Chapter 4
CoBRA: A Coevolutionary Metaheuristic for Bi-level Optimization

François Legillon, Arnaud Liefooghe, and El-Ghazali Talbi

Abstract. This article presents CoBRA, a new parallel coevolutionary algorithm for bi-level optimization. CoBRA is based on a coevolutionary scheme to solve bi-level optimization problems. It handles population-based meta-heuristics on each level, each one cooperating with the other to provide solutions for the overall problem. Moreover, in order to evaluate the relevance of CoBRA against more classical approaches, a new performance assessment methodology, based on rationality, is introduced. An experimental analysis is conducted on a bi-level distribution planning problem, where multiple manufacturing plants deliver items to depots, and where a distribution company controls several depots and distributes items from depots to retailers. The experimental results reveal significant enhancements with respect to a more classical approach, based on a hierarchical scheme.

4.1 Introduction

Bi-level optimization problems allow to model a large number of real-life applications, with a hierarchical structure between two decision makers. It includes companies which have to face a legislator and security constraints [10], companies trying to predict consumer reaction [8], or a supply chain where a company has to predict its supplier reaction to determine the real cost of its decision [3].

Metaheuristics are a class of approximate algorithms focusing on finding good-quality solutions for large-size and complex problems, in a reasonable time [20]. While most of the existing literature about bi-level optimization focuses on

François Legillon
Tasker and INRIA Lille Nord Europe, France
e-mail: francois.legillon@inria.fr

Arnaud Liefooghe · El-Ghazali Talbi
University of Lille 1, CNRS, INRIA, France
e-mail: {arnaud.liefooghe,talbi}@lifl.fr

E.-G. Talbi (Ed.): *Metaheuristics for Bi-level Optimization*, SCI 482, pp. 95–114.
DOI: 10.1007/978-3-642-37838-6_4

small-size linear problems (see for example [1, 9]), many real-life applications involve large-size instances and complex NP-hard problems, justifying the use of meta-heuristics. Meta-heuristics for bi-level optimization can be divided in two main classes. On the one hand, *hierarchical algorithms* try to solve the two levels sequentially, improving solutions on each level to get a good overall solution on both levels. Such algorithms include the repairing algorithm [12] , which considers the lower-level problem as a constraint and solve it during the evaluation step, or the constructing algorithm [13] which applies two improving algorithms on a population, one for each level, sequentially until meeting a stopping criterion. On the other hand, *coevolutionary algorithms* maintain two populations, one for each level, and try to improve it separately, while exchanging periodically information to keep an overall view on the problem, like in [16]. In cooperative coevolution, different sub-populations evolve a part of the decision variables, and complete solutions are built by means of a cooperative exchange of individuals from sub-populations [18].

This article focuses on a coevolutionary approach. Sub-problems involved in bi-level optimization can be tackled by meta-heuristics. Finding a good way to combine two meta-heuristics in order to solve a bi-level optimization problem would give a general methodology for bi-level optimization. First, we introduce a new algorithm, the *Co*evolutionary *B*i-level method using *R*epeated *A*lgorithms (CoBRA) . This coevolutionary meta-heuristic is able to face general bi-level optimization problems, possibly involving complex large-size problems. Next, we introduce a new method for performance assessment, the rationality, able to more fully grasp the bi-level aspect of the problems than the Pareto efficiency. Rationality is based on the proximity from the optimum of the lower-level variables with the corresponding upper-level variables fixed. At last, to evaluate the performance of CoBRA against classical hierarchical approaches, we give an experimental analysis on a bi-level transportation problem involving a supply chain, the bi-level multiple depot vehicle problem introduced in [3]. This analysis includes the modeling of the problem, the instantiation of CoBRA on it and the study of the results with respect to the rationality metrics.

The paper is organized as follows. Section 4.2 gives the necessary background on bi-level optimization. Section 4.3 presents the new coevolutionary algorithm proposed in the paper for bi-level optimization, namely CoBRA. In Section 4.4, we discuss the issue of assessing the performance of approximate algorithms in bi-level optimization. The bi-level transportation problem under investigation in this paper is presented in Section 4.5, both in a single-objective and a multi-objective formulation. The experimental analysis of CoBRA is given Section 4.6. At last, the final section concludes the paper and gives directions for further research.

4.2 Bi-level Optimization

In this section we introduce a general bi-level optimization problem, and give a quick overview of state-of-the-art meta-heuristics for bi-level optimization.

4.2.1 General Principles of Bi-level Optimization

Bi-level optimization problems may be defined by the tuple (S,F,f) where S represents the set of feasible solutions, F the objective function(s) of the upper-level, and f the objective function(s) of the lower-level. For any $x \in S$ we separate the upper-level variables and the lower-level variables, respectively in x_u and x_l.

We define, for every x_u fixed, the set of rational reactions $R(x_u)$ as the set of x_l optimal in f.

$$R(S,f,x_u) = \begin{cases} \min_{x_l} f(x = (x_u,x_l)) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ \text{s.t. } x \in S \end{cases}$$

The bi-level problem consist in finding the solution $x \in S$ which is optimal with respect to f for x_u fixed and, respecting this constraint, optimal in F.

$$BP(\mathscr{S},\mathrm{F},\mathrm{f}) = \begin{cases} \min \mathrm{F}(x) \\ x \in \mathscr{S} \\ \text{s.t.} \begin{cases} x = (x_u,x_l) \\ x_l \in R(\mathscr{S},\mathrm{f},x_u) \end{cases} \end{cases}$$

Those problems induce a hierarchy between two decision makers:

- The *leader*, who chooses the upper part of the decision variables, x_u, and who tries to optimize $F(x)$.
- The *follower*, who chooses the lower part of the decision variables, x_l, and who tries to optimize $f(x)$.

The leader decides first. Then, the follower, knowing the leader decision, has to decide, in the view of optimizing its own objective function(s) f, without regarding the upper objective function(s) F. To optimize his choice, the leader then has to predict the follower *reaction*. This hierarchy can conduct to a higher complexity than both sub-problems. For instance, a NP-hard problem can be obtained from two linear problems [2].

This definition of bi-level optimization corresponds to the optimistic case, where the leader can "choose" the (x_u,x_l) couple in the set of $(x_u,x_l) \in S$ where $x_l \in R(x_u)$: the reaction has to be optimal, but if several reactions are optima (*i.e.* $|R(x_u)| > 1$) the leader has the last word . There exists a pessimistic case [14] which is not treated in this paper, where x_l is chosen as the leader worst case scenario in the set of rational responses.

4.2.2 Meta-heuristic Approaches for Bi-level Optimization

Meta-heuristics are approximate algorithms which allow to tackle large-size problem instances by delivering satisfactory solutions in reasonable time [20]. Due to

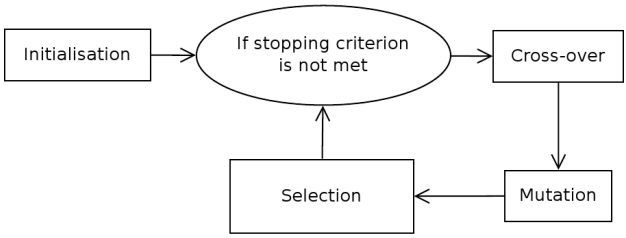

Fig. 4.1 General scheme of an evolutionary algorithm

their complexity, most bi-level optimization problems are tackled by approaches which involve a model reformulation masking the bi-level aspect of the problem (see [1, 9, 11, 15]), or involve meta-heuristics. Evolutionary algorithms are meta-heuristics mimicking the species evolution. We will use in this article several terms related to evolutionary algorithms: an *individual* is a feasible solution, a *population* is a set of individuals, a *mutation* is the creation of a new individual from an existing one, generally keeping some properties. A cross-over is the creation of individual(s), called *offspring*, from several other individuals called *parents*. The process of applying cross-over and mutation operators to a population in order to create a new population is called *generation*. On each generation, a *selection* step consists in selecting individuals to meet defined goals. Evolutionary algorithms consist in creating multiple generations and applying selections until a stopping criterion is met (Fig. 4.1). The reader is referred to [20] for more details about population-based meta-heuristics and evolutionary algorithms.

In this paper, we focus on coevolutionary approaches, a sub-group of meta-heuristics extending the evolutionary scheme. Coevolutionary algorithms consists in associating several evolutionary algorithms and applying transformations, such as mutation and cross-over, to distinct populations. A coevolution operator is then regularly applied between sub-populations to keep a global view on the whole problem. Oduguwa and Roy described BiGA [16], a coevolutionary algorithm to solve bi-level problems.

BiGA starts by initializing two distinct sub-populations using a heuristic, pop_u for the upper level and pop_l for the lower, then the upper part of the solutions is copied from pop_u to pop_l. Then during a parametrized number of generations, a selection process based on the respective level fitness values is applied on both sub-populations, followed by a mutation/crossover step. Then the sub-populations are evaluated, sorted, and coevolved, by copying the upper (resp. lower) variables to the lower (resp. upper) sub-population. At last, an archiving process occurs, before looping again to the selection step. The pseudo-code of BiGA is given in Algorithm 5.

Algorithm 5: BiGA

Data: initial population pop
$pop_l \leftarrow \text{selection}_{lower}(pop)$;
$pop_u \leftarrow \text{selection}_{upper}(pop)$;
Coevolution(pop_u, pop_l);
while *Stopping criterion not met* **do**
  Crossover(pop_u), crossover(pop_l);
  Mutation(pop_u), mutation(pop_l);
  Evaluation(pop_u), evaluation (pop_l);
  Elitist coevolution (pop_u, pop_l);
  Evaluation(pop_u), evaluation (pop_l);
  Archiving(pop_u), archiving (pop_l);
end
return *archive*

4.3 CoBRA, a Coevolutionary Meta-heuristic for Bi-level Optimization

In this section we introduce CoBRA, a new meta-heuristic to tackle bi-level problems.

4.3.1 General Principles

Most of literature works focus on linear bi-level problems (*ie*: formed with two linear sub-problems) or lower-level problems solvable in a reasonable amount of time. They use this property to discard the bi-level aspect of the problem. This article tries to define a more general methodology to solve bi-level optimization problems. The complexity of the considered problems lead us to consider the use of meta-heuristic, to obtain good-quality solutions in a reasonable amount of time.

We introduce a meta-heuristic, CoBRA, a *co*evolutionary *b*ilevel method using *r*epeated *a*lgorithms. Extending Oduguwa and Roy's BiGA [16], it is a co-evolutionary meta-heuristic consisting in improving incrementally two different sub-populations, each one corresponding to one level, and periodically exchanging information with the other.

4.3.2 CoBRA Components

In order to instantiate CoBRA to solve a bi-level optimization problem, generic and problem-specific components have to be defined. Generic components, which can correspond to both sub-problems, consist in choosing the following:

- An improvement algorithm for each level, to improve the solutions on its level. We use, for single-objective levels, a classic evolutionary algorithm, and, for

multi-criterion levels, NSGA-II. Those algorithms are classic population-based meta-heuristic approaches [20].
- A coevolution strategy to decide how populations should exchange information.
- An archiving strategy to record the best solutions on every level, and to prevent the coevolution to change completely the sub-populations on a single generation.
- A stopping criterion to decide when the algorithm should stop.

Problem specific components still have to be designed to use CoBRA:

- Initialization operators, generally heuristics, which create a base population to begin the search process.
- Variation operators, level-specifics, which are then used by the improvements algorithms.
- Evaluation operators, corresponding to the f and F functions from the bi-level optimization model.

Figure 4.2 illustrates the outline of CoBRA.

4.3.3 General Algorithm

CoBRA is a coevolutionary algorithm using for each level a different population, and a different archive (Algo. 9.4). At each iteration, we apply the improvement algorithms, we archive the best solutions obtained, then we apply a selection operator to keep a constant size to the archive and to the populations. The final iteration step is then to coevolve the two sub-populations. Once the stopping criterion is met, CoBRA returns the lower-level archive.

Extending the BiGA approach, CoBRA involves several differences from the former:

1. The main difference is that CoBRA applies a complete algorithm, possibly iterating a certain number of generations, over each main algorithm iteration, instead of just applying variation operators. Evaluation process occurs during those improvement algorithms.

Algorithm 6: CoBRA

Data: initial population pop
$pop_u \leftarrow_{copie}$ pop;
$pop_l \leftarrow_{copie}$ pop;
while *Stopping criterion not met* **do**
  upper improvement (pop_u) and lower improvement (pop_l);
  upper archiving (pop_u) and lower archiving (pop_l);
  selection (pop_u) and selection (pop_l);
  coevolution(pop_u, pop_l);
  adding from upper archive (pop_u) and from lower archive (pop_l);
end
return *lower archive*

2. The coevolution process is not necessarily elitist: default coevolution strategy (Algo. 7) randomly coevolves solutions with each other.
3. The selection operations and the archives take place right after the improvement.

Algorithm 7: Random coevolution

Data: Populations *upPop* and *lowPop* of same size, *op* coevolution operator
Shuffle *upPop*;
foreach *i from 0 to size(upPop)* **do**
  op(*upPop*[*i*],*lowPop*[*i*]);

4.4 Performance Assessment and Bi-level Optimization

In this section, we introduce two new metrics for assessing the performance of heuristics on solving bi-level optimization problems.

4.4.1 Motivations

Being a problem with two different objective functions, a natural approach to tackle bi-level optimization problems would be to use a Pareto-based multi-objective approach . However bi-level optimization problems have a different structure. A good solution considering a similar problem approximating the Pareto frontier could be of bad quality in the bi-level way.

Bi-level optimization aim at identifying solutions in the form (x_u, x_l) which give good upper objective vectors, while being near the optimum regarding the lower objective for x_u fixed. This leads to the existence of good quality solutions not being on the Pareto frontier, and solutions on the Pareto frontier not necessarily being good quality solutions. Fig. 4.3 gives an example of objective functions giving a bi-level solution corresponding to a dominated solution in the Pareto sense. F and f are respectively the upper and the lower-level objective functions to be minimized, the leader chooses in $\{$d,e,f$\}$ and the follower in $\{$a,b$\}$. The Pareto front would be composed of $\{$(d,a),(f,a)$\}$ while the bi-level solution is (e,a).

We introduce the notion of *rationality* which correspond to the difficulty to improve a solution (x_u, x_l), with x_u fixed, according to the lower-level objective function. A rational solution is a solution where the follower reaction is rational, seeking for the optimality of its own objective function(s). We introduce two different rationality metrics, the direct one and the weighted one.

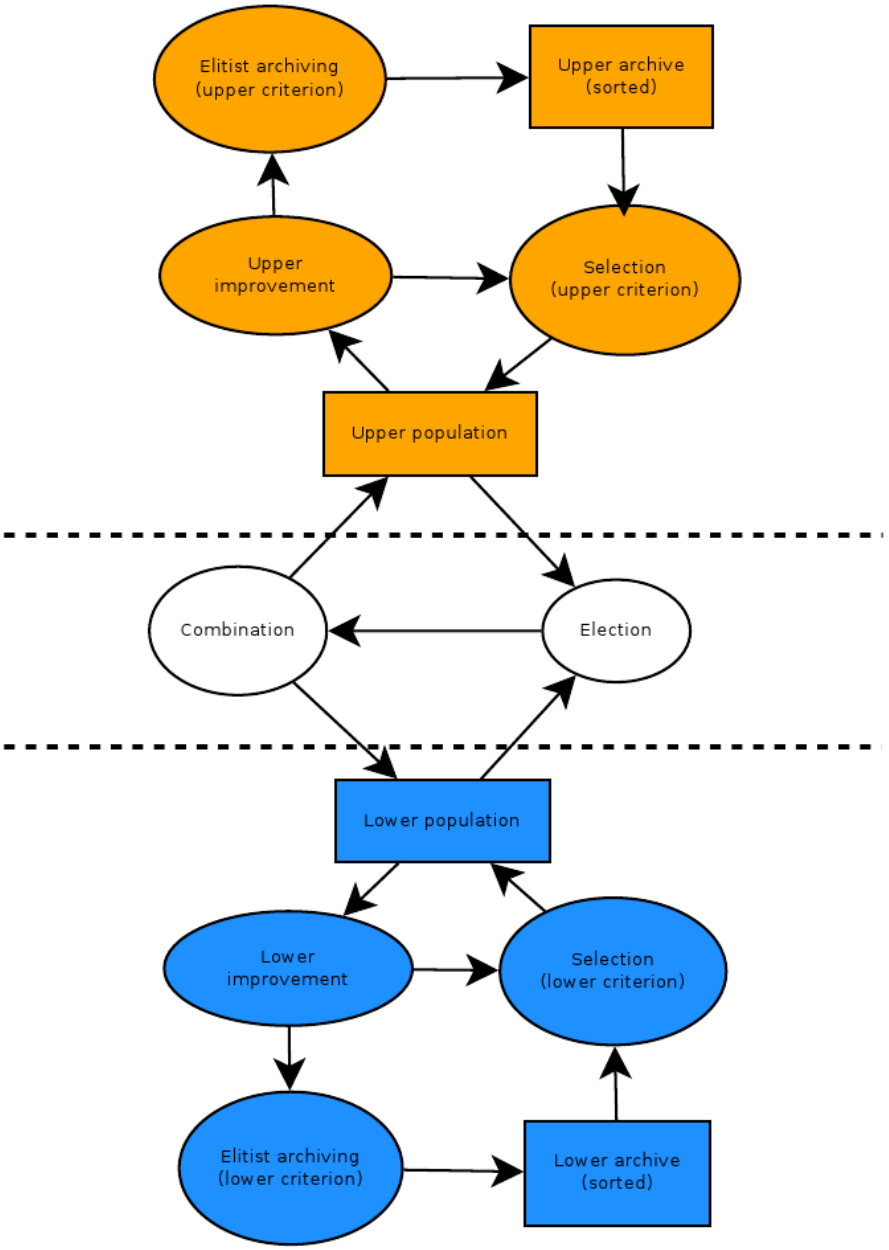

Fig. 4.2 CoBRA outline

F	a	b
d	0	1000
e	1	∞
f	300	∞

f	a	b
d	100	99
e	1001	∞
f	99	∞

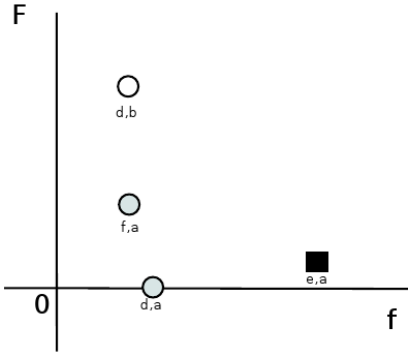

Fig. 4.3 Example of lower-level and upper-level objective functions whose optimal solution is dominated in terms of Pareto dominance

4.4.2 Rationality

4.4.2.1 Direct Rationality

The *direct rationality* measure corresponds to the difficulty of improving a solution without regarding the actual improvement: we simply consider the "improvability". To evaluate it for a population, we apply a parametrized number of time a "good" lower-level algorithm, and count how many times the algorithm did improve the solution (Algo. 8).

4.4.2.2 Weighted Rationality

The *weighted rationality* is another rationality measure working on the same principle as the direct rationality with the difference that, instead of counting how many times the algorithm was able to improve the solution, we also consider how much it was improved. Being able to improve a fitness by 0.001 or by 1000 does not give the same result to the rationality, whereas the direct approach would consider both as the same (Algo. 9). For bi-level optimization problems involving a multi-objective lower-level sub-problem, we used the multiplicative ε-indicator, an indicator to compare sets of objective vectors [21].

4.4.3 Discussion

The weighted rationality metric was introduced to compare results for a bi-level optimization problem composed with a hard lower-level problem. All the tested algorithms giving a bad direct rationality, we noticed that some algorithms were still doing better and were far nearer to the optimal on the lower-level than others. The weighted rationality is able to differentiate such algorithms.

Algorithm 8: Direct rationality test

```
Data: AlgoLow, pop, ni number of iterations
counter ← 0;
foreach gen from 1 to ni do
    neopop ← pop;
    found ← false;
    AlgoLow(neopop);
    foreach x in neopop do
        if (not found) and (x dominates an element of pop) then
            counter++;
            found ← false;
        end
    end
end
return counter/ni
```

Algorithm 9: Weighted rationality test

```
Data: AlgoLow, pop, ni number of iterations
ratio ← 0;
foreach gen from 1 to ni do
    neopop ← pop;
    AlgoLow(neopop);
    ratio=ratio+ε_ind(pop,neopop)/ni;
end
return ratio
```

We can note that those methods are not absolute, in the sense that we have to compare the algorithm using another algorithm, thus introducing a bias. Those measures compare the capacity of a meta-heuristic to use improvement algorithms, but do not actually compare the overall capacity to tackle the problem. To this end, we have to ensure that none of the tested algorithms is biased toward the improvement used by the rationality evaluation.

4.5 Application to Bi-level Transportation

In this section we define a bi-level transportation problem , involving two different companies in a supply chain: the leader transports goods from depots to retailers answering to the retailers demand, and a follower manages plants producing goods for the leader. The leader starts by deciding which depots should deliver goods, then the follower decides how to manufacture the goods, both decisions influencing the overall cost of solutions. Two variants of this problem are here considered, a single-objective one, and a multi-objective one.

4.5.1 A Bi-level Multi-depot Vehicle Routing Problem

The first problem, introduced by Calvete and Galé [3], consists of a bi-level problem where the leader controls a fleet of vehicles to deliver items from several depots to retailers, on the same principle as the classical multi-depot vehicle routing problem (MDVRP) . The follower controls a set of plants, and has to produce the items and deliver them to the depots according to the demand of the retailers it serves, thus answering a flow problem. The leader tries to minimize the total distance of his routes and the buying cost of the resources (depending on the lower-level decision). The follower minimizes the production cost and the distance traveled by the produced goods. The follower has to directly transport from plants to depots.

4.5.1.1 Problem Description

Let K, L, R and S denote the sets of plants, of depots, of retailers and of vehicles, respectively. Let E be the edge set between retailers and depots, b_r the demand of retailer r, $c^a_{i,j}$ the cost of transporting from depots or retailers i to j for the leader, $c^b_{k,l}$ the cost to buy and unload a unit produced in plant k into depot l for the leader, and $c^c_{k,l}$ the operational cost for plant k to produce and deliver to depot l for the follower.

The upper objective function is to minimize the sum of deliver costs from depots to retailers and buying from plants .

$$F(x,y) = \sum_{s \in S} \sum_{(i,j) \in E} c^a_{i,j} x^s_{i,j} + \sum_{k \in K} \sum_{l \in L} c^b_{k,l} y_{k,l}$$

with x the leader variables representing the routes chosen to deliver retailers, and y the follower variables representing the affectation of plants to depots. Then, the lower-level objective function is to minimize the sum of costs of producing items in plants and delivering it to depots.

$$f(x,y) = \sum_{k \in K} \sum_{l \in L} c^c_{k,l} y_{k,l}$$

The leader and follower follow a hierarchical order, where the leader choose routes, creating a demand for the depots corresponding to the retailers to be delivered, and where the follower has to respond to this new demand by associating a part of his plant production to depots.

$$\sum_{k \in K} y_{k,l} \geq \sum_{s \in S_l} \sum_{r \in R_s} b_r, \forall l \in L$$

Several other VRP-related constraints are omitted to improve readability. See [3] for more details about the problem.

4.5.1.2 Solution Representation

In the optic of doing an evolutionary algorithm, a solution representation was necessary. Using a generic bi-level representation, we had to decide a representation for each level. For the upper-level, we use a permutation: every retailer and every route (each route being associated to a depot) has an attributed number.

The route numbers in the permutation determine the routes start, and every retailers represent in order the actual route (Fig. 4.4). This representation facilitate the solution integrity, and suppress the need to check the number of routes and the "one visit per retailer" constraint. We use for the lower-level problem a more classical double matrix M, M_b^a representing the ratio of production sent from a to b. The quantity effectively sent is scaled down at the evaluation step if the sum of a column are over 1, and rounded down if not integer. This indirect representation permits to use classical algorithms without much adaptation work.

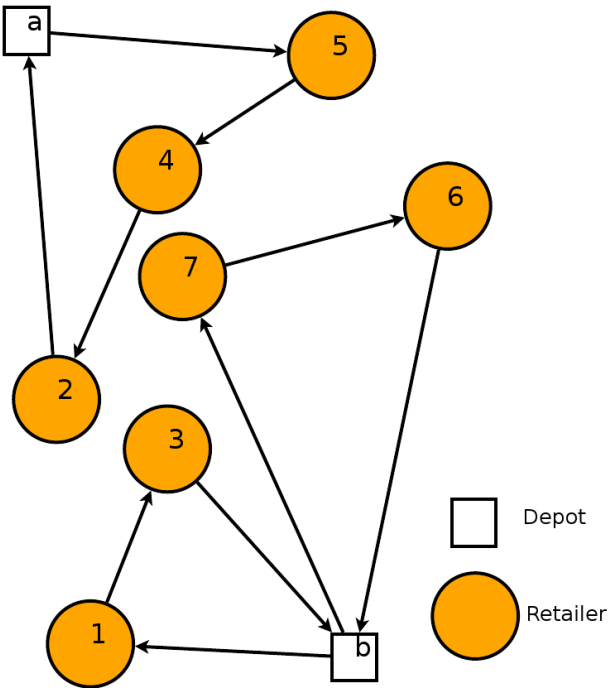

Fig. 4.4 Example of a VRP with 7 retailers, 2 depots, and 2 routes *per* depot, from the permutation $[5,4,2,9,7,6,10,1,3,8]$. Squares are for depots {a,b}, circles for retailers {1,2,3,4,5,6,7}.

4.5.1.3 Problem Instances

Two sets of instances[1] were generated to experiment the CoBRA efficiency. S_1 consist of instances created from MDVRP instances following the *modus operandi* described in [3]. We add as many plants as there are depots randomly located on the map. Then we set their maximal production to ensure that the instance is feasible. c^b and c^c follows a method described in [3]. Set S_1 contains 10 instances created from the 10 instances provided by Cordeau [4]. The second set S_2 consists of the same instances in which a higher fixed number of plants of 50 was added. Those instance parameters are described in Table 4.1.

Table 4.1 Description of S_1 and S_2 instances, R corresponding to the number of routes by depot

Instance	Depot	R	Plants (S_1)	Plants (S_2)	Retailer
bipr01	4	1	4	50	48
bipr02	4	2	4	50	96
bipr03	4	3	4	50	144
bipr04	4	4	4	50	192
bipr05	4	5	4	50	340
bipr06	4	6	4	50	288
bipr07	6	1	6	50	72
bipr08	6	2	6	50	144
bipr09	6	3	6	50	216
bipr10	6	4	6	50	288

4.5.2 A Multi-objective Bi-level Multi-depot Vehicle Routing Problem

The multi-objective bi-level multi-depot routing problem (M-BiMDVRP) is a variant of the BiMDVRP where the follower minimizes two costs instead of just one distance between plants and depots, aiming at finding a Pareto front approximation. The follower has to directly transport from plants to depots for this problem too. The lower-level objective function vector becomes:

$$\mathrm{f}(x,y) = \left(\sum_{k \in K} \sum_{l \in L} \mathrm{c}^c_{k,l} y_{k,l}, \sum_{k \in K} \sum_{l \in L} \mathrm{c}^d_{k,l} y_{k,l} \right)$$

$\mathrm{c}^d_{k,l}$ being another operational cost of plant k, to produce and delivering a unit of good to depot l, similar to c^c. While the leader still have to chose how to deliver

[1] Benchmark files are publicly available on the paradiseo website in the problems section at the following URL: `http://paradiseo.gforge.inria.fr/index.php?n=Problems.Problems`.

products from depots to retailers, the follower has to respond to a bi-objective problem , his goal being to find solutions which are Pareto efficient (see [5] for details on Pareto efficiency). We kept the same sets of instances as in BiMDVRP, to which we added the c^d cost independently generated on the same way as the c^c one.

4.6 Experimental Analysis

In order to evaluate the relevance of CoBRA for bi-level optimization, we conduct in this section an experimental analysis against a repairing algorithm, a classical approach which consider the lower-level optimality condition as a constraint, and simply try to find the best upper-level variable while "repairing" the lower-level one at the evaluation step.

4.6.1 Experimental Design

We conduct a two-part experimental analysis. In the first part, we apply the two algorithms on the bi-level multi-depot vehicle routing problem (BiMDVRP). We ran CoBRA and the repairing algorithm for BiMDVRP on S_1, and for M-BiMDVRP on S_1 and S_2. We run both of the algorithms 30 times with different seed values, since both algorithms use stochastic components.

Both algorithms use the same components (*i.e.* the improvement algorithms, the stopping criterion, the variation operators and the initializers). The reparation algorithm does not use any archiving or coevolution operator, and a different evaluation operator which apply a lower-level improvement algorithm before evaluating a solution. Once the stopping criterion is met, we evaluate the population with respect to three criteria:

- the population average upper-level fitness value,
- the direct rationality,
- the weighted rationality.

4.6.2 CoBRA instantiation for BiMDVRP an M-BiMDVRP

To use CoBRA on the BiMDVRP problem, several problem-specific components have to be chosen.

4.6.2.1 Upper-Level Problem-Related Components

For the MDVRP upper problem we use a combination of three variation operators:

RBX [19] is a cross-over operator copying routes from a parent, and then completing the offspring with routes from the other parents by removing visited retailers.

SBX [19] is a cross-over operator creating a new route, by taking half of a route starting from a single depot in each parents, keeping the order of each half, and then completing the offspring with the other routes and removing visited retailers.

Or-opt [17] is a mutation operator taking several retailers from a route and putting it in another. This operator changes the number of route which neither of the SBX and RBX can do.

Operators are applied on solutions uniformly chosen in the population.

4.6.2.2 Lower-Level Problem-Related Components

For the lower-level problem we use a combination of two operators:

UXover [6] is a crossover operator choosing elements uniformly for each parent solution matrix and putting it in the offspring.

Uniform mutation [7] is a mutation operator that add a parametrized real value $r_{lmut} \in [-0.5, 0.5]$ to each element of the solution matrix with a p_{lmut} probability.

4.6.2.3 Stopping Condition

The algorithm uses three stopping criteria, one for each improvement algorithm and one for the overall algorithm. Improvement algorithms use a generational stopping criterion which continue for a fixed number p_g of generations. The overall algorithm uses a lexical continuator which continue until no better solution is found for a fixed parameter p_l of generations, by using a lexical comparator (*i.e.* by comparing sequentially the objective values on each level).

4.6.2.4 Selection Operators

The algorithm uses three selection operators to choose which solution to keep from a generation to the next one, one for each improvement algorithm, and one for the overall algorithm. We use on both improvement algorithms a deterministic tournament, which randomly selects two solutions from the population and keep the best one. For the overall algorithm we use a survive-and-die replacement politic, which keeps a parametrized proportion of the best solutions n_{sad} from the last generation, and apply a deterministic tournament on the remaining part of the population in order to generate the next generation.

4.6.2.5 Archiving Strategy

The algorithm uses archives to keep record of the best solutions found over all generations. We define two different archive strategies depending on the number of lower-level objective function:

Single-objective lower-level strategy. We use a straight-forward archive that keeps the n best found solutions according the the level fitness value

Multi-objective lower-level strategy. The upper-level archive keeps the same strategy than in the single-objective case. The lower-level archive, at the insertion of a new individual i, starts by deleting any solution Pareto-dominated by i then inserts i if it's not dominated by any individual from the archive. If the archive size goes over n, we remove from the archive the worst elements according to the upper-level fitness values until the archive size returned under n.

4.6.2.6 Numerical Parameters

To use those components and CoBRA, the following parameters have to be set:

- n: the populations size, set to 100
- r_{lmut}: the uniform mutation adding parameter, set to 0.5
- p_{lmut}: the uniform mutation probability parameter, set to 0.1
- p_g: the number of generations each improvement generates, set to 10
- p_l: the number of generations CoBRA continues without improvement, set to 100
- n_{sad} the proportion of best solutions that are kept from the last generation, set to 0.8

4.6.3 Experimental Results

4.6.3.1 BiMDVRP

Table 4.2 shows numerical results for CoBRA and the repairing algorithm on instances from S_1. Here are displayed the average upper-level fitness value, and the best fitness value obtained in the lower-level archive, as well as the direct rationality metric value. Since direct rationality was enough to rank the algorithms, the weighted measure was not used.

CoBRA has a significantly better score for the rationality, on all the instances. For both algorithms, rationality is not related to the instance size. The repairing algorithm is doing better for the upper-level fitness value.

4.6.3.2 M-BiMDVRP

Tables 4.3 and 4.4 show the experimental results over the sets S_1 and S_2, respectively. The average and the best upper-level fitness values obtained in the lower archive, and the weighted rationality measure are given. Direct rationality did not permit to significantly decide between the coevolutionary and the hierarchical approach.

Both algorithms obtain similar upper-level fitness values. CoBRA is still having a better rationality. The rationality gap between CoBRA and the repairing algorithm increases with the instance size. The number of evaluations done by the algorithms are shown on Figure 4.5. The repairing algorithms needs a lot more evaluations, impairing the computational cost of the approach.

Table 4.2 Average upper-level fitness value, best upper-level fitness value and weighted rationality value for BiMDVRP instances from S_1

	Averaged fitness		Best fitness		Direct rationality	
Instance	CoBRA	Repair	CoBRA	Repair	CoBRA	Repair
bipr01	1883	**1848**	1676	**1626**	**0.6**	7.5
bipr02	4049	**3338**	3718	**2880**	**1.3**	5.4
bipr03	6058	**5849**	5604	**4712**	**2.7**	23.5
bipr04	**7172**	7368	6568	**6051**	**1.9**	19.9
bipr05	9750	**8535**	9493	**7179**	**0.6**	5.4
bipr06	15237	**11637**	14837	**9656**	**0.7**	5.9
bipr07	3165	**2917**	2851	**2453**	**0.9**	1.5
bipr08	7207	**5348**	6801	**4736**	**2.2**	22.9
bipr09	9825	**8326**	9343	**7042**	**2.0**	22.2
bipr10	14418	**12413**	13419	**12412**	**0.6**	13.5

Table 4.3 Average upper-level fitness value, best upper-level fitness value and weighted rationality value for M-BiMDVRP instances from S_1

	Average Fitness		Best Fitness		Weighted rationality	
Instance	CoBRA	Repair	CoBRA	Repair	CoBRA	Repair
mbipr01	**3151**	3570	**2930**	3002	**0.66**	21.50
mbipr02	**5980**	6559	**5729**	5792	**4.83**	140.50
mbipr03	**11459**	12369	**10887**	11230	**77.49**	562.76
mbipr04	**12985**	14346	**12568**	13158	**6.84**	195.84
mbipr05	**16067**	16872	**15317**	15982	**1.82**	52.89
mbipr06	**19408**	21291	**18523**	20079	**158.02**	839.89
mbipr07	**5195**	5790	4915	**4758**	**8.71**	253.39
mbipr08	**10566**	11691	**9943**	10543	**21.36**	106.70
mbipr09	**15948**	17519	**15330**	16247	**41.13**	727.62
mbipr10	**20849**	22798	**20361**	21523	**86.45**	1040.10

Table 4.4 Average upper-level fitness value, best upper-level fitness value and weighted rationality value for M-BiMDVRP instances from S_2

	Averaged fitness		Best fitness		Weighted rationality	
Instance	CoBRA	Repair	CoBRA	Repair	CoBRA	Repair
mbipr01	**3187**	3630	**2912**	3155	**16.69**	67.52
mbipr02	**6155**	6798	**5808**	6236	**25.61**	89.10
mbipr03	**11226**	12390	**10865**	11544	**31.55**	197.58
mbipr04	**13703**	14934	**13240**	14113	**27.18**	208.44
mbipr05	**15349**	16773	**14753**	16092	**111.10**	357.05
mbipr06	**19894**	21986	**19314**	21132	**45.41**	306.62
mbipr07	**5243**	5849	**4796**	5239	**18.86**	82.46
mbipr08	**10598**	11649	**10131**	10866	**15.02**	198.85
mbipr09	**15862**	17517	**15357**	16535	**21.61**	229.93
mbipr10	**20747**	22843	**20207**	22019	**39.43**	349.27

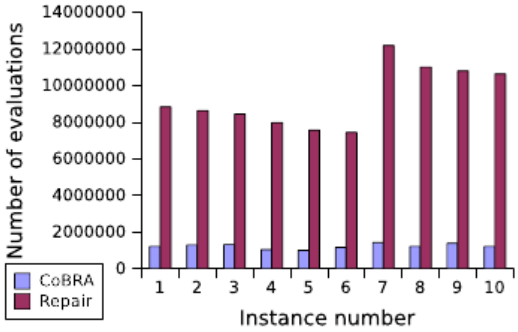

Fig. 4.5 Average number of evaluations required by CoBRA and the repairing algorithm on M-BiMDVRP instances from Set S_2

4.6.3.3 Discussion

CoBRA has a significant advantage in terms of rationality for all the runs we performed, while it does not always give a better upper-level fitness value. Rationality indicates the quality of the reaction predicted by the algorithm. A bad prediction is likely to lead to a bad solution: once applied to a real-life situation the follower will have greater chances to chose a better reaction for his own objective function(s), degrading the solution quality for the leader. Since CoBRA has a better rationality, we can better predict the outcome of the decisions. Thus we can conclude that CoBRA is more adapted to the bi-level aspect of the problem.

An explanation why the hierarchical algorithm does not select the more rational response would be that once an irrational solution $x = (x_u, x_l)$ is obtained, through a badly done reparation, which gives a better upper-level fitness value than the more rational response $x' = (x_u, x'_l)$, the overall algorithm will have a tendency to discard x' and keep x. We can conclude that the reparation approach needs either a good lower-level heuristic, an exact lower-level algorithm, or some properties over the problem (such as a strong correlation between the two levels) to be able to produce rational responses. This is the reason why the coevolution allows CoBRA to get a better rationality. We can conclude that the coevolutionary approach can give a significant enhancement for this problem.

4.7 Conclusions and Future Works

In this paper, we described *CoBRA*, a new general methodology to solve bi-level optimization problems. We introduced the concept of *rationality* for bi-level optimization problems and two new metrics to compare the performance of evolutionary algorithms for such purpose. Using those two metrics, we compared CoBRA against

a classical hierarchical approach on a bi-level optimization problem of production/-transportation in its single-objective and multi-objective variants. Experimental results showed a significant advantage to the CoBRA approach in tackling the bi-level multiple depot problem against a classical hierarchical approach.

As future work, it would be interesting to look up a possible integration of diversification principles into CoBRA. Instead of considering the upper-level fitness values in the lower-level archive, it could be more efficient to keep a good diversity in the archive. This would give the opportunity for the algorithm to escape from local optima easier. Furthermore, the design of CoBRA is intrinsically parallel, since two sub-populations evolve independently,parallel computation would improve the performance in terms of computational time.

References

1. Anandalingam, G., White, D.: A solution method for the linear static stackelberg problem using penalty functions. IEEE Transactions on Automatic Control 35(10), 1170–1173 (1990)
2. Ben-Ayed, O., Blair, C.: Computational difficulties of bilevel linear programming. Operations Research 38(3), 556–560 (1990)
3. Calvete, H., Galé, C.: A Multiobjective Bilevel Program for Production-Distribution Planning in a Supply Chain. In: Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, pp. 155–165 (2010)
4. Cordeau, J., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2), 105–119 (1997)
5. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York (2001)
6. Deb, K., Agrawal, R.: Simulated binary crossover for continuous search space. Complex systems 9(2), 115–148 (1995)
7. Deb, K., Goyal, M.: A combined genetic adaptive search (geneas) for engineering design. Computer Science and Informatics 26, 30–45 (1996)
8. Didi-Biha, M., Marcotte, P., Savard, G.: Path-based formulations of a bilevel toll setting problem. In: Optimization with Multivalued Mappings, pp. 29–50 (2006)
9. Eichfelder, G.: Multiobjective bilevel optimization. Mathematical Programming 123(2), 419–449 (2010)
10. Erkut, E., Gzara, F.: Solving the hazmat transport network design problem. Computers & Operations Research 35(7), 2234–2247 (2008)
11. Fliege, J., Vicente, L.: Multicriteria approach to bilevel optimization. Journal of optimization theory and applications 131(2), 209–225 (2006)
12. Koh, A.: Solving transportation bi-level programs with differential evolution. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 2243–2250. IEEE (2008)
13. Li, X., Tian, P., Min, X.: A hierarchical particle swarm optimization for solving bilevel programming problems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 1169–1178. Springer, Heidelberg (2006)
14. Loridan, P., Morgan, J.: Weak via strong stackelberg problem: New results. Journal of Global Optimization 8, 263–287 (1996), doi:10.1007/BF00121269

15. Lv, Y., Hu, T., Wang, G., Wan, Z.: A penalty function method based on Kuhn-Tucker condition for solving linear bilevel programming. Applied Mathematics and Computation 188(1), 808–813 (2007)
16. Oduguwa, V., Roy, R.: Bi-level optimisation using genetic algorithm. In: 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002), pp. 322–327 (2002)
17. Or, I.: Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Northwestern University, Evanston (1976)
18. Potter, M.A., Jong, K.A.D.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8, 1–29 (2000)
19. Potvin, J., Bengio, S.: The vehicle routing problem with time windows part II: genetic search. INFORMS Journal on Computing 8(2), 165 (1996)
20. Talbi, E.-G.: Metaheuristics: from design to implementation. Wiley (2009)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)