A Competitive Approach for Bi-Level Co-Evolution

Conference Paper · May 2018

DOI: 10.1109/IPDPSW.2018.00101

4 authors, including:

Emmanuel Kieffer
University of Luxembourg
27 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Grégoire Danoy
University of Luxembourg
139 PUBLICATIONS **693** CITATIONS
SEE PROFILE

Pascal Bouvry
University of Luxembourg
550 PUBLICATIONS **5,432** CITATIONS
SEE PROFILE

A Competitive Approach for Bi-level Co-evolution

Emmanuel Kieffer
SnT Interdisciplinary Centre
University of Luxembourg
emmanuel.kieffer@uni.lu

Grégoire Danoy, Pascal Bouvry
FSTC-CSC-ILIAS
University of Luxembourg
{firstname.lastname}@uni.lu

Anass Nagih
LCOMS Research Unit
University of Lorraine, France
anass.nagih@univ-lorraine.fr

***Abstract*—Real-life problems often involve several decision makers whose decisions impact each other. When two decision makers decides sequentially, these problems are referred to as bi-level optimization problems. Generally modeled as nested optimization problems, they are NP-hard even for two linear and continuous levels. Such problems often occur in situations were only a part of the decision variables is controlled by each decision makers. Their final objective value is thus subject to each other's decision. From the first decision maker point of view, it is necessary to predict the "rational reaction" of the second decision maker which may have a conflicting objective function. The first decision maker should therefore ensure that this reaction will not have a disastrous effect on its own final objective value. The inherent complexity of bi-level optimization problems led researchers to consider metaheuristics. Among the most promising metaheuristics, co-evolutionary algorithms proved their abilities to tackle large scale problems. Unfortunately, BOPs are naturally strongly epistatic. In this work, we propose an hybrid competitive co-evolutionary algorithm (CARBON) to tackle this pitfall. We compare CARBON against another co-evolutionary approach for bi-level optimization problems, i.e., COBRA. Experimental results demonstrate the abilities of CARBON to break the inherent nested structure that makes bi-level optimization problems so difficult.**

***Index Terms*—Competitive co-evolution, Bi-level optimization, GP hyperheuristics**

I. Introduction

Multi-level optimization problems have recently received a renew of interest. They are particularly well-suited for problems involving multiple and sequential decision makers. Bi-level optimization problems are special cases implying two decision makers and take their origins in game theory with non-cooperative Stackelberg Games [1]. They are two-stage models where the first player is called *leader* and the second player is called the *follower*. In these games, the payoff obtained by the leader strongly depends on the reaction of the follower. The leader has definitely an advantage over the follower by setting his decision first. However, he should be able to guess the resulting optimal strategy of the follower. This implies that a decision which may seem very promising for the leader can be deteriorated by the decision of the follower. Consequently, the leader has to take the follower decision into account. The mathematical models resulting from these games has a nested structure, i.e., the follower optimization problem is embedded inside the constraints of the leader. Bracken and McGill [2] proposed first this nested modeling which they called: "mathematical programs with optimization problems in the constraint". In the literature, many appellation have been used to designate the two levels. In the remainder of this paper, we will use the term "upper-level" and "lower-level". Upper-level decision maker will refer to the leader while lower-level decision maker will refer to the follower. The same principle will be used to describe the two levels and their associated optimization problems.

Many approaches have been proposed in the literature to solve continuous bi-level optimization problems. On the contrary, very few algorithms have been design to cope with combinatorial versions even if they are the most encountered problems in real applications. Metaheuristics have been proposed to cope with bi-level optimization complexity. Nevertheless most of them are based on a nested optimization scheme which repeatedly solves one level after the other. This scheme is very time consuming. A competitive co-evolutionary algorithm would be a trivial solution to solve bi-level optimization problems. Indeed, we would have two populations which will evolve competitively. One population would evolve a set of upper-level decisions according to the upper-level objective while the second one would evolve a set of lower-level decisions. However, the main issue with such a co-evolutionary scheme in bi-level optimization is the strong epsitatic links between the upper-level and lower-level decision variables. The nested structure is a drawback since each upper-level decision leads to a different lower-level instance and thus search space. Consequently, some upper-level and lower-level solutions may not be compatible when they are paired. In order to obtain independence between the two populations, we can consider another approach which consists in having one population representing the upper-level decisions and another one representing a set of algorithms which are evolved to solve any kind of lower-level instances. We not only evolve the upper-level decision but also the abilities of the algorithms to solve lower-level instances. We chose to evolve greedy heuristics since they are fast and easily modifiable. We experiment our new approach on the Bi-level Cloud Pricing problem which is a large scale combinatorial problem. We compare our numerical results with COBRA, a co-evolutionary algorithm designed to tackle bi-level optimization problems.

The remainder of this article is organized as follows. We first describe formally Bi-level Optimization. The related work section introduces bi-level optimization as well as co-evolution. Section 3 introduces CARBON, i.e., the proposed competitive co-evolutionary algorithm as well as the Bi-level Cloud Pricing Optimization. Experiment setups and results are discussed in

$$
\begin{aligned}
\min \quad & F(x,y) \\
\text{s.t.} \quad & G(x,y) \leq 0 \\
& \min \quad f(x,y) \\
& \text{s.t.} \quad g(x,y) \leq 0 \\
& x,y \geq 0
\end{aligned}
$$

Program 1: General Bi-level Optimization Program

section 4. Finally, the last section provides our conclusions and proposes some possible perspectives.

II. Bi-level Optimization

Program 1 illustrates the general mathematical formulation of bi-level optimization problems. The obvious nested structure presupposes that a bi-level feasible solution should be a lower-level optimal solution. This property makes bi-level optimization problems strongly NP-hard even though both levels are continuous linear problems.

where F, f : $\mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}, G : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^p$ and $\mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^q$.

Let us introduce some definitions and properties:

- The bi-level decision vector: (x, y)
- The upper-level decision vector: x
- The lower-level decision vector: y
- The upper-level objective function: $F(x, y)$
- The lower-level objective function: $f(x, y)$
- The upper-level set of constraints: $G(x, y) \leq 0$
- The lower-level set of constraints: $g(x, y) \leq 0$
- The constraint region of the general bi-level problem: $S = \{(x, y) : x \in X, y \in Y, G(x, y) \leq 0, g(x, y) \leq 0\}$
- The parametric lower-level optimization problem: $LL(x)$: $\min f(x, y) \quad s.t. \quad g(x, y) \leq 0$
- The feasible set for the lower-level problem parametrized by $x \in X$: $S_L(x) = \{y \in Y : g(x, y) \leq 0\}$.
- The projection of S onto the upper-level decision space: $Proj_{S(X)} = \{x \in X : \exists y \in Y, G(x, y) \leq 0, g(x, y) \leq 0\}$
- The lower-level rational decision set for $x \in S(X)$: $P(x) = \{\hat{y} \in Y : \hat{y} \in \arg\min[f(x, y) : y \in S_L(x)]\}$
- The Inducible Region: $IR = \{(x, \hat{y}) \in S, \hat{y} \in P(x)\}$

Program 1. describes a general bi-level problem where x is the decision vector controlled by the upper-level decision maker while y is the decision vector made by the lower-level decision maker. Each decision makers posses his own objective function, i.e., $F(x, y)$ for the upper-level decision maker and $f(x, y)$ for the lower-level decision maker. Notice that each objective function depends on the two levels of variables. Therefore, the upper-level decision maker must know the decision of the lower-level decision maker before computing $F(x, y)$. In addition, a nested problem can be designated as bi-level only if the decision of the upper-level decision maker impacts the decision of the lower-level decision maker. Else we could solve both problems separately. The lower-level problem is a parametric optimization problem ($LL(x)$) whose parameters are the upper-level decision variables. Its is very important for the upper-level decision maker to determine the consequences of his decision. For this purpose, he wishes to forecast the lower-level rational set, i.e., $P(x)$ which is a set representing all the optimal lower-level solutions according to the upper-level decision x. If for a given upper-level decision x, the lower-level decision maker has many optimal choice, we need to turn to set-valued optimization and more precisely from point to set mapping. Another solution is to consider two cases:

- The optimist case: we choose $\hat{y} = \arg\min\{F(x, y) : y \in P(x)\}$.
- The pessimist case: we choose $\hat{y} = \arg\max\{F(x, y) : y \in P(x)\}$.

We place our work in the optimistic case if $P(x)$ is not a singleton. Many works use this assumption since no optimality guaranties exist in the pessimistic case. The structure of Program 1. also highlights the fact that bi-level feasibility is determined by the lower-level optimality. As a result, IR represents all bi-level feasible solutions of the bi-level optimization problem. This is in this set that the upper-level decision maker is searching the best (or optimal) solution. The major difficulty lies in the fact that IR is not known a priori and has to be built during the optimization. Bi-level optimization problems are problems where the feasible search area is unknown and differs from S, i.e., the constraint region obtained from the intersection of both constraints sets. Basically, $IR \subset S$ and has to be determined, which is resource consuming. An interesting example (see 1) proposed by Mersha and Dempe [3] depicts the situation where upper-level constraints cause a discontinuous IR. In such a situation, an upper-level decision maker selecting $x = 6$ will observe a lower-level rational reaction $y = 12$. Unfortunately, $(x = 6; y = 12)$ is an non-feasible upper-level solution. Why? The lower-level decision maker is indifferent to the upper-level constraints. Nothing forces the lower-level decision maker to satisfy those constraints. However, the upper-level decision maker has to take the lower-level constraints into account since they are indirectly part of his constraint set. As a consequence, the upper-level decision maker may not end with a feasible solution.

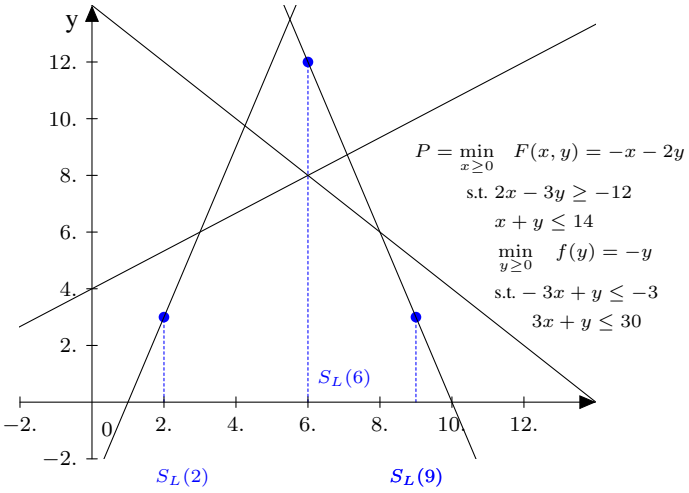

Fig. 1: Example of Non-discontinuous IR

III. Related work

Strongly related to game theory, bi-level optimization problems generalize Stackelberg games[1] introduced by H. von Stackelberg in 1952. Their current formulation has been introduced by the seminal work of Bracken and McGill in [2] who designated them in their works as "mathematical programs with optimization problems in the constraints". The first applications provided by [4] were related to defense applications. But we can find many other application domains where bi-level modeling has been exploited to represent mathematically problems. Transportation [5], planning [6] and management [7] are domains where bi-level optimization problems can typical be found in practice since they naturally involve several decision makers. Famous problems like the "Toll setting problems" have been intensively studied in the literature [8], [9], [10]. Some bi-levels models have been proposed to tackle environment economics problems [11],[12]. The main objective is to determine a tax policy that offers a compromise between revenues and environmental impacts. We can also find bi-level problems when designing the conditions of chemical reactions [13], [14]. Indeed, the amount of reactants transformed into products strongly depends on the initial conditions and the precise moment where the system reached an equilibrium state. The optimal design of structures can be also considered using bi-level modeling [15]. Structural optimization generally requires to minimize the weight or cost of a structure knowing that some forces and stresses are defined in terms of variational equalities. Control Theory also have problems which are bi-level by nature [16], [17]. Finally, bi-level optimization is particularly well-adapted for meta-optimization. For instance, evolutionary algorithms and machine learning techniques often requires a large number of parameters. The determination of those parameters can be achieved through bi-level optimization [18], [19].

Bi-level optimization problems have been proven strongly NP-hard even if both levels are continuous and linear [20]. Some exact approaches have been designed but remain unsuitable to tackle large scale applications. Nevertheless, it is very interesting to observe the different approaches proposed in the literature. One very relevant approach to cope with two nested optimization models is to reformulate the lower-level using its Karush-Kuhn-Tucker conditions [21]. This transformation provides a single-level optimization problem which can be then addressed with standard approaches. Obviously, it can only be applied on optimization problems with continuous decision variables. Another exact approach is based on a modified version of the simplex to enumerate vertices when dealing with linear bi-level optimization problems. A third category extracts gradient information for the lower-level problem in order to apply directional derivatives at the upper-level [22]. Three main categories can be identified for integer and mixed bi-level problems. The first one deals with reformulation by applying decomposition techniques [23] while the second one adapts the well-know Branch & Bound paradigm. Some works have been devoted to parametric optimization since the lower-level problem is a parametric optimization problem [24]. Finally, trusted-region approaches have been investigated in [25]. These methods attempt to approximate some regions of the objective function. When the approximation is accurate, the region is expanded, otherwise it is contracted. When dealing with complex and large bi-level optimization problems, metaheuristics come into play. Metaheuristics have been successfully and widely used in single-level optimization cases to tackle NP-hard problems. Since bi-level optimization problems are already NP-hard when both levels are linear, the scope of application has been extended. The classification of bi-level evolutionary algorithms is very similar to the one for exact optimization approaches. Among all the works from the literature, we can distinguish 5 resolution strategies:

1) Nested sequential (NSQ)
   - Repairing approach (REP)
   - Constructive approach (CST)
2) Single-level transformation (STA)
3) Co-evolutionary (COE)
4) Multi-objective (MOA)
5) Lower-level approximation (APP)

The legacy approaches belong to the NSQ categories. They are all based on nested and sequential approaches consisting in solving alternatively both levels. The REP approach handles the lower-level problem as a standard constraint and attempts to repair solutions if they do not solve efficiently the lower-level problem [26]. The CST is the most simple one. It solves sequentially both levels until meeting a certain stopping criterion[27]. Closely related to the transformations performed with exact approaches, the STA category makes use of the KKT conditions to obtain single-level problems that are then solved using an evolutionary algorithm [28]. We also observe in the literature some works where bi-level optimization problems are transformed into multi-objective but single-level level problems. Such transformations do not consist in using the lower-level objective as a second objective since a bi-level solution is not necessarily a Pareto optimal

solution. A new methodology has been introduced in [29] to ensure a perfect equivalence between the original bi-level optimization problems and their single-level multi-objective counterparts. A recent bi-level evolutionary algorithm based on quadratic approximations (BLEAQ) has been proposed by Sinha and Deb in [30]. This algorithm tries to approximate the lower-level decision variables using the upper-level decision variables in order to reduce the number of evaluations. Another approach based on surrogate-based optimization and proposed by Kieffer et al. in [31] has been employed to approximate the LL objective value instead of lower-level decision variables. Despite their interesting work-flow and results, methods based on lower-level approximation have only been designed to cope with continuous bi-level optimization problems.

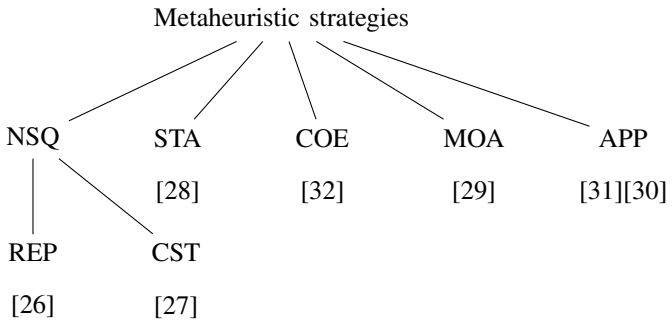

Fig. 2: Extended bi-level metaheuristics taxonomy

To the best of our knowledge, there have been very few attempts at designing bi-level co-evolutionary algorithms to tackle bi-level optimization problem. It is very difficult to break the nested structure especially when we want to apply a mechanism which relies on more or less on independent parts. In [32], the authors proposed to optimize both levels separately while keeping an exchange mechanism based on a random co-evolutionary operator. The latter is applied on both populations after a certain number of generation for both levels. Largely inspired by BIGA [33], the main difference lies in the fact that COBRA applies an upper and an lower improvement phase which evolves both populations independently during a certain number of generations. Finally, a more recent approach, CODBA, proposed in [34], aims at generating from the upper-level solutions many LL populations. The authors then evaluate in parallel each sub-population. Each individual of these LL populations mate using crossover with the best archived LL solutions until no more improvement occurs at LL. Although the authors categorize their algorithm as co-evolutionary, the fact that it uses a single thread reduce it to a simple nested optimization algorithm. The next section will detail our methodology to propose a co-evolutionary algorithm while breaking the nested structure.

IV. CARBON: A HYBRID BI-LEVEL CO-EVOLUTIONARY ALGORITHM

A. Motivations and description

Co-evolution occurs when intimate species influence each other's evolution. Several co-evolutionary models have inspired the evolutionary computing field. The predator/prey model is a famous example. We generally distinguish two classes of co-evolutionary algorithms: competitive and cooperative. Competitive co-evolution has been first proposed by Hillis in [35] for Sorting Networks. Then many applications stem from his results and notably in game theory where players often compete. Competition focuses on the abilities of species to evolve and develop new skills to outperform the other specie during an armed race. On the contrary, cooperative co-evolution relies on the skills emerging from species tending to collectively work to solve a common problem. The seminal work of Potter and De Jong [36] shows how cooperative co-evolution can be employed to optimize multi-dimensional functions. In [37], the authors used cooperative co-evolution as a constraint handling method by decomposing the constraint set through multiple populations. To the best of our knowledge, all bi-level co-evolutionary algorithms considered in the scientific literature rely on two populations. Each population attempts to evolve independently the upper-level and lower-level decision variables. Nevertheless, the lower-level decision strongly depends on the upper-level decision. For example, [31] and [30] rely on this fact to obtain either an approximation of the lower-level objective value or an approximation of the lower-level decision variables. Trying to optimize independently both levels can raise many issues. For example, it is possible that pairing the two level decisions x and y provides a non-feasible solution. Such an example has been provided in section II. In addition, the bi-level formulation indicates us that two upper-level decisions x_1 and x_2 can produce two incompatible lower-level instances. By incompatible, we mean non-overlapping lower-level search spaces. Consequently and according to the bi-level definition, two bi-level solutions $S_1 = (x_1, y_1)$ and $S_2 = (x_2, y_2)$ can be compared if and only if both solutions are bi-level feasible unless they share the same upper-level decision, i.e., $x_1 = x_2$. When handling constraints, metaheuristics often used some distance to the feasible search space to penalize non-feasible solutions. For bi-level optimization problems, it is very difficult to obtain such a distance measure since bi-level feasibility implies lower-level optimality. Therefore to compare two bi-level solutions, we should be able to measure how well they perform at the lower-level. Since each upper-level decision x generates a new lower-level instance, the lower-level optimal value should be taken relatively to x. In order to obtain a good measure of bi-level feasibility, we only need to find a lower-level optimality measure. In this purpose, we consider the lower-level gap to optimality which can be defined as follows:

$$\%\text{-}gap(x) = 100 * \frac{A(x) - LB(x)}{LB(x)} \tag{1}$$

where $A(x)$ is the lower-level solution value obtained after applying an algorithm A on the upper-level decision x. $LB(x)$ is a lower-bound according to the upper-level decision x.

Using the gap we can compare two solutions even if they did not stem from the same upper-level decision. The solution with the lowest gap indicates that the upper-level decision maker has a better estimation of the lower-level rational reaction. This estimation is essential for the upper-level decision maker to avoid an overestimation of his own objective value. Let us take the example described in section II. If the upper-level decision maker sets the UL decision $x = 6$ and by means of optimization obtains $y = 8$. The upper-level decision maker will wrongly suppose that $x = 6$ is the best upper-level decision to take. Unfortunately such a decision will inevitability lead the follower to choose $y = 12$ since it is the lower-level rational reaction to $x = 6$. In this case, the upper-level decision maker will not even obtain a feasible solution. Consequently, it is highly important to obtain the best forecast of the possible lower-level reactions.

In this work, CARBON is a competitive hybrid co-evolutionary algorithm. Two populations obey to the predator-prey model except that the second population will not directly evolve lower-level solutions but the mean to get to them. With this approach, we obtain a second population independent from any upper-level decision. The prey will be the upper-level decision variables and the predators will be the lower-level heuristics evolved to provide lower-level rational reactions with low gap. According to Fig. 3, upper-level decision variables are evolved using evolutionary operators that can be found typically in Genetic Algorithm (GA) while heuristics are evolved using Genetic Programming (GP) operators. The goal of this second population is basically to learn a set of accurate heuristics. Generation of heuristics by mean of Genetic programming is called GP hyper-heuristics. Contrary to standard hyper-heuristics algorithms that are based on several pre-existing heuristics, GP hyper-heuristics allow to build heuristics from scratch. Initiated by Burke et al. [38], this approach enables the automation of the assembly of heuristics and have encountered a real success in cutting and packing [39], function optimization [40] and other additional domains [41],[42],[43]. Interested readers can find more application cases and a more detailed description of GP hyper-heuristics in the survey by Burke et al. [38]. This article proposes CARBON, the first algorithm for heuristic generation through a competitive co-evolutionary process.

CARBON attempts to find the optimal upper-level solutions while finding the best strategies (heuristics) to determine the lower-level rational reaction. We need to adopt the upper-level decision maker point of view. When we face a competitive problem where the decision of the opponent has direct consequences on our payoff, we always wish to determine the impact of our decision first. Therefore, we look for strategies that can be adopted by this opponent. Here, the lower-level decision maker takes his decision after our decision and is assumed to be rational. This concept is embedded in CARBON. While the first population will evolve the upper-level decision variables with standard genetic operators, heuristic generation requires more explanations. Since heuristics are problem specific, we will first introduce the real application case considered in this work, i.e., the Bi-level Cloud Pricing Optimization Problem.

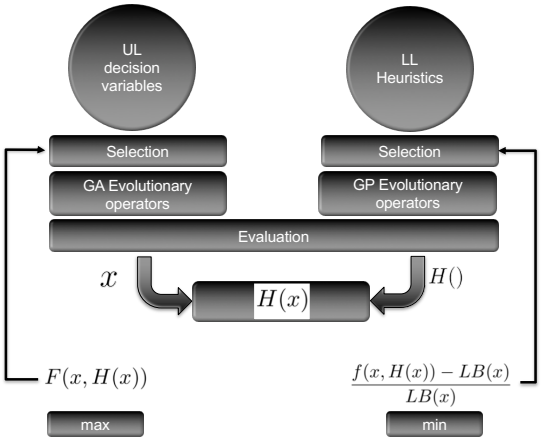

Fig. 3: CARBON workflow

B. Application to the Bi-level Cloud Pricing Optimization Problem

The Bi-level Cloud Pricing Optimization problem (BCPOP) models the need for a Cloud Service Provider (CSP) to determine an optimal pricing in a very competitive market knowing that Cloud Service Customers (CSC) buy the offers minimizing their total cost. The upper-level decision maker is clearly the CSP while a CSC can represent a lower-level decision maker. Generally, CSPs tend to sell services in bundles (i.e., sets). This strategy generally complexifies the decision taken by CSCs who should ensure that all their needs are covered while minimizing their total cost. For the sake of simplicity, we will consider a single rational CSC. Program 1 illustrates the BCPOP mathematical model. We take the point of view of a CSP who wants to sell his bundles on a dense market of services sold as bundles. $x_j \in \{0, 1\}$, $\forall j \in \{1, ..., M\}$ are the lower-level decision variables and indicate whether or not the lower-level decides to buy bundle j on the market. $c_j \in \{0, 1\}$, $\forall j \in \{1, ..., M\}$ are the prices of bundles. The first L bundles are the ones belonging to our CSP. At the upper-level, they represent the upper-level decision variables which become parameters/coefficients at lower-level. b^k, $\forall k \in \{1, ..., N\}$ represents the service requirements for services k while q_j^k expresses the number of services k occurring in the bundle j. In addition, q_j^k, $\forall k \in \{1, ..., N\}$ is a column-vector of size N corresponding to the distribution of all services in bundle j.

As it can be observed in Program. 2, the lower-level problem is covering problem. Covering problems are NP-hard. Large scale versions are generally tackled using heuristics or meta-heuristics. A possible greedy heuristic could rely on some scoring function that permits to sort bundles. According to this scoring function, the CSC adds each bundle inside his

$$
\begin{aligned}
\max \quad & F = \sum_{j=1}^{L} c_j x_j \\
\text{s.t.} \quad & \min \quad f = \sum_{j=1}^{M} c_j x_j \\
& \text{s.t.} \quad \sum_{j=1}^{M} q_j^k x_j \geq b^k \quad \forall k \in \{1, ..., N\} \\
& c_j \geq 0 \qquad\qquad\qquad \forall j \in \{1, ..., L\} \\
& x_j \in \{0, 1\} \qquad\qquad \forall j \in \{1, ..., M\}
\end{aligned}
$$

Program 2: Bi-level Cloud Pricing Program

basket until all service requirements are satisfied. Of course, he wishes to minimize his total cost. One strategy would be to find the best scoring function providing an order minimizing this total cost. A GP hyper-heuristic approach could answer this question. Indeed this function could be built using some discriminant features like the cost c_j, the distribution of service inside a bundle q_j and so forth. The original data from the problem are thus the building blocks of this function and are officially called the "Terminal set". The scoring function will have some operators to combine these features together which we call the "Operator set". The goal of the GP is to encode a population of scoring functions as a population of syntax trees where each terminal node is picked in the "Terminal set" and each intermediate node is picked in the "Operator set". According to Fig. 3, the upper-level population will evolve possible pricings and the lower-level population will evolve scoring functions. These functions, once embedded in the aforementioned greedy heuristic, will constitute valid heuristics to solve the lower-level instances parametrized with the CSC pricings. Table I describes the Terminal and Operator sets that we consider to solve the lower-level covering problem embedded in the BCPOP constraints. Notice that we consider the dual values and relaxed optimal solution after continuous relaxation. Indeed, continuous relaxation will be in any case computed since we require it to compute the lower-level gap which is the objective value of the second population (see Fig. 3). It measures how accurate is the solution provided by a generated heuristic.

In the next section, a comparison is provided between CARBON and COBRA (described in the related work section). First, we introduce all parameters and the considered instances to perform the experiments. Then, we discuss the results and explain the main differences between both bi-level co-evolutionary algorithms.

V. Experimental results

A. Setups and parameters

The lower-level covering problem embedded inside the BCPOP have non-binary matrix coefficients. Unfortunately, we did not find any existing instances for this problem. Instead of generating them from scratch, we turned our attention to the OR-library. This library provides various instances for different well-know combinatorial problems. The closest problem with such non-binary matrix coefficients and binary decision variables is the Multi-dimensional Knapsack Problem (MKP). We therefore modified the MKP instances found at the OR-library such that all $\leq$-constraints becomes $\geq$-constraints. We also ensure that each modified instance have non-empty search space. We select 9 types of instances with $n \in \{100, 250, 500\}$ decision variables and $m \in \{5, 10, 30\}$ constraints.

TABLE I: Functions and terminal sets implemented in this work

Name	Description
Operators	
+	Add two inputs
-	Substract two inputs
*****	Multiply two inputs
%	Divide two inputs with protection
mod	Modulo b.t.w. two inputs with protection
Terminal sets/ Arguments	
c_j	Cost of the current item j
q_j^k	Distribution of service k in bundle j
b^k	Required number of service k
d_k	Dual value for service k after LP relaxation
$\bar{x}_j$	Solution value for bundle j after LP relaxation

Concerning the algorithms, Table II describes all parameters used during the experiments. In order to fairly compare them, we adopted the same number of upper-level and lower-level fitness evaluation. COBRA (see Algorithm 1.) implements archives at both levels to keep track of the best results. We also adopted this strategy in CARBON. The only difference between the set of parameters of both algorithms occurs at lower-level since we employ GP operators and not GA operators. Therefore, you can observe a reproduction operator which is very typical in GP.

Algorithm 1 COBRA algorithm [32]

```
pop ← create_initial_pop()
pop_upper ← copy_upper(pop)
pop_lower ← copy_lower(pop)
while stopping criterion is not met do
    upper improvement (pop_upper) and lower improvement (pop_lower)
    upper archiving (pop_upper) and lower archiving (pop_lower)
    selection (pop_upper) and selection (pop_lower
    coevolution(pop_upper,pop_lower)
    adding from upper archive (pop_upper) and from lower archive (pop_lower)
end while
return lower archive
```

Finally, experiments for both algorithms were carried out using the HPC facility of the University of Luxembourg [44].The python library DEAP [45] has been considered for the implementation CARBON and COBRA.

B. Numerical results

After 30 runs for each algorithms on the 9 different instances, we recorded the best results in terms of %-gap (see

TABLE II: Parameters used for both Bi-level evolutionary approaches

	CARBON	COBRA
Independent runs	30	30
UL encoding	continuous values	continuous values
UL Population size	100	100
UL Archive size	100	100
UL Fitness evaluations	50000	50000
UL Selection	Binary Tournament	Binary Tournament
UL Crossover operator	Simulated binary	Simulated binary
UL Crossover probability	0.85	0.85
UL Mutation operator	Polynomial	Polynomial
UL Mutation probability	0.01	0.01
LL encoding	syntax trees	binary values
LL fitness evaluations	50000	50000
LL Archive size	100 –	100
LL Selection	Tournament	Binary Tournament
LL Crossover operator	(GP) One-point	(GA) Two-points
LL Crossover probability	0.85	0.85
LL Mutation operator	(GP) uniform	(GA) swap
LL Mutation probability	0.1	$\frac{1}{\#variables}$
LL Reproduction probability	0.05	–

TABLE III: %-gap

		%-GAP to LL optimality	
# Variables	# Constraints	CARBON	COBRA
100	5	1.13	9.71
100	10	1.87	12.33
100	30	3.13	23.31
250	5	0.37	25.19
250	10	0.76	26.08
250	30	1.62	27.75
500	5	0.15	30.07
500	10	0.34	34.68
500	30	0.74	35.19
Average		1.12	24.92

$$
\begin{aligned}
\min \quad & F(x,y) = -x - 2y \\
\text{s.t.} \quad & 2x - 3y \geq -12 \\
& x + y \leq 14 \\
& \min_{y \geq 0} \quad f(y) = -y \\
& \text{s.t.} \; -3x + y \leq -3 \\
& \qquad 3x + y \leq 30
\end{aligned}
$$

Program 3: Example from the Introduction section

Table III) and upper-level fitness value (see Table IV). In COBRA, data are extracted from the lower-level archive. In order to obtain fair comparison results, we apply the same extraction protocol for CARBON. Our first discussion will be devoted to Table III. It can be easily observed that CARBON provides better lower-level solutions than COBRA. The %-gap which measures the distance from lower-level solutions to their continuous lower bound is much smaller for CARBON. These results indicates that upper-level decision maker can forecast more accurately the lower-level rational reaction of the lower-level decision maker. We recall that CARBON directly minimize the gap while COBRA minimize the lower-level objective value. In a single-level optimization problem, both would achieve the same results since we are getting closer to the optimal solution when we consider either of these objectives. However, in Bi-level Optimization, each upper-level decision x leads to a different lower-level instances with a different lower-level optimal solution. The disadvantages of COBRA is a work-flow which does not take into account that many different lower-level instances stem from different upper-level decision. Let us take the example provided in the Introduction section and recall hereafter as Program. 3. If we set $x = 2$ then the resulting lower-level instance will be $min\{f(y) = -y | y \leq 3\}$ with optimal $\hat{y} = 3$. Setting $x = 6$ leads to the lower-level instance $min\{f(y) = -y | y \leq 12\}$ with optimal $\hat{y} = 12$. On the contrary, CARBON does not rely on any lower-level solution value but on its gap. Therefore, whatever upper-level decision you are considering, CARBON can evaluate how accurate is a lower-level solution.

Let us focus now on the upper-level fitness values reported in Table IV. A first observation let us believe that the upper-level objective value is much better for COBRA than CARBON. Indeed, the upper-level fitness value is higher for each instance. In the context of the BCPOP, it means that the pricing obtained with COBRA lead to a better payoff than for CARBON. In fact, this is completely wrong. We cannot observe the upper-level fitness value without the gap values. COBRA led to larger gaps than CARBON. This means that COBRA has less accurate lower-level solutions than CARBON.

To prove it, let us return to the general bi-level mathematical program illustrated by Program. 1. It can be written: $min\{F(x,y) | G(x,y) \leq 0, \quad f(x,y) \leq w(x)\}$ where $min\{f(x,y) | g(x,y) \leq 0\}$ with $x \geq 0$ and $y \geq 0$. $w(x)$ is the optimal lower-level solution value for the the upper-level decision x. Let suppose with know to algorithms H_1 and H_2 such that $w(x) \leq H_1(x) \leq H_2(x)$. Considering the following sets:

$$
\begin{aligned}
S_{opt} &= \{G(x,y) \leq 0, \quad f(x,y) \leq w(x)\} \\
S_{H_1} &= \{G(x,y) \leq 0, \quad f(x,y) \leq H_1(x)\} \\
S_{H_2} &= \{G(x,y) \leq 0, \quad f(x,y) \leq H_2(x)\}
\end{aligned} \tag{2}
$$

We can easily affirm that $S_{opt} \subset S_{H_1} \subset S_{H_2}$. Consequently, $min\{F(x,y) | (x,y) \in S_{opt}\} \geq min\{F(x,y) | (x,y) \leq S_{H_1})\} \geq min\{F(x,y) | (x,y) \in S_{H_2})\}$. Indeed, an approximation of the lower-level makes the upper-level less constrained leading to a relaxation, i.e., an overestimation of the payoff for the upper-level decision maker. If we now focus

on the BCPOP and the results obtained by CARBON and COBRA. We obtained statistically that

$$\frac{w(x) - LB(x)}{LB(x)} \leq \frac{A_{carbon}(x) - LB(x)}{LB(x)} \leq \frac{A_{cobra}(x) - LB(x)}{LB(x)}$$
$$\Leftrightarrow w(x) \leq A_{carbon}(x) \leq A_{cobra}(x) \tag{3}$$

where $A_{carbon}(x)$ and $A_{cobra}(x)$ are respectively the lower-level objective values for the upper-level decision x for CARBON and COBRA. Finally, we can conclude that $S_{opt} \subset S_{carbon} \subset S_{cobra}$ and therefore $max\{F(x,y)|(x,y) \in S_{opt}\} \leq max\{F(x,y)|(x,y) \leq S_{carbon})\} \leq max\{F(x,y)|(x,y) \in S_{cobra})\}$. We recall that the Program. 2 is maximized at upper-level.

In conclusion, the approximation done at lower-level lead to an upper-level relaxation since the upper-level is less constrained. Here, the results obtained in Table IV implies that CARBON provides tighter upper-bounds than COBRA since the upper-level problem is a maximization problem.

TABLE IV: UL objective values

		UL objective value	
# Variables	# Constraints	CARBON	COBRA
100	5	10964.07	14710.78
100	10	8976.39	15226.79
100	30	8669.49	14762.83
250	5	25750.66	35479.64
250	10	26897.33	38283.71
250	30	24338.39	39368.26
500	5	50177.28	73529.34
500	10	49441.39	75041.02
500	30	48904.15	75386.02
Average		28235.46	42420.93

Finally to conclude this section, we choose to discuss two average convergence curves for the instance with $n = 500$ variables and $m = 30$ constraints. The first one depicted in Fig.4 represents the average convergence over the 30 runs obtained with CARBON. Fig.5 illustrates the ones obtained with COBRA. We can easily observe that the convergence curves are smoother for CARBON with a steady increase for the upper-level fitness and a steady decrease for the gap. On the contrary, we can note that both convergence curves have a see-saw shape which indicates us that each improvements phase (see Algorithm 1 for the pseudo-code) deteriorates the other level. In conclusion, COBRA still seems very close to a nested meta-heuristics despite the use of a co-evolutionary operator. The fact that it relies on independent improvement phases do not facilitate its parameterization. Indeed, how should be set the number of improvement generation for each level? Should it be unbalanced? Unlike COBRA, we see that CARBON is able to break the nested structure. The convergence curves clearly show that both populations have steady improvements contrary to COBRA.

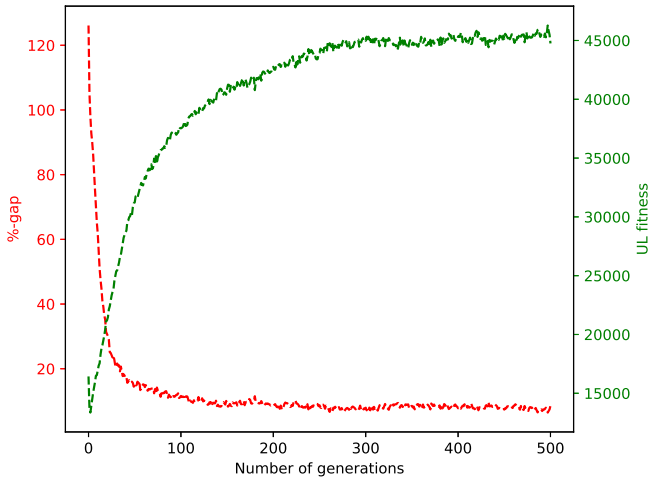

Fig. 4: Convergence curve for both CARBON populations

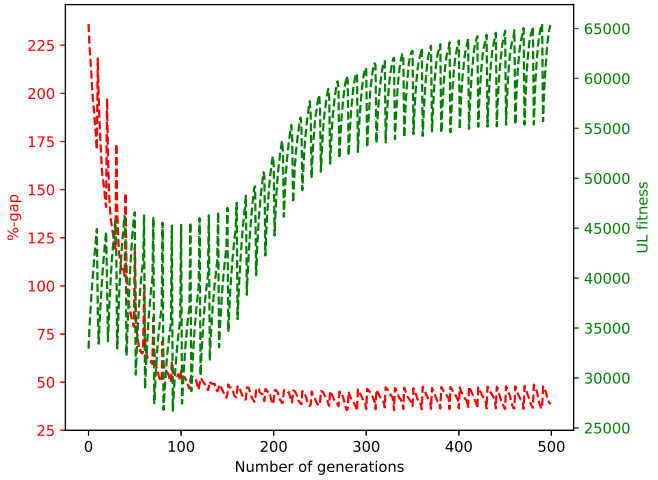

Fig. 5: Convergence curve for both COBRA populations

VI. Conclusion

In this work, we introduced CARBON, a novel competitive co-evolutionary algorithm to solve bi-level optimization problems. We discussed the main difficulty to create a co-evolutionary algorithm when dealing which two nested optimization problems. Instead of considering, two strongly dependent populations, we make use of GP hyper-heuristic to generate and learn automatically efficient heuristics to solve the parametric lower-level problem. Indeed, each upper-level decision induces a new lower-level instance that need to be solved. Heuristics are fast but less accurate than standard meta-heuristics. Therefore, using the evolutionary paradigm, it is possible to evolve those heuristics to make them more accurate. The proposed hybrid bi-level co-evolutionary algorithm, i.e., CARBON, pairs the bests upper-level decision with the best heuristics to provide a very good approximation of the lower-level rational reactions and thus a realistic payoff for the upper-level decision maker. Numerical experiments on the

Bi-level Cloud Pricing Optimization Problem have shown that CARBON can better handle the nested structure than COBRA, a reference in terms of bi-level co-evolutionary algorithm. Future works will be devoted to multiple-level problems with deeper nested structure in order to analyze the limitations of CARBON in terms of co-evolution.

REFERENCES

[1] H. Stackelberg, *The theory of the market economy*, 1952.

[2] J. Bracken and J. McGill, "Mathematical programs with optimization problems in the constraints," vol. 21, pp. 37–44, 1973.

[3] A. G. Mersha and S. Dempe, "Linear bilevel programming with upper level constraints depending on the lower level solution," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 247 – 254, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300305011197

[4] J. Bracken and J. McGill, "Defense applications of mathematical programs with optimization problems in the constraints," vol. 22, pp. 1086–1096, 1974.

[5] A. Migdalas, "Bilevel programming in traffic planning: Models, methods and challenge," *Journal of Global Optimization*, vol. 7, no. 4, pp. 381–405, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF01099649

[6] W.Candler, J.Fortuny-Amat, and B.McCarl, "The potential role of multilevel programming in agricultural economics," *American Journal of Agricultural Economics*, vol. 63, no. 3, pp. 521–531, 1981.

[7] J. Bard, "Coordination of a multidivisional organization through two levels of management," vol. 11, pp. 457–468, 1983.

[8] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard, "A bilevel model for toll optimization on a multicommodity transportation network," *Transportation Science*, vol. 35, no. 4, pp. 345–358, 2001. [Online]. Available: http://dx.doi.org/10.1287/trsc.35.4.345.10433

[9] J. Gonzalez-Velarde, J.-F. Camacho-Vallejo, and G. Pinto, "A scatter search algorithm for solving a bilevel optimization model for determining highway tolls," vol. 19, pp. 5–16, 04 2015.

[10] V. V. Kalashnikov, R. C. H. Maldonado, J.-F. Camacho-Vallejo, and N. I. Kalashnykova, "A heuristic algorithm solving bilevel toll optimization problems," *The International Journal of Logistics Management*, vol. 27, no. 1, pp. 31–51, 2016.

[11] G. Whittaker, R. Fre, S. Grosskopf, B. Barnhart, M. Bostian, G. Mueller-Warrant, and S. Griffith, "Spatial targeting of agri-environmental policy using bilevel evolutionary optimization," *Omega*, vol. 66, pp. 15 – 27, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048316000086

[12] M. Bostian, G. Whittaker, B. Barnhart, R. Fre, and S. Grosskopf, "Valuing water quality tradeoffs at different spatial scales: An integrated approach using bilevel optimization," *Water Resources and Economics*, vol. 11, pp. 1 – 12, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212428415300025

[13] A. U. Raghunathan and L. T. Biegler, "Mathematical programs with equilibrium constraints (mpecs) in process engineering," *Computers & Chemical Engineering*, vol. 27, no. 10, pp. 1381 – 1392, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135403000929

[14] W. Halter and S. Mostaghim, "Bilevel optimization of multi-component chemical systems using particle swarm optimization," in *2006 IEEE International Conference on Evolutionary Computation*, 2006, pp. 1240–1247.

[15] J. Herskovits, A. Leontiev, G. Dias, and G. Santos, "Contact shape optimization: a bilevel programming approach," *Structural and Multidisciplinary Optimization*, vol. 20, no. 3, pp. 214–221, Nov 2000. [Online]. Available: https://doi.org/10.1007/s001580050149

[16] V. Suryan, A. Sinha, P. Malo, and K. Deb, "Handling inverse optimal control problems using evolutionary bilevel optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 1893–1900.

[17] M. Johnson, N. Aghasadeghi, and T. Bretl, "Inverse optimal control for deterministic continuous-time nonlinear systems," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 2906–2913.

[18] J. Z. Liang and R. Miikkulainen, "Evolutionary bilevel optimization for complex control tasks," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 871–878. [Online]. Available: http://doi.acm.org/10.1145/2739480.2754732

[19] A. Sinha, P. Malo, P. Xu, and K. Deb, "A bilevel optimization approach to automated parameter tuning," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: ACM, 2014, pp. 847–854. [Online]. Available: http://doi.acm.org/10.1145/2576768.2598221

[20] J. Bard, "Some properties of the bilevel programming problem," vol. 68, pp. 371–378, 1991, technical Note.

[21] J. Bard and J. Falk, "An explicit solution to the multi-level programming problem," vol. 9, pp. 77–100, 1982.

[22] G. Savard and J. Gauvin, "The steepest descent direction for the nonlinear bilevel programming problem," vol. 15, pp. 275–282, 1994.

[23] P. Fontaine and S. Minner, "Benders decomposition for discretecontinuous linear bilevel problems with application to traffic network design," *Transportation Research Part B: Methodological*, vol. 70, no. 0, pp. 163 – 172, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261514001611

[24] M.Köppe, M. Queyranne, and C. T. Ryan, "A parametric integer programming algorithm for bilevel mixed integer programs," *ArXiv e-prints*, Jul. 2009.

[25] B. Colson, P. Marcotte, and G. Savard, "A trust-region method for nonlinear bilevel programming: Algorithm and computational experience," *Computational Optimization and Applications*, vol. 30, no. 3, pp. 211–227, Mar 2005. [Online]. Available: https://doi.org/10.1007/s10589-005-4612-4

[26] A. Koh, "Solving transportation bi-level programs with differential evolution," in *2007 IEEE Congress on Evolutionary Computation*, Sept 2007, pp. 2243–2250.

[27] X. Li, P. Tian, and X. Min.

[28] K. Sahin and A. Ciric, "A dual temperature simulated annealing approach for solving bilevel programming problems," *Computers and Chemical Engineering*, vol. 23, no. 1, pp. 11 – 25, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0098135498002671

[29] J. Fliege and L. Vicente, "Multicriteria approach to bilevel optimization," *Journal of Optimization Theory and Applications*, vol. 131, no. 2, pp. 209–225, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10957-006-9136-2

[30] A. Sinha, P. Malo, and K. Deb, "An improved bilevel evolutionary algorithm based on quadratic approximations," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1870–1877.

[31] E. Kieffer, G. Danoy, P. Bouvry, and A. Nagih, "Bayesian optimization approach of general bi-level problems," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 1614–1621. [Online]. Available: http://doi.acm.org/10.1145/3067695.3082537

[32] F. Legillon, A. Liefooghe, and E. Talbi, "Cobra: A cooperative coevolutionary algorithm for bi-level optimization," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, June 2012, pp. 1–8.

[33] V. Oduguwa and R. Roy, "Bi-level optimisation using genetic algorithm," in *Artificial Intelligence Systems, 2002. (ICAIS 2002). 2002 IEEE International Conference on*, 2002, pp. 322–327.

[34] A. Chaabani, S. Bechikh, and L. B. Said, "A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 1659–1666.

[35] W. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 228 – 234, 1990.

[36] 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.

[37] E. Kieffer, G. Danoy, P. Bouvry, and A. Nagih, "A new co-evolutionary algorithm based on constraint decomposition," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2017, pp. 492–500.

[38] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring Hyper-heuristic Methodologies

with Genetic Programming," *Computational Intelligence*, vol. 1, pp. 177–201, 2009. [Online]. Available: http://www.cs.nott.ac.uk/{~}gxo/papers/ChapterGPasHH09.pdf

[39] E. K. Burke, M. R. Hyde, and G. Kendall, "Grammatical evolution of local search heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 406–417, 2012.

[40] M. Oltean, "Evolving evolutionary algorithms using linear genetic programming," *Evol. Comput.*, vol. 13, no. 3, pp. 387–410, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1162/1063656054794815

[41] A. Elyasaf, A. Hauptman, and M. Sipper, "Evolutionary design of freecell solvers," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 4, pp. 270–281, Dec 2012.

[42] R. R. van Lon, T. Holvoet, G. Vanden Berghe, T. Wenseleers, and J. Branke, "Evolutionary synthesis of multi-agent systems for dynamic dial-a-ride problems," in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '12. New York, NY, USA: ACM, 2012, pp. 331–336. [Online]. Available: http://doi.acm.org/10.1145/2330784.2330832

[43] S. Nguyen, M. Zhang, and M. Johnston, "A genetic programming based hyper-heuristic approach for combinatorial optimisation," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 1299–1306. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001752

[44] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos, "Management of an academic hpc cluster: The ul experience," in *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*, Bologna, Italy, July 2014, pp. 959–967.

[45] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.