamma_l = 3.76$ and $c_l = 5.96$. Besides, $sev = (1.0, 1.0, 0.2)^T$.

Figure 2, shows the effect of varying δ_f, γ_f and c_f (plot Fig. 2-a)) on the optimal solution (plot 2-b)) and on the corresponding objective function values for the optima solution 2-c)). According to our DBOPs classification this model corresponds to a DBOPs with dynamism just in the lower-level subproblem. However, as Fig. 2-c) shows, the upper-level objective function is also affected by the changes in the lower-level subproblem. This is because the interaction between variables of both levels, which is stated by the functional constraint.

The reader must be aware that real-life DBOPs can be more complex than this illustrative model. For instance, models in higher dimension, with stronger interactions among decision variables [26], could not be solved by analytical or numeric techniques. Besides, if the change function (e.g. Eq. 12) and the frequency of such changes are not known, then we must to rely on approximation optimization methods for finding near optimal solution quickly, that is, before the occurrence of a new environment in the near future. In what follows we explain our proposal to deal with such scenarios.

3 Proposed Approach

As mentioned before, coevolutionary algorithms (CoEAs), are among the most general approaches for tackling BOPs. In this context, a successful experience has been recently reported by [15]. Basically, a CoEA performed a pairwise optimization process (coevolution), by exploiting the separable structure of the problem at hand. Usually this is the case in bi-level optimization problems.

On the other hand, in the context of dynamic environments, using multi-population and self-adaptive approaches have shown to be very effective [9, 22, 24], specially, when combined with the differential evolution metaheuristic [28]. While the use of several populations enables a proper exploration of the search space, self-adaptation contributes to enhance the algorithm diversity and the optimum tracking over time.

From these facts it is reasonable to expect that a suitable method for solving DBOPs should involve a combination of the above approaches. In this sense, we propose a hybrid metaheuristic that comprises a simple coevolutionary approach based on the works of [25, 15] and the mSQDE algorithm from [22]. Specifically, mSQDE is a self-adaptive, multipopulation algorithm proposed for dynamic optimization. We have based our selection on the reported success of such approaches in their respective domains.

Figure 3, outlines the general structure of the proposed method, named *CoEvoMSQDE*. Note that CoEvoMSQDE acts as a coordinator by deciding *what*, *when* and *how* the coevolution process is carried out. Two mSQDE algorithms/instances, denotes as $mSQDE_u$ and $mSQDE_l$, are used for independently optimizing the upper-level and lower-level subproblems.

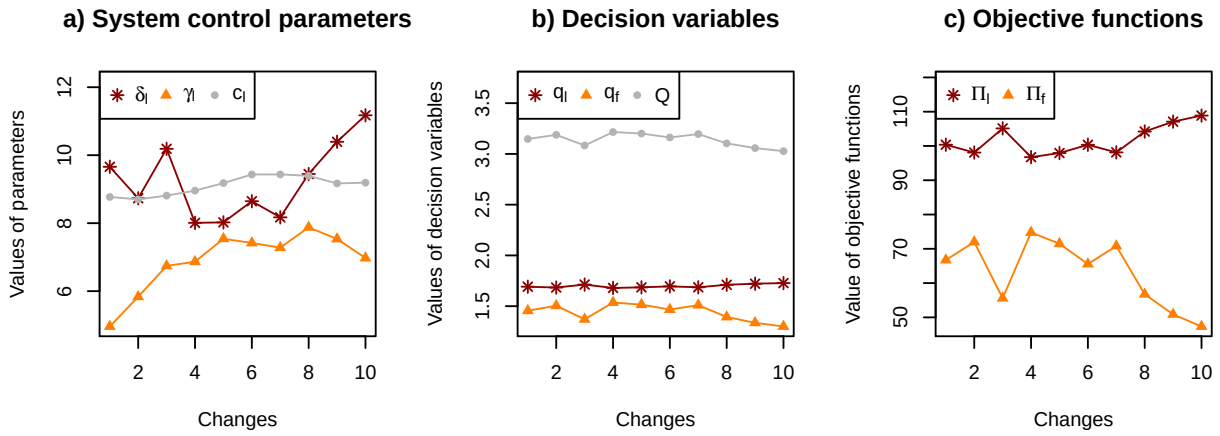

Fig. 2. Effects of changing system control parameters over time a), on decision variables b) and objective functions c), for the optimal solution

Each mSQDE is composed by a set of populations and every population comprises a set of solutions to the problem at hand.

As pointed out by [30], stating *what*, *when* and *how* is a key issue in designing CoEAs. So, here we explore different mechanisms. Specifically, regarding *what* information is exchanged, we consider three alternatives:

1. The global best solution of mSQDE instances (g),
2. The best solutions of sub-populations of mSQDE instances (G) and
3. All solutions of the best sub-population of mSQDE instances (P).

Regarding *how* the exchange process is carried out, we consider the following alternatives:

1. Exchange with preference in the upper-level algorithm. We refer to this scheme as *u*. The upper-level algorithm is updated with the current best solution of the lower-level algorithm. Then, all solutions of the upper-level algorithm are evaluated and its global best solution is updated. Finally, this new global best solution of the upper-level algorithm is sent to the lower-level algorithm.

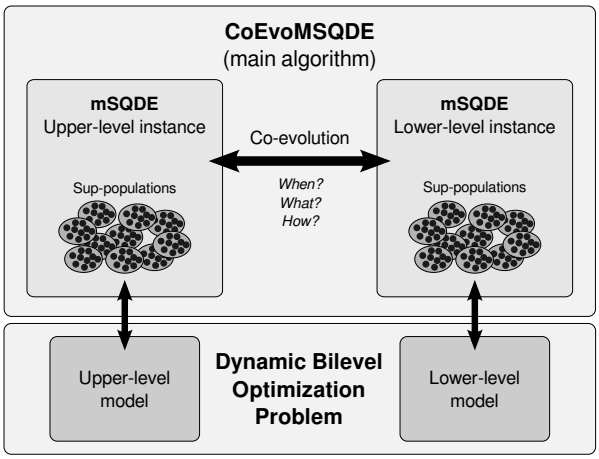

Fig. 3. Proposed hybrid approach for solving DBOPs

2. Exchange with preference in the lower-level algorithm. Referred as *l*. It operates as the previous scheme, but using the lower-level algorithm first.
3. Exchange without preference. Referred as *w*. The algorithms exchange their current best solutions, without intermediary evaluation and selection process.

With the aim of illustrating these alternatives, Fig. 4, shows them through task diagrams over

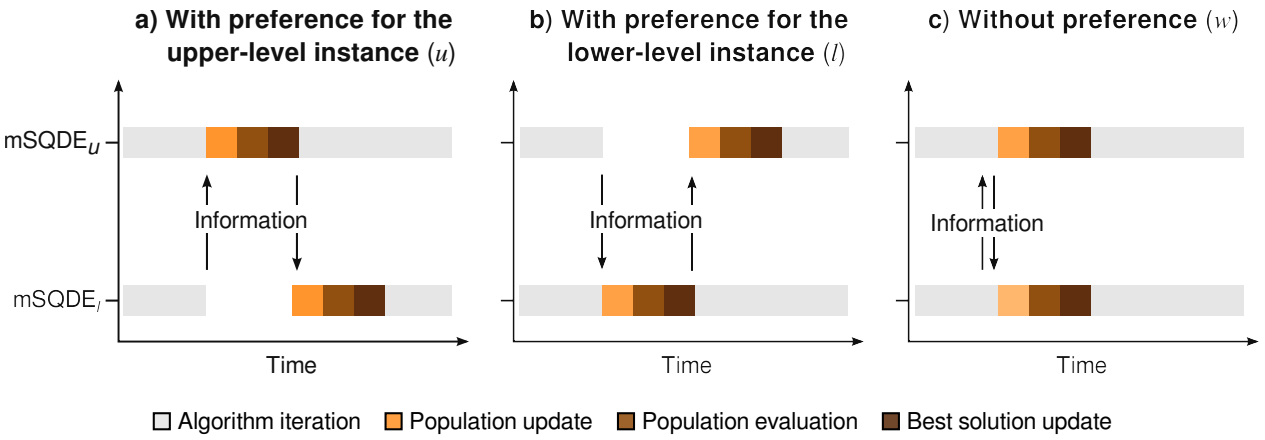

Fig. 4. Alternatives for the information exchange process in the CoEvoMSQDE algorithm

time. Please note that the operation involved in these schemes are marked by different tonalities.

Finally, regarding *when* the exchange process will be done, we simply do it after a predefined number of iterations.

The main steps of the *CoEvoMSQDE* method are depicted in Algorithm 1. Note that the first two steps are devoted to set the problem definitions to the instances $mSQDE_u$ and $mSQDE_l$, which are responsible for evolving two different populations related to the upper-level and lower-level problems, respectively. Further, the $mSQDE_u$ instance initialize its population, by generating random solutions in the search space, while the $mSQDE_l$ do the same by copying the upper-level solutions. It is important to note that we assumed that both instances have populations of the same size. At the main cycle (steps 5-11), the exchange condition is checked first. In the case that it is met, the exchange process is performed. Otherwise, $mSQDE_u$ and $mSQDE_l$ iterate.

Specifically, the iterative process of algorithms $mSQDE_u$ and $mSQDE_l$ include: a change detection mechanism based on the re-evaluation of the global best solution; an exclusion principle to prevent two subpopulations exploring the same region of the search space; the use of several subpopulations for efficiently exploring the search space; and a self-adaptive strategy to maintain a proper balance of diversity in subpopulations.

Regarding to the self-adaptive strategy of mSQDE it is important to state that it has been defined as class of self-adaptation applied to the mechanisms for DOPs [24]. More specifically, it includes self-adaptation into the *diversity during the run* mechanism. Such mechanism is based on the generation of the so-called *quantum* individuals proposed by [2]. In the original scheme, these random individuals are generated in a hypersphere with a predefined radius r_{cloud} during the run. So, the algorithm ability for tracking the optimum depends on setting r_{cloud} to a similar value of the shift severity. However, this latter information could be unknown in some real-world scenarios. For solving this issue, [22] proposed that each DE conventional individual (candidate solution), codify in its representation a realization of parameter r_{cloud}, which allows for generating a quantum individual.

Formally, be conventional individuals denoted as $y_i = \langle x_i, f_i, rc_i \rangle$, where x_i is the vector of decision variables, f_i is the corresponding objective value and rc_i is the realization of r_{cloud}. Then, rc_i is mutated as follows:

$$\tilde{rc}_i \leftarrow \begin{cases} rand_1 \cdot \lambda \cdot r_{excl} & \text{if } rand_2 < \tau, \\ rc_i & \text{otherwise,} \end{cases} \quad (13)$$

where $\tau, \lambda \in [0, 1]$ are the mutation rate and the scaling factor, respectively. The r_{excl} is the exclusion radius from the original approach of [2], which is aimed to limit the exploration area of

the subpopulations. Random numbers $rand_1$ and $rand_2$ are generated uniformly.

It can be observed that a new $\tilde{rc}_i$ is obtained with probability τ, from a variation of the product $\lambda \cdot r_{excl}$. So, the original rc_i is kept with probability $1 - \tau$.

The above features are depicted by Algorithm 2. For more details, the reader is referred to [22, 24].

```
   // Initialization
1  Set the upper-level problem definition to
   algorithm mSQDE_u;
2  Set the lower-level problem definition to
   algorithm mSQDE_l;
3  Randomly initialize population of mSQDE_u;
4  Copy the mSQDE_u population in the
   mSQDE_l;
   // Main cycle
5  while not stopping condition is met do
6      if exchanged condition is met then
7          Exchange information between
           mSQDE_u and mSQDE_l according the
           considered scheme: (u, l, or w);
8      else
9          Iterate mSQDE_u and mSQDE_l
           according to the steps of Algorithm 2 ;
10     end
11 end
```

Algorithm 1: Main steps of the CoEvoMSQDE algorithm

4 Computational Experiments and Results

The main goals of the computational experiments are: to study the information exchange mechanism proposed and to analyze the performance of our coevolutionary approach for solving BDOPs.

In order to evaluate our approach, we need test problems that not only fit the model given in Eq. 10, but also involve the three scenarios identified in Sec. 2.3. To the best of our knowledge, such test problems are not available, thus we will use existing dynamic functions for the upper level and lower level subproblems.

```
1  Apply exclusion principle;
2  Detect changes in the environment;
3  if not change is detected then
4      for each subpopulation k do
5          Evolve conventional individuals
           according the steps of differential
           evolution metaheuristic;
6          Generate quantum individuals
           according the self-adaptive strategy
           proposed in [22] - (Eq. 13);
7          Update the global best solution of
           subpopulation k;
8          Update the global best solution of the
           algorithm;
9      end
10 end
11 else
12     for each subpopulation k do
13         Reevaluate subpopulation k
           conventional individuals;
14     end
15 end
```

Algorithm 2: Iteration process of the $mSQDE$ algorithm.

In this sense, one suitable candidate is the well-known Moving Peaks Benchmark (MPB) [4], specially the so-called *Scenario 2* which offers a multimodal objective function composed of several *peaks*.

In turn, every peak i is defined by a height (H_i), a width (W_i) and a position (X_i), which change after certain time steps. So, the objective function is defined as follows:

$$MPB(x) = \max_i \{H_i - W_i \cdot f_p(X_i - x)\}, \quad (14)$$

where f_p is the *peak function*, which gives a specific shape to the peaks.

In general, f_p is a minimization function with optimal solution at $x^* = 0$ and $f(x^*) = 0$. Note that in the MPB, the system control parameters ϕ are the heights, widths and positions of the peaks. These features change after some Δe function evaluations and according to a predefined severity. More details on the MPB can be found in [4].

Based on the model of DBOPs given in (10) and the MPB's objective function of Eq. 14 we can derive the following scenarios:

$$DBOP_{upper} := \begin{cases} \max\limits_{x\in\Omega, y\in\Gamma} MPB_u(x,t) + MPB_l(y), \\ \text{subject to} \\ \qquad \max\limits_{y\in\Gamma} MPB_l(y), \end{cases} \tag{15}$$

$$DBOP_{lower} := \begin{cases} \max\limits_{x\in\Omega, y\in\Gamma} MPB_u(x) + MPB_l(y,t), \\ \text{subject to} \\ \qquad \max\limits_{y\in\Gamma} MPB_l(y,t), \end{cases} \tag{16}$$

$$DBOP_{both} := \begin{cases} \max\limits_{x\in\Omega, y\in\Gamma} MPB_u(x,t) + MPB_l(y,t), \\ \text{subject to} \\ \qquad \max\limits_{y\in\Gamma} MPB_l(y,t). \end{cases} \tag{17}$$

Here, objective functions MPB_u and MPB_l are two different instances of Eq. 14, that is, with different dynamics and system control parameters. Note that the dynamism in the MPB functions is denoted by including the time t as an argument. So, the model of Eq. 15 (resp. Eq. 16) represents a BDOP with a dynamic lower-level (resp. upper-level) subproblem. On the other hand, the model of Eq. (17) involves dynamic objective functions at both subproblems. It is worth noting that model $DBOP_{lower}$ is not dynamic only at the lower-level subproblem, because the dynamic MPB_l is also present in the upper-level objective function as a summand. However, we have considered this case, since by adding the lower-level MPB_l to the upper-level function we obtain a bi-level relationship between both models. Otherwise, we would have two independent models which can be solved independently.

4.1 Description of the Experiments

We divided the experiments in two groups according to our goals:

1. The effect of the exchange mechanisms in the three BDOPs scenarios and
2. The performance of the best variants of CoEvoMSQDE algorithm in more complex scenarios.

In the first group we tested the exchange mechanisms previously described, in the scenarios $DBOP_{upper}$, $DBOP_{lower}$ and $DBOP_{both}$. Then, the best CoEvoMSQDE variants from this analysis, are tested in more complex scenarios.

Table 1 contains the parameters setting used for the subproblem instances MPB_u and MPB_l that are used in scenarios $DBOP_{upper}$,$DBOP_{lower}$ and $DBOP_{both}$.

Table 1. Parameters setting for subproblem instances MPB_u and MPB_l in the DBOP scenarios.

Parameter	Setting
Dimension (D)	5
Search space (Ω)	$[0, 100]^5$
Number of peaks	10
Peak heights (H_i)	$\in [30, 70]$
Peak widths (W_i)	$\in [1, 12]$
Peak function (f_p)	$f_{cone}(X) = \sqrt{\sum_{d=1}^{D} X_d^2}$
Shift severity (sev)	1.0
Change frequency (Δe)	5000
Correlation coefficient (λ)	1.0

Regarding the parameters setting of the CoEvoMSQDE algorithm, note that we have two levels. At the top level, we have the exchange mechanism (that we will study next), while at the bottom level, we have the mSQDE instances which will use the same parameter settings suggested by [22]. Specifically, the mSQDE instances will be composed of 10 sub-populations, each having 10 individuals (i.e. 5 conventional and 5 quantum ones). The scaling factor and the mutation probability affecting the self-adaptive strategy, were defined as $\lambda = 0.3$ and $\tau = 0.5$, respectively.

Defining a suitable performance measure for assessing the behavior of an algorithm, in both bi-level and dynamic optimization environments, is currently an active research area. In the case of bi-level optimization, [30] suggests employing *error rates* for the upper-level and lower-level objective functions in case the true optimums of both functions are known. When the optimum is not known, in [15] proposed *rationality*-based measures.

On the other hand, in dynamic environments several measures exist, the *offline error* [4] and the *best error before the change* [17], being two of the most employed. While the offline error indicates the average performance of the algorithm at every time step, the best error before the change only takes into account the last time step before a new change occurs in the environment. In any case, choosing the right measure primarily depends on the research goal. In our case, we are focused on assessing the algorithm performance in those time periods where the problem remains unchanged, so the best error before the change results appropriate. Formally, this measure is defined as:

$$e_{bc} = \frac{1}{C}\sum_{c=1}^{C} |f_{opt}(c) - f_{best}(c)|, \qquad (18)$$

where C is the number of changes in one run and $f_{opt}(c)$ and $f_{best}(c)$ are the objective function values for the problem optimum and the best solution of the algorithm before the change c, respectively.

We performed 30 runs for each pair of problem-algorithm instance, using different random seeds. Besides, we assumed that each problem instance would change 100 times every Δe function evaluation.

4.2 Influence of the Exchange Mechanism

In this group of experiments, the goal is to analyze the influence of the exchange mechanisms considered in the CoEvoMSQDE algorithm. Remember that the exchange mechanisms are composed of the *when*, the *what* and the *how* strategies.

Statistically speaking, such strategies will be the factors of the experiments and we consider three levels for these factors. In the case of the *what* and *how* the strategies described in Sec. 3 were selected, while for the case of *when*, the number of iterations between exchanges will be 1, 10 and 20. The combination of these levels lead to $3 \times 3 \times 3 = 27$ algorithm instances or variants. Each one will be noted as $when + what + how$. For example, the CoEvoMSQDE instance with an exchange process performed after 1 iteration, using the global best solution as exchanged information (g) and with preference on the upper-level algorithm, will be referred to as $1 + g + u$.

These 27 variants were tested in scenarios $DBOP_{upper}$, $DBOP_{lower}$ and $DBOP_{both}$. Then the results in terms of the error before the change, were statistically analyzed using the non-parametric Friedman test, according to the suggestions of [10]. This test allows us to identify differences among the algorithms and provides an average rank for each algorithm, where that the lower the rank, the better the algorithm.

Figure 5 shows the average ranks obtained by the 27 variants that we considered. Note that we have divided the analysis in four main groups: results in scenario $DBOP_{upper}$ (Fig. 5-a), results in scenario $DBOP_{lower}$ (Fig. 5-b), results in scenario $DBOP_{both}$ (Fig. 5-c) and results by considering all problem instances (Fig. 5-d).

In these graphs we highlighted with a black bar those variants with the best average rank. For instance, in the scenario $DBOP_{upper}$, the best variant is $10+g+l$, while in the case of $DBOP_{lower}$ there are 3 variants, i.e., those of the form $20 + g + \{u, l, w\}$. However, for scenario $DBOP_{both}$ and for all problem instances, the best ranked variant is again the $10 + g + l$. In order to analyze whether these best variants are statistically different from the rest, we relied on the Holm's post-hoc test. In this sense, graphs from Fig. 5 display, using a lighter dark tonality, those variants that are not different from the best. Observe that, in scenarios like $DBOP_{upper}$ and $DBOP_{lower}$ there are 11 variants with similar performance.

Despite the relevance of these specific conclusions, the major observation here is the influence of *what* information is exchanged and *when*. For example, when the exchange is made every 1 iteration, the variants performance is low,

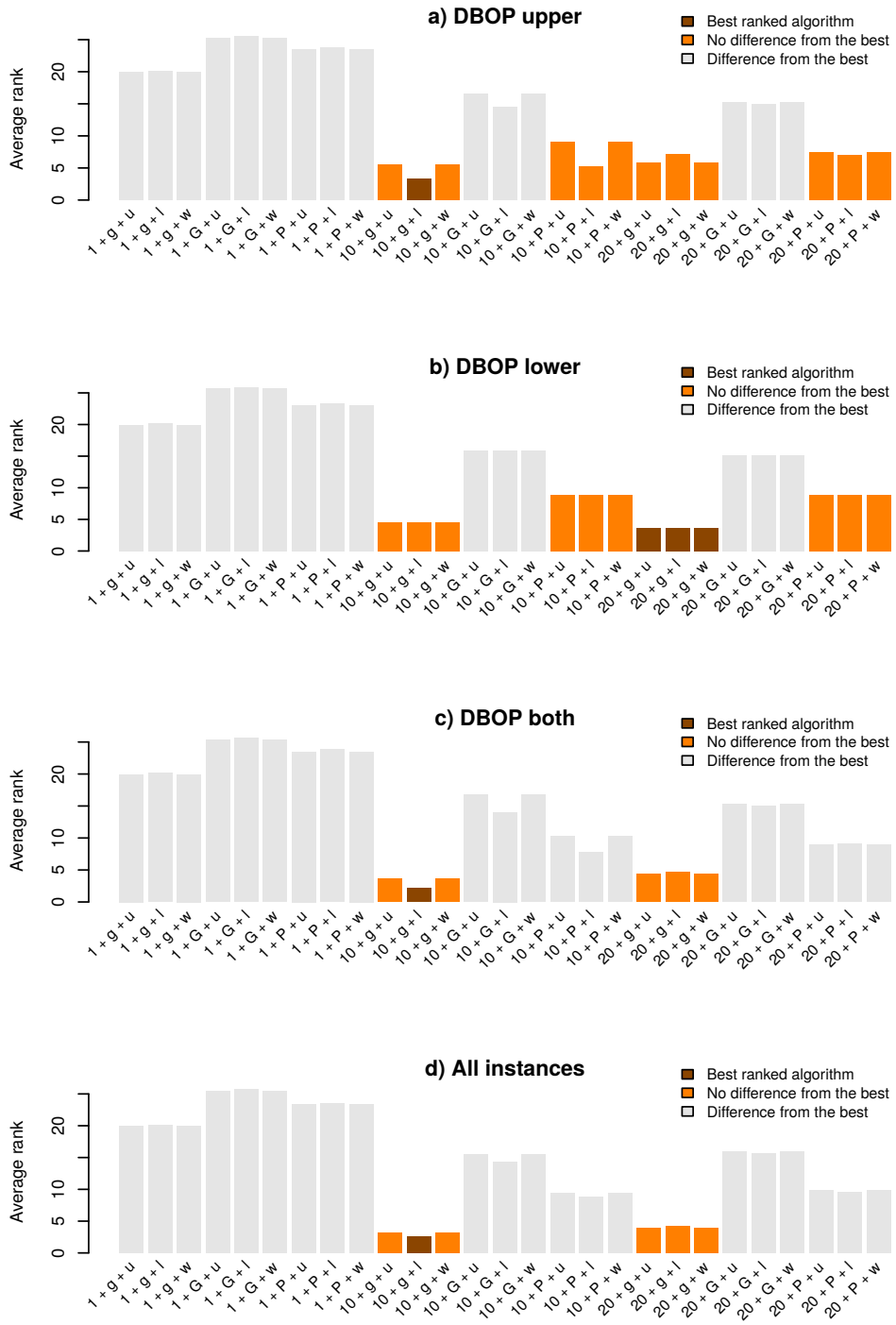

Fig. 5. Effects of different coevolutionary schemes for each BDOP type, in terms of the average ranking of the variants according to the Friedman test ($\alpha = 0.05$). Comparisons against the best algorithm are calculated using Holm's test ($\alpha = 0.05$)

regardless of the problem instance. This could be an obvious fact if we take into account that both, $mSQDE_u$ and $mSQDE_l$ instances do not have enough time to improve their respective searches and thus having their best solutions frequently replaced.

On the contrary, variants with exchanges every 10 and 20 iterations are much better, since the populations have more time to evolve. To better understand this aspect, recall that the problem instances we considered change every 5000 function evaluations. On the other hand, our tested variants used 202 function evaluations at every iteration (since $mSQDE_u$ and $mSQDE_l$ count on a population size of 100 individuals and an additional function evaluation is performed by the change detection mechanism). Hence, the number of performed iterations before the change is $5000/202 \approx 24$. It means that variants with the exchange process performed every 10 iterations, have at most $24/10 \approx 2$ exchanges before the change, while variants of the form $20 + ...$ only one exchange. This also explains why $10 + ...$ variants are better than $20 + ...$

Similarly, the type of information exchanged has a relevant impact on the algorithm's performance. From the results obtained, it is worth noting that exchanging the global best solution (g) between $mSQDE_u$ and $mSQDE_l$ instances is the best strategy, followed by the P and G. To see such a difference, Fig. 6 shows the evolution over 100 changes in the environment, of the best fitness before the change for variants $10+g+l$, $10+G+l$ and $10+P+l$.

The plots in left column (e.g. Fig. 6-a, c and e), correspond to the upper-level subproblem. On the other hand, the plots in the right column (e.g. Fig. 6-b, d and f), show the evolution of this measure for the lower-level subproblem, where the objective function is $f(y) = MPB_l(y)$. From these graphs it can be observed that g variant achieves the best approach to the problem optimum over time. In the case of the P variant note that it has a similar performance, but with an oscillating behavior, being more pronounced for the G variants. Such behavior is better observed in scenario $DBOP_{upper}$ (Fig. 6-b), where the lower-level subproblem is stationary along the run.

Finally, in contrast with the *when* and *what* schemes analysis, results showed that no substantial differences exist regarding *how* to perform the exchange. However, a slight advantage is observed for the l strategy, that is, with preference on the lower sub-algorithm. Note that this result holds for all scenarios. So far, the above conclusions are based on very basic scenarios. So, it would be interesting to verify whether these "best" variants are also successful in other, more complex scenarios.

The experiments in the next section will focus on this aspect.

4.3 Results in More Complex Scenarios

Based on the previous results, we will explore the performance of successful variants of the proposed method over more complex scenarios. Specifically, we focus on the variants using: $\{10, 20\} + \{g, P\} + \{l, u\}$. The combination of these schemes correspond to eight different variants, where the best ones from the previous section have been included.

In this experiments, we just focus on the $DBOP_{both}$ scenario. One easy way to derive more complex problem instances from $DBOP_{both}$ is to use different peak functions and search space dimensions in the subproblems.

We consider the following peak functions:

$$f_{sphere}(X) = \sum_{d=1}^{D} X_d^2, \tag{19}$$

$$f_{quadric}(X) = \sum_{d=1}^{D} \left(\sum_{i=1}^{d} X_i \right)^2, \tag{20}$$

$$.f_{schwefel}(X) = \sum_{d=1}^{D} |X_d| + \prod_{d=1}^{D} |X_d|. \tag{21}$$

In the former case, we consider different combinations of these functions (including f_{cone}), for the upper-level and lower-level subproblems, thus we obtain 16 new different instances based on the $DBOP_{both}$ scenario. In these cases, the problem size is 10 (with 5 variables in the lower and upper levels).

ISSN 2007-9737

Table 2. Mean of the best error before the changes ± standard error in the $DBOP_{both}$ scenario with different peak functions in the lower-level and upper-level subproblems, for several variants of the algorithm CoEvoMSQDE

Upper-level f_p	Lower-level f_p	10+g+u	10+g+l	10+P+u	10+P+l	20+g+u	20+g+l	20+P+u	20+P+l
Cone	Cone	3.58±0.10	**3.46±0.08**	5.51±0.14	5.30±0.11	3.76±0.09	3.88±0.09	5.73±0.11	5.54±0.09
	Sphere	2.71±0.09	**2.63±0.09**	4.69±0.13	4.42±0.11	3.00±0.07	3.08±0.08	5.07±0.08	4.76±0.10
	Quadric	3.51±0.11	**3.45±0.10**	5.55±0.16	5.29±0.12	3.83±0.08	3.77±0.07	6.00±0.13	5.64±0.12
	Schwefel	4.69±0.11	**4.51±0.10**	6.91±0.25	6.58±0.12	4.86±0.10	4.88±0.09	6.95±0.10	6.97±0.11
Sphere	Cone	2.17±0.07	**1.99±0.04**	4.54±0.23	3.93±0.13	2.64±0.06	2.55±0.05	4.79±0.12	4.45±0.10
	Sphere	1.47±0.06	**1.24±0.05**	3.65±0.27	2.99±0.08	1.91±0.06	1.79±0.06	3.85±0.18	3.77±0.11
	Quadric	2.35±0.07	**2.16±0.07**	4.77±0.27	4.10±0.14	3.45±0.19	3.32±0.18	5.95±0.28	5.87±0.26
	Schwefel	3.30±0.09	**3.12±0.08**	6.09±0.40	4.99±0.10	3.71±0.09	3.61±0.10	6.28±0.22	5.61±0.20
Quadric	Cone	3.41±0.09	**3.14±0.07**	8.71±2.03	5.38±0.15	3.71±0.07	3.62±0.08	6.16±0.16	5.95±0.11
	Sphere	2.65±0.09	**2.42±0.07**	7.42±0.97	4.59±0.11	3.01±0.08	2.90±0.08	5.90±0.38	5.20±0.08
	Quadric	3.48±0.09	**3.30±0.08**	7.64±0.96	5.75±0.12	4.69±0.18	4.61±0.19	7.61±0.35	7.16±0.22
	Schwefel	4.46±0.11	**4.28±0.11**	8.71±0.61	6.56±0.16	4.72±0.10	4.56±0.12	7.56±0.14	7.19±0.14
Schwefel	Cone	4.87±0.15	**4.51±0.11**	8.52±0.72	6.92±0.13	5.40±0.14	5.43±0.14	7.86±0.13	7.78±0.13
	Sphere	3.99±0.11	**3.76±0.12**	7.38±0.62	6.12±0.15	4.95±0.14	4.93±0.15	7.60±0.17	7.40±0.19
	Quadric	6.37±1.44	**4.77±0.14**	12.09±1.84	7.41±0.19	7.20±0.30	7.19±0.30	10.11±0.39	10.02±0.41
	Schwefel	5.81±0.16	**5.49±0.13**	8.73±0.34	8.03±0.21	6.36±0.14	6.33±0.13	9.08±0.19	8.97±0.13

Values in bold-face correspond to the best variant.

Table 3. Mean of the best error before the changes ± standard error in the $DBOP_{both}$ scenario with different dimensions in the lower-level and upper-level subproblems, for several variants of the algorithm CoEvoMSQDE.

Upper-level D	Lower-level D	10+g+u	10+g+l	10+P+u	10+P+l	20+g+u	20+g+l	20+P+u	20+P+l
2	2	1.16±0.07	**0.62±0.03**	2.49±0.09	1.85±0.06	1.00±0.04	0.85±0.03	2.23±0.04	2.12±0.06
	5	1.82±0.06	**1.43±0.03**	3.28±0.06	2.74±0.04	1.81±0.04	1.55±0.03	3.30±0.06	3.13±0.04
	8	3.36±0.15	**2.99±0.11**	5.02±0.18	4.46±0.12	3.29±0.09	3.33±0.09	4.87±0.12	4.65±0.11
	11	8.98±0.47	7.30±0.22	10.58±0.48	8.31±0.21	7.85±0.31	**7.21±0.26**	10.14±0.47	8.96±0.24
5	2	2.85±0.11	**2.61±0.08**	4.39±0.09	4.17±0.08	3.14±0.08	3.27±0.08	4.91±0.08	4.67±0.10
	5	3.58±0.10	**3.46±0.08**	5.51±0.14	5.30±0.11	3.76±0.09	3.88±0.09	5.73±0.11	5.54±0.09
	8	5.18±0.14	**5.02±0.12**	7.43±0.21	7.00±0.15	5.78±0.14	5.78±0.11	7.66±0.17	7.61±0.14
	11	9.63±0.36	**8.76±0.28**	12.76±0.49	11.42±0.22	9.65±0.28	9.13±0.20	12.96±0.29	12.31±0.20
8	2	4.84±0.10	**4.50±0.09**	6.37±0.16	5.87±0.12	4.76±0.11	4.77±0.11	6.31±0.10	6.15±0.11
	5	5.26±0.11	**4.94±0.09**	7.30±0.16	6.77±0.11	5.57±0.10	5.59±0.10	7.40±0.11	7.49±0.11
	8	7.21±0.17	**6.64±0.15**	9.55±0.23	8.82±0.19	7.38±0.18	7.41±0.16	9.15±0.13	9.14±0.17
	11	12.61±0.40	**10.80±0.22**	15.75±0.56	12.80±0.23	12.18±0.35	11.81±0.31	14.16±0.18	13.44±0.25
11	2	8.88±0.12	**8.54±0.10**	10.92±0.14	10.16±0.16	8.96±0.11	9.04±0.09	10.22±0.15	10.28±0.13
	5	9.40±0.15	**9.24±0.15**	11.67±0.14	11.27±0.13	9.96±0.11	9.95±0.09	11.31±0.16	11.15±0.18
	8	11.28±0.19	**10.68±0.14**	13.30±0.19	12.48±0.15	11.49±0.19	11.34±0.15	13.44±0.18	13.23±0.18
	11	17.00±0.65	**14.82±0.24**	19.55±0.56	17.38±0.31	15.71±0.30	15.12±0.27	18.62±0.32	18.41±0.29

Values in bold-face correspond to the best variant.

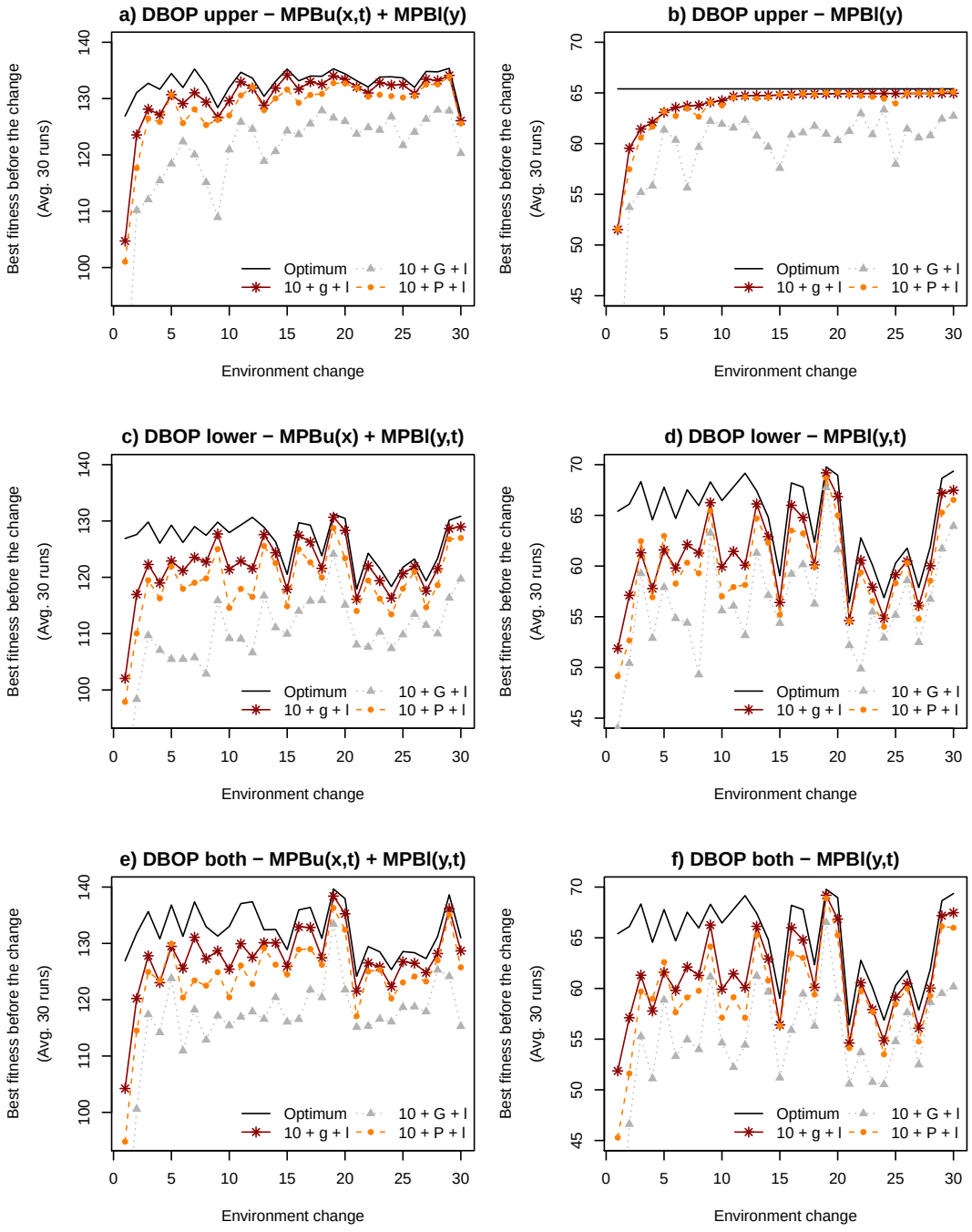

Fig. 6. Evolution of the best error before the change for variants of the form $10 + \{g, G, P\} + l$ for the upper-level and lower-level subproblems, in scenarios $DBOP_{upper}$, $DBOP_{lower}$ and $DBOP_{both}$

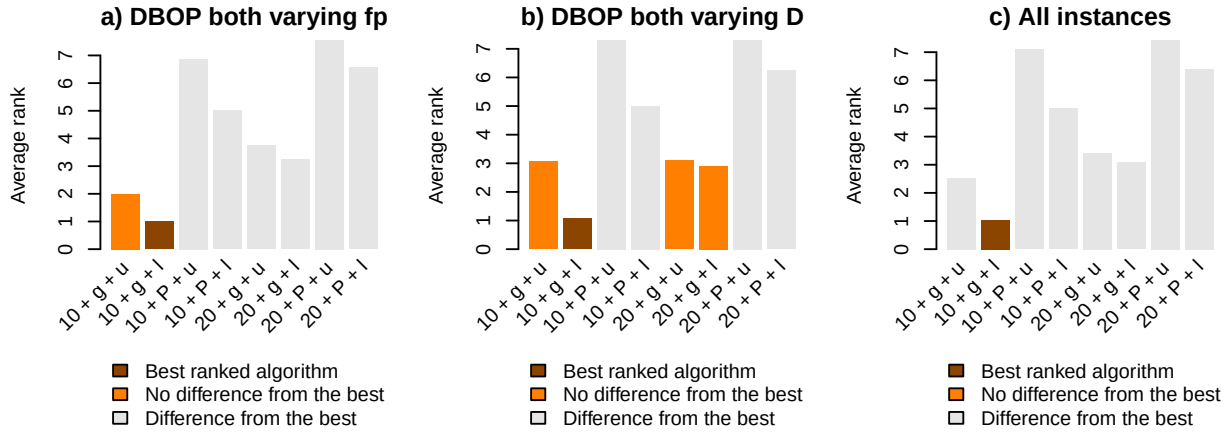

Fig. 7. Statistical results from Friedman and Holm tests ($\alpha = 0.05$) for the variants of the form $\{10, 20\} + \{g, P\} + \{u, l\}$ in $DBOP_{both}$ scenario, varying the peak function (16 different problems) and dimensions(also 16 different problems)

Next and using just the function f_{cone}, we consider different combinations of problems sizes at the lower and upper level.

The possible dimensions are $D = \{2, 5, 8, 11\}$, thus leading to 16 problem instances.

The results in terms of the average best error before the change are shown in Tables 2 and 3.

Similar conclusions to the previous experiments can be drawn. For instance, note that the best variant is $10 + g + l$ for both groups of problem instances, even for the more complex ones (final rows of the tables).

In order to statistically confirm these results, we again apply the Friedman test. The average rank for each method variant in both groups of problem instances is given in Fig. 7-a) and b). Note that we also extend the analysis by considering the results in all the problem instances (Fig. 7-c)). In all cases, the $10 + g + l$ variant is found as the best algorithm. However, according to the Holm's test, its superiority is only statistically significant when all problem instances are considered.

5 Conclusion and Future Works

In this paper a hybrid approach for solving dynamic bi-level optimization problems (DBOPs) is proposed. Specifically, we focused on combining a coevolutionary scheme with a multipopulation mSQDE algorithm specifically designed for dynamic environments. While the coevolutionary algorithm deals with the bi-level feature of the problem, the mSQDE deals with the dynamic optimization of the upper-level and lower-level subproblems.

Several mechanisms for performing the information exchange between the mSQDE instances, were studied. Overall, the results from the computational experiments revealed that, for the scenarios considered, the decisions stating *what* kind of information and *when* the exchange process is made, have a more important impact in the algorithm performance than *how* the information exchange is done.

In order to further promote the research in this direction, we included the tested algorithms and problems in the recently proposed tool DynOptLab [23]. The reader can find the related source code at DynOptLab website[1].

Acknowledgements

Authors has the support of a FOCICYT project from the Technical State University of Quevedo, Ecuador.

[1] http://modo.ugr.es/DynOptLab

References

1. **Alba, E., Nakib, A., & Siarry, P. (2013).** *Metaheuristics for Dynamic Optimization*, volume 433 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg.
2. **Blackwell, T. & Branke, J. (2006).** Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 4, pp. 459–472.
3. **Boussaïd, I., Lepagnot, J., & Siarry, P. (2013).** A survey on optimization metaheuristics. *Information Sciences*, Vol. 237, No. 0, pp. 82–117.
4. **Branke, J. (1999).** Memory enhanced evolutionary algorithms for changing optimization problems. **Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., & Zalzala, A.**, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, IEEE Press, pp. 1875–1882.
5. **Camacho-Vallejo, J.-F., González-Rodríguez, E., Almaguer, F.-J., & González-Ramírez, R. G. (2015).** A bi-level optimization model for aid distribution after the occurrence of a disaster. *Journal of Cleaner Production*, Vol. 105, pp. 134–145.
6. **Chen, Z. & Song, Z. (2012).** Dynamic portfolio optimization under multi-factor model in stochastic markets. *OR Spectrum*, Vol. 34, No. 4, pp. 885–919.
7. **Colson, B., Marcotte, P., & Savard, G. (2005).** Bilevel programming: A survey. *4OR*, Vol. 3, No. 2, pp. 87–107.
8. **Cruz, C., González, J. R., & Pelta, D. (2011).** Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, Vol. 15, No. 7, pp. 1427–1448.
9. **du Plessis, M. C. & Engelbrecht, A. P. (2012).** Using competitive population evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, Vol. 218, No. 1, pp. 7–20.
10. **García, S., Molina, D., Lozano, M., & Herrera, F. (2009).** A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *J Heuristics*, Vol. 15, pp. 617–644.
11. **Hansen, P., Jaumard, B., & Savard, G. (1992).** New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 5, pp. 1194–1217.
12. **Jeroslow, R. (1985).** The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, Vol. 32, No. 2, pp. 146–164.
13. **Jin, Y. & Branke, J. (2005).** Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, Vol. 9, No. 3, pp. 303–317.
14. **Kocvara, M. & Outrata, J. (2006).** Effective reformulations of the truss topology design problem. *Optimization and Engineering*, Vol. 7, No. 2, pp. 201–219.
15. **Legillon, F., Liefooghe, A., & Talbi, E.-G. (2013).** Cobra: A coevolutionary metaheuristic for bi-level optimization. In **Talbi, E.-G.**, editor, *Metaheuristics for Bi-level Optimization*, volume 482 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, pp. 95–114.
16. **Li, C., Nguyen, T. T., Yang, M., Yang, S., & Zeng, S. (2015).** Multi-population methods in unconstrained continuous dynamic environments: The challenges. *Information Sciences*, Vol. 296, No. 0, pp. 95–118.
17. **Li, C., Yang, S., Nguyen, T. T., Yu, E. L., Yao, X., Jin, Y., Beyer, H.-G., & Suganthan, P. N. (2008).** Benchmark generator for cec'2009 competition on dynamic optimization. Technical report, Department of Computer Science, University of Leicester, U.K.
18. **Linnala, M., Madetoja, E., Ruotsalainen, H., & Hämäläinen, J. (2012).** Bi-level optimization for a dynamic multiobjective problem. *Engineering Optimization*, Vol. 44, No. 2, pp. 195–207.
19. **Marinakis, Y. & Marinaki, M. (2013).** A bilevel particle swarm optimization algorithm for supply chain management problems. In **Talbi, E.-G.**, editor, *Metaheuristics for Bi-level Optimization*, volume 482 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, pp. 69–93.
20. **Meyer-Nieberg, S. & Beyer, H.-G. (2007).** Self-adaptation in evolutionary algorithms. In **Lobo, F., Lima, C., & Michalewicz, Z.**, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, pp. 19–46.
21. **Nguyen, T. T., Yang, S., & Branke, J. (2012).** Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, Vol. 6, No. 0, pp. 1 – 24.

22. **Novoa-Hernández, P., Corona, C. C., & Pelta, D. A. (2013).** Self-adaptive, multipopulation differential evolution in dynamic environments. *Soft Computing*, Vol. 17, No. 10, pp. 1861–1881.
23. **Novoa-Hernández, P., Corona, C. C., & Pelta, D. A. (2015).** A software tool for assisting experimentation in dynamic environments. *Applied Computational Intelligence and Soft Computing*, Vol. 2015, pp. 1–12. Article ID 302172.
24. **Novoa-Hernández, P., Corona, C. C., & Pelta, D. A. (2016).** Self-adaptation in dynamic environments - a survey and open issues. *International Journal of Bio-Inspired Computation*, Vol. 8, No. 1, pp. 1–13.
25. **Oduguwa, V. & Roy, R. (2002).** Bi-level optimisation using genetic algorithm. *Artificial Intelligence Systems, 2002. (ICAIS 2002). 2002 IEEE International Conference on*, pp. 322–327.
26. **Sinha, A., Malo, P., & Deb, K. (2014).** Test problem construction for single objective bilevel optimization. *Evolutionary Computation Journal*, Vol. (In Press).
27. **Sinha, A., Malo, P., Frantsev, A., & Deb, K. (2014).** Finding optimal strategies in a multi-period multi-leader-follower stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, Vol. 41, No. 0, pp. 374 – 385.
28. **Storn, R. & Price, K. (1997).** Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341–359.
29. **Sun, D., Benekohal, R., & Waller, S. (2006).** Bi-level programming formulation and heuristic solution approach for dynamic traffic signal optimization. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 21, No. 5, pp. 321–333.
30. **Talbi, E.-G. (2013).** A taxonomy of metaheuristics for bi-level optimization. In **Talbi, E.-G.**, editor, *Metaheuristics for Bi-level Optimization*, volume 482 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, pp. 1–39.

Article received on 19/02/2017; accepted on 09/08/2017.
Corresponding author is Pavel Novoa-Hernández.