A co-evolutionary decomposition-based algorithm for the bi-level knapsack optimisation problem

Abir Chaabani* and Lamjed Ben Said

SMART Lab,
ISG,
University of Tunis, Tunisia
Email: abir.chaabani@gmail.com
Email: lamjed.bensaid@isg.rnu.tn
*Corresponding author

Abstract: Bi-level optimisation problems (BOPs) are a class of challenging problems with two levels of optimisation tasks. These problems allow to model a large number of real-life situations in which a first decision maker, hereafter the leader, optimises his objective by taking the follower's response to his decisions explicitly into account. In this context, a new proposed algorithm called CODBA-II was suggested to solve combinatorial BOPs. The latter was able to improve the quality of generated bi-level solutions regarding to recently proposed methods. In fact, a wide range of applications fit the bi-level programming framework and real-life implementations still scarce. For this reason, we propose in this paper a co-evolutionary decomposition-based bi-level algorithm for the bi-level knapsack optimisation problem. The computational algorithm turned out to be quite efficient on both computation time and solution quality regarding to other competitive EAs.

Keywords: bi-level combinatorial optimisation; evolutionary methods; bi-level knapsack problem; BKP.

Reference to this paper should be made as follows: Chaabani, A. and Ben Said, L. (2020) 'A co-evolutionary decomposition-based algorithm for the bi-level knapsack optimisation problem', *Int. J. Computational Intelligence Studies*, Vol. 9, Nos. 1/2, pp.52–67.

Biographical notes: Abir Chaabani received her BSc, MSc and PhD in Computer Science from the University of Tunis, Tunisia, in 2011, 2013, and 2017, respectively. She is an assistant of Computer Science at the Higher Institute of Management of Tunis, Tunisia. Her main research interests are evolutionary computation, multi-objective optimisation, metaheuristics, bi-level optimisation, and their applications.

Lamjed Ben Said received his BSc in Computer Science from the University of Tunis, Tunisia, in 1998. He obtained his MSc and PhD in Computer Science from the University of Paris VI, France, in 1999 and 2003 respectively. He was a research fellow for 3 years at France Telecom (R&D). He is also a Professor of Computer Science at the University of Tunis. He is also the head of the SMART lab and the director of the Higher Institute of Management of Tunis. His research interests cover multiagent

systems, multi-criteria decision-making, evolutionary computation, supply chain management, and behavioural economics.

1 Introduction

Bi-level programs have been the subject of extensive study both from a theoretical and a practical point of view (Dempe, 2003). These type of problems describe a hierarchical system which is composed of two levels of decision makers (i.e., optimisation tasks). The outer optimisation task is commonly referred to as the upper level optimisation problem and the inner one is commonly referred to as the lower level optimisation problem. In such situation, the lower level problem appears as a constraint, such that only an optimal solution to the lower level problem is a possible feasible candidate to the upper level one. This nested structure explain the difficulty associated with solving this class of problems. For this reason, there have several effort towards developing non-nested solution methods in order to solve BOPs. Due to the difficulty associated with solving such NP-hard problems, most proposed works studied for a long time tackle the linear case (i.e., the simplest case) in which all objective functions and constraints are linear with respect to the decision variables (Wen and Hsu, 1991; Jeroslow, 1985). This gives an idea about the kind of challenges offered by bi-level problems with complex (nonlinear, non-convex, discontinuous, etc.) objective and constraint functions.

A variety of resolution methods have been developed, involving classical and evolutionary approaches. The first category commonly used to handle bi-level problems include the KKT approach (Kuhn and Tucker, 1951), branch-and-bound method (Bard and Falk, 1982), and descent method (Savard and Gauvin, 1994), etc. Despite the significant progress made in classical optimisation towards solving bi-level optimisation problems (BOPs), most of these approaches are rendered inapplicable for bi-level problems with higher levels of complexity. The main shortcoming of these methods is that they heavily depend on the mathematical characteristics of the BOP, which makes them unsuitable to handle real world problems' difficulties such dimensionality, nonlinearity, etc. For this reason, most of the classical proposed approaches are too restrictive and they are applicable only to a small class of BOPs. Motivated by this observation, several researchers have recently proposed the use of evolutionary algorithms (EAs) to solve BOPs. These techniques have been successfully applied to handle mathematical programming problems and applications that do not adhere to regularities like continuity, differentiability or convexities. Due to these properties of EAs, attempts have been made to solve BOPs using these methods such as Shepherd and Sumalee (2004), Legillon et al. (2012), Marinakis et al. (2007), Chaabani et al. (2016), just to cite a few. Although, EAs are able to tackle complex BOPs, their use is computationally costly since we need to make a very high number of function evaluations (FEs) to solve the problem. In fact, the evaluation of each upper level solution requires executing another EA to find the lower level (near-) optimal solution, thereby making the overall bi-level optimisation computationally very intensive. Taking the example of a lower population of 50 individuals that is evolved for 50 generations, the evaluation of each upper level solution requires 2,500 (50 * 50) additional FEs. Now, if the upper level population size is 100 and it is evolved for 100 generations, the total number of FEs of this nested evolutionary scheme is 25 millions FEs, which is

computationally costly and time consuming. To address these problems, attempts have been made to reduce the computational expense of evolutionary bi-level optimisation (EBO) by utilising meta-modeling-based principles. We cite the work of Sinha et al. (2013) who proposed the use of quadratic approximation at the lower level to have an approximated optimal solution with very few number of FEs. This meta-modeling-based approach is shown to be efficient on many benchmark problems. However, this work is restricted to the continuous case.

An other interesting observation regarding the EBO literature consists in that most works have been recently proposed for continuous search spaces. However, little attention has been paid for the combinatorial case. The main reason of this interest is the relative facility associated with continuous optimisation problems (Talbi, 2013). To this end, we need to expect a growing interest in solving combinatorial BOPs. For all these reasons, we have recently proposed a new family of co-EAs based on decomposition mechanism:

- A co-evolutionary decomposition-based bi-level algorithm (CODBA) (Chaabani et al., 2015a, 2017a): Is a nested bi-level approach applying a co-evolutionary scheme at the lower level problem. CODBA exploits three main mechanisms which are decomposition, multi-threading, and co-evolution within the lower level in order to cope with the high computational cost of BOPs. The idea is to use a decomposition-based method as a surrogate to the lower level complexity. In this way, the proposed approach decomposes the lower level population into a set of well-distributed sub-populations using a new proposed method for discrete decision space decomposition called discrete space decomposition method (DSDM). The generated sub-populations co-evolve simultaneously in a parallel manner using multiple threads and exchange information via recombination with the best lower level individuals.
- An improved CODBA-II (Chaabani et al., 2015b): Is an improved version of the CODBA method. It incorporates decomposition, multi-threading, and co-evolution within both levels (upper and lower) with the aim to further cope with the high computational cost of the over-all bi-level search process and to improve the quality of generated solutions.

The proposed approaches have demonstrated a good performance on the bi-level multi-depot vehicle routing problem (Bi-MDVRP), which is a well-known NP-hard combinatorial BOP (Chaabani et al., 2015a). Thus, it will be interesting to evaluate the proposed scheme on other several combinatorial problems to prove the merit of the proposed approach from effectiveness and efficiency viewpoints.

Many real-life and well-known academic and real world problems are generally modeled as combinatorial BOPs, e.g., in transportation (network design, optimal pricing), economics (Stackelberg games, principal-agent problem, taxation, policy decisions), management (network facility location, coordination of multi-divisional firms), engineering (optimal design, optimal chemical equilibria), just to cite a few. Dempe identified more than 80 references in the literature describing applications of bi-level problems (Mansi et al., 2012). In particular, the bi-level knapsack problem (BKP), which represents an interesting example of discrete bi-level problem, has been proposed in the bi-level framework, modelling different real world situations. Moreover, real applications of this problem can be found in revenue management,

telecommunications, capacity allocation, and transportation, etc. For this reason, we are interested to solve the BKP using our co-evolutionary decomposition-based algorithm.

The main contributions of this paper are the following:

- Applying a co-evolutionary decomposition-based (CODBA-II) scheme to solve the bi-level KP in which new bi-level formulation of the KP variants is used.
- Reporting comparative results between CODBA-II, CODBA, COBRA, and a hierarchical EAs for solving the BKP.

2 Bi-level optimisation concepts

As its noun indicates, a BOP contains two levels of optimisation:

1 the upper level task which is called the leader problem

2 the lower level which is called the follower problem.

Each level has its own objective function, constraints, and decision variables. In consequence, we have two kinds of variables:

1 the upper level variables x_u

2 the lower level variables x_l.

It is important to note that for the follower problem, the optimisation task is performed with respect to the variables x_l and the variables x_u act as parameters. Hence, for each x_u corresponds a different follower problem whose optimal solution needs to be determined. All variables (x_u and x_l) are considered in the leader problem and the optimisation is expected to be performed with respect to both sets of variables. The analytical formulation of a BOP is given by the following definition:

Definition 1 (BOP): BOP is formulated as:

$$\underset{x_u \in X_U, x_l \in X_L}{Min} F(x_u, x_l) \tag{1}$$
$$s.t. \begin{cases} x_l \in ArgMin \{f(x_u, x_l), \} g_i(x_u, x_l) \leq 0, \\ G_j(x_u, x_l) \leq 0,\ i = 1, ..., I \, and \, j = 1, ..., J \end{cases}$$

where $G_j : X_U \times X_L \to \mathbb{R}$ denotes the upper level constraints, and $g_i : X_U \times X_L \to \mathbb{R}$ represents the lower level constraints, respectively. The difficulty in bi-level optimisation arises from the fact that only lower level optimal solutions can be considered as feasible members, if they also satisfy the upper level constraints. For instance, a member $x^1 = (x_u^1, x_l^1)$ can be considered feasible at the upper level only if x^1 satisfies the upper level constraints, and x_l^1 is an optimal solution to the lower level problem corresponding to x_u^1. The decision variables x_u and x_l could be continuous, discrete, or even mixed. In this paper, we are interested in problems where variables at both levels are discrete. Moreover, the presented definition of bi-level optimisation corresponds to the optimistic case which is commonly studied in the literature (Sinha

et al., 2017).

Definition 2 (constrained set): The constraint set of BOP is:

$$\Omega = \{(x_u, x_l)/G(x_u, x_l) \leq 0,\ and\, g(x_u, x_l) \leq 0\} \tag{2}$$

For each upper-level vector x_u, $g(x_u, x_l) \leq 0$ defines the feasible set $\Omega(x_u)$ of the lower level problem.

Definition 3 (rational reaction set): The rational reaction set $M(x_u)$ may be defined as:

$$M(x_u) \in \Omega(x_u) = Argmin\{f(x_u, x_l) :\ g(x_u, x_l) \leq 0\} \tag{3}$$

The rational reaction set $M(x_u)$ reflects the dependence of the decision taken at the upper-level on the decision taken at the lower level. Considering such vector x_u, the vector x_l is chosen as an optimal solution $x_l = M(x_u)$ of the lower optimisation problem parameterised by vector x_u. In this way, the solution $M(x_u)$ may be seen as the rational reaction of the follower on the leader's decision x_u. Besides, $M(x_u)$ may be composed of multiple optimal solutions, i.e., there is a multiple lower level optimal vector corresponding to any upper level solution. Moreover, $M(x_u)$ can contain only a single valued vector, i.e., there is a single lower level optimal vector corresponding to any upper level.

3 BKP: overview

Knapsack problem (KP) is one of the classical NP-hard problems in combinatorial optimisation and computer science. There are a lot of generalisation of the classical KP as a bi-level model (BKP) in the literature. The first attempts was proposed by Dempe (2002) which considers two decision makers in different levels (i.e., leader and follower). The leader first determines the capacity of the knapsack and afterwards the follower, assumed to be selfish, packs items to the knapsack in order to maximise his own profit. In other words, the lower level problem solves a 0–1 KP subject to the capacity set by the leader. The latter earns a profit from the items selected by the follower, and consequently both decision makers seek to maximise their own profits. Dempe and Ritcher (2000) formulated this problem as a mixed-integer bi-level program, and proposed a branch-and-bound algorithm to solve it. This formulation gives rise to two variants:

1 the optimistic case where once the follower encounters multiple optimum solutions, he chooses the one that maximises the objective of the leader

2 the pessimistic case where he chooses just the opposite.

In Brotcorne et al. (2009) considered the same problem formulation of Dempe and Richer, and developed a dynamic programming algorithm that was able to outperform Dempe and Richter's branch-and-bound algorithm. This presented BKP formulation is well suited to model real applications like right financing problem or revenue management problem, in which, we found a leader entity shares his capital between saving accounts with fixed rate of return and a risky investment, through an intermediary, such as, a bank or a broker. The intermediary which is the follower entity

1 buys shares or bonds to maximise his revenue while respecting the leader's budget constraint (knapsack constraint)

2 obtains a return from his own investments.

In Mansi et al. (2012) consider a bi-level knapsack variant where both decision makers pack items into a single common knapsack with a pre-specified capacity C. The item set is split into two parts. The leader starts by packing some of his items into the knapsack, and then the follower adds some further items from his set. Indeed, DeNegree (2011) suggests yet another variant, where both players hold their own private knapsacks and choose items from a common item set. First, the leader packs some of the items into his private knapsack, and then the follower picks some of the remaining items and packs them into his private knapsack. The objective of the follower is to maximise the profit of the items in his knapsack, and the objective of the leader is to minimise this profit.

Lastly, Chen and Zhang (2013) proposed another bi-level knapsack formulation in which each level controls a knapsack set with probably different capacity. Differently to the above presented formulations, the leader does not control the other's knapsack, however, he can influence the profit of the items. Given a set of items, each with a fixed size and a profit, the two levels select items and pack them into their own knapsacks under the capacity constraint. Same items can be packed simultaneously to different knapsacks. However, in this case, the profit of such items can vary. In such situation, the leader problem packs items into his knapsack to maximise the total profit, while the follower problem should care about the total profits of items packed into the two knapsacks in order to maximise his own profit. A real example of this scenario is the following: consider the item of a knapsack as a project and the leader and the follower problems are two investors. To put project i into investor j, it would cost him w_i, and reward him with a profit of $p_i + a_i$. In this way, a project i has a p_i revenue if it is chosen by one investor, otherwise, it takes a $p_i + a_i$ revenue if it is chosen by both investors. In this way, a_i can be either positive or negative depending on the leader-follower relation (cooperative or conflictive). Consequently, the profit of a project depends on the decision taken at both investors. The Chen and Zhang formulation is described by equations (4) and (5).

$$\begin{aligned} \underset{x_u, x_l}{Max}\ F(x_u, x_l) = & \sum_{i=1}^{n} p_i(x_i + y_i) + 2\sum_{i=1}^{n} a_i x_i y_i \\ s.t. & \begin{cases} \sum_{i=1}^{n} w_i x_i \leq W_1 \\ x_i \in \{0, 1\}, \end{cases} \end{aligned} \tag{4}$$

Equation (4) presents the upper level problem formulation in which n denotes the number of items to be chosen. Each item i has a w_i weight and a profit p_i. Let $x_i,\ y_i \in \{0, 1\}$ correspond to the choices of the leader and follower for item i. W_1 is the total capacity controlled by the leader. The first part in equation, is used to maximise the total profit of the leader and the follower tasks. While, the second part treats the influence performed between the two tasks. In this way, a_i takes 0 if item i is chosen by one level, otherwise, each problem gains $p_i + a_i$ (a_i could be positive or negative). We note

that a company (one level problem) cares only about its own profit. The lower level formulation is described by equation (5):

$$\underset{x_u,x_l}{Max}\, f(x_u,x_l) = \sum_{i=1}^{n} p_i y_i + \sum_{i=1}^{n} a x_i y_i \quad (5)$$
$$s.t. \begin{cases} \sum_{i=1}^{n} w_i y_i \leq W_2 \\ y_i \in \{0,1\} \end{cases}$$

In equation (5), W_2 denotes the knapsack capacity controlled by the follower. We assume here that when multiple choices are available to the follower, he will choose one arbitrarily. An example of cooperative real interpretation of this formulation could be that, the leader is the government, and the follower is a company. The government tries to improve the social income (i.e., maximise the revenue of both levels). As well, a private company wants to maximise his own profits.

4 A co-evolutionary decomposition-based algorithm for the BKP

Recently, a CODBA-II (Chaabani et al., 2015b) has been proposed that is shown to perform better than representative and prominent works in the bi-level combinatorial optimisation research area. This latter has demonstrated a good results on the well-known bi-MDVRP combinatorial problem. Thus, the main motivation behind this work is to exploit the CODBA-II scheme to solve an other interesting bi-level combinatorial problem which is the BKP. In this paper, we choose to use the Chen and Zhang bi-level formulation which is well adopted to model several real-world situations.

CODBA-II is proposed as a nested approach with a co-evolutionary scheme at both upper and lower level in order to exploit the strengths of both strategies (nested and co-evolutionary methods) and to cope with the whole BOP cost. This scheme applies three main mechanisms which are:

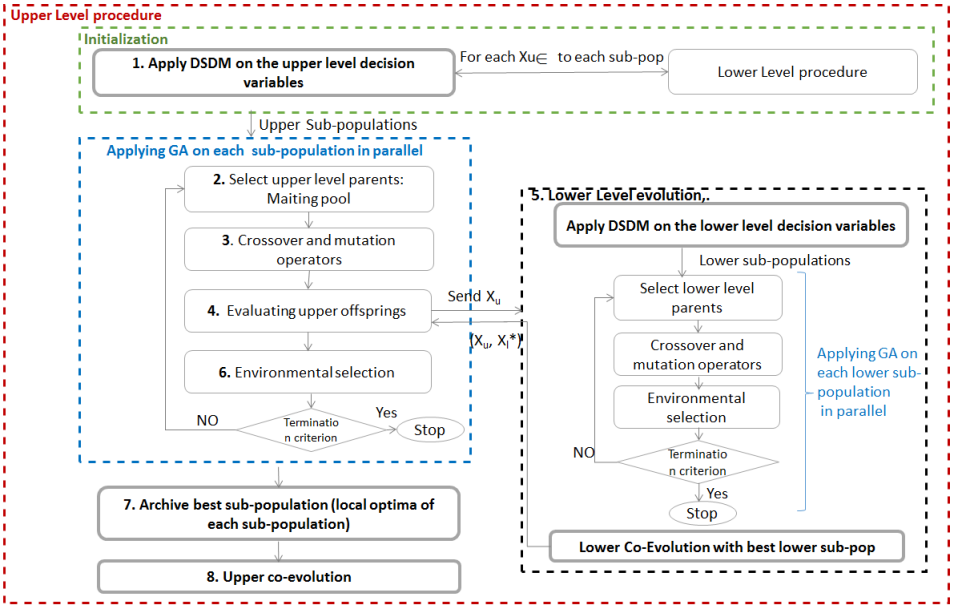
1 the decomposition

2 the multi-threading

3 the co-evolution to reduce the bi-level complexity and to improve the solution quality.

The first mechanism consists in generating several well-distributed sub-populations over the whole search space. Each sub-population could be seen as a cluster. Thus, the clusters'centroids are well-distributed to cover as possible the whole search space. In this way, each generated sub-population controls and evolves a specific region in order to find the local optimum. These sub-populations are evolved in parallel using multiple threads such that each one is assigned to a distinct thread. When the evolution process of the different sub-populations terminates, we store their best solutions into an archive and a co-evolution step takes place to keep a global view on the whole problem. Figure 1 illustrates the basic idea of CODBA-II-KP and step-by-step procedure of the proposed algorithm on the BKP is described in the following subsection.

Figure 1 Basic scheme of CODBA-II (see online version for colours)



4.1 Upper level optimisation procedure

Step 1 *Initialisation scheme:* We generate M_1 well-distributed upper level sub-populations on the whole discrete decision space using our recently proposed decomposition-based method, called DSDM. The latter is a DSDM proposed to generate diversified sub-populations in discrete decision space. The method applies a fixed parameter F_{dec} to determine the M_1 upper sub-populations. For more details about the DSDM method, readers are invited to confer to Chaabani et al. (2017b). Indeed Figure 2 presents an illustrative example of the method application with a small size problem.

At this stage, the lower level optimisation problem is executed to identify the optimal lower level solutions. In fact, the upper level fitness is assigned based on both upper level function value and constraints [described by equation (4)].

We note that in this work we code the upper solution using a permutation having size equal to the number of items. Thus, a binary encoding is used, where every chromosome is a string of bits, 0 or 1. If one decision maker chooses item i, then $x_i = 0$ otherwise $x_i = 1$.

Step 2 *Upper level parent selection:* We choose $(\frac{SPS_1}{2})$ population members from each upper level parent sub-population using tournament selection operator where SPS_1 is the upper sub-population size.

Step 3 *Variation at the upper level:* Perform the crossover and mutation operations in order to create an offspring sub-population for each upper parent sub-population. We choose to use the single point crossover and the bit

string mutation for crossover and mutation, respectively. We note that these operators are performed in parallel using the multi-threading mechanism.

Step 4 *Lower level optimisation:* Solve the lower level optimisation problem for each offspring using the decomposition-based co-evolutionary parallel scheme (presented in Subsection 4.2).

Step 5 *Offspring evaluation:* Combine each upper parent sub-population with its corresponding upper offspring population and evaluate them using the upper level objective function and constraints [presented in equation (4)].

Step 6 *Environmental selection:* Fill each new upper level sub-population using a replacement strategy. In fact, each new upper level sub-population is formed with the SPS_1 best solutions of the combined one. If the stopping criterion is met then store the best found upper level solution in the archive; otherwise, return to step 2.

Step 7 *Co-evolution:* The archive contains the best found solutions regarding to the evolved sub-populations (i.e., local optima solutions). Thus, to have a global view of the whole search space and to be able to determine the global (near-) optimal solution, a co-evolution step is required. Each sub-population member is crossed-over with one of the best archive members of the other sub-populations. Thereafter, we combine each sub-population with its obtained offspring sub-population and we update the sub-population by selecting the best SPS_1 ones. This process is repeated until the best upper level fitness value is no more improved for a K generations or $MaxGenCoEvol$ is attained where $MaxGenCoEvol$ is the maximum allowed number of generations for co-evolution. It is important to note that all sub-populations are to be evolved simultaneously using a thread for each subpopulation.

4.2 Lower level optimisation procedure

Step 1 *Lower level decomposition:* For each upper level solution, we generate M_2 well-distributed lower level sub-populations on the whole discrete decision space using the DSDM method. Once these sub-populations are generated, each sub-population member is evaluated using the lower level objective function and constraints [equation (5)].

Step 2 *Lower level parent selection:* We choose ($\frac{SPS_2}{2}$) population members from each lower level parent subpopulation using tournament selection where SPS_2 is the lower sub-population size.

Step 3 *Variation at the lower level:* Perform the single point crossover and the bit string mutation for crossover and mutation operators, respectively.

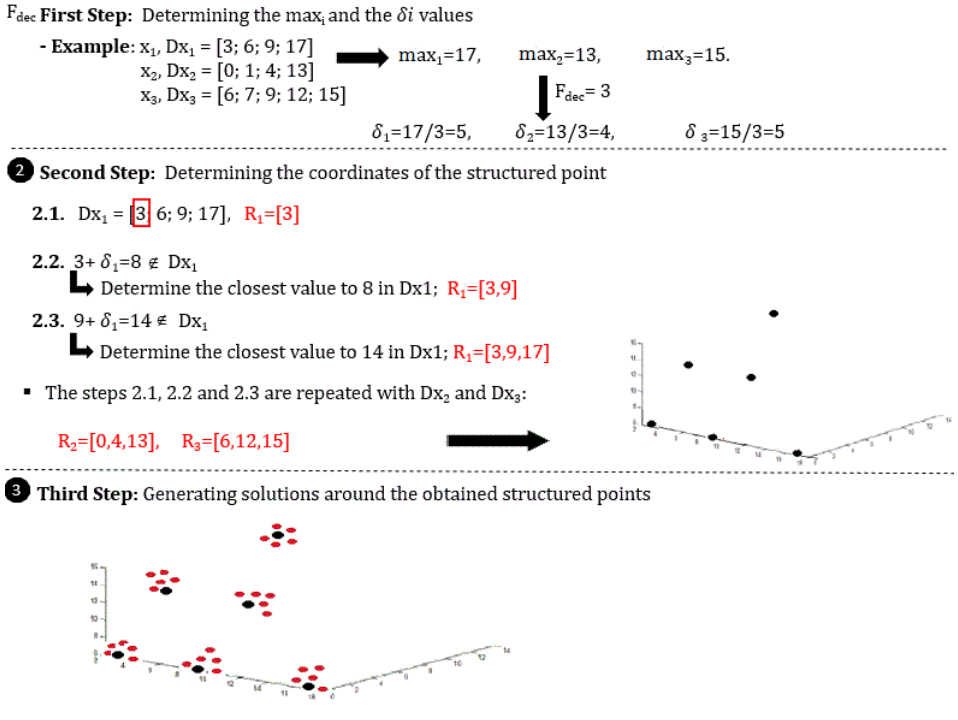
Step 4 *Offspring evaluation:* Combine each parent sub-population with its corresponding offspring population and evaluate them using the lower level objective function and constraints [presented in equation (5)].

Step 5 *Environmental selection:* Fill each new lower level sub-population using a replacement strategy. In fact, each new lower level sub-population is formed

with the SPS_2 best solutions of the combined one. If the stopping criterion is met then store the best found lower level solution in the archive; otherwise, return to step 2.

Step 6 *Co-evolution:* Similarly to the upper level procedure, each sub-population member is crossed-over with an archive members that represents a best local optimum than the current one. In this way, we obtain an offspring population for each sub-population. Thereafter, we combine each sub-population with its corresponding offspring population and we update the sub-population by selecting the best SPS_2 ones. This process is repeated until the best lower level fitness function value is no more improved for a K generations or $MaxGenCoEvol$ is attained.

Figure 2 Illustration of the DSDM with an exemplified decision problem with three variables (see online version for colours)



5 Experimental study

The main motivation of this paper is to evaluate the performance of the recently proposed CODBA-II scheme on the well-known combinatorial BKP. In this section, we first present the empirical study design: the used benchmark problems, the performance indicators, and the used parameters setting. Then, we report a comparative experiments against three algorithms within this research area:

1 CODBA: The baseline co-evolutionary decomposition-based algorithm that follows the nested structure with a co-evolutionary decomposition method at the lower level only (Chaabani et al., 2015a).

2 COBRA: A cooperative co-evolutionary bi-level algorithm (Legillon et al., 2012).

3 Repair method: A simple nested strategy that uses genetic algorithm at both levels to handle BOPs.

Moreover, to facilitate a fair comparison, the operators and parameter setting of both algorithms are kept the same. In this manner, any improvement attained in the search can be attributed to the efficiency of such algorithm scheme. We note that all algorithms are coded in Java programming language and all simulations are performed on the same machine (Intel Core i7-4500 CPU 1.8 GHz, 8 GB RAM).

Table 1 Description of the used BKP benchmark problems

	Knapsack_capacity	*Number-items*
Bi-P01	165	10
Bi-P02	26	5
Bi-P03	190	6
Bi-P04	50	7
Bi-P05	104	8
Bi-P06	170	7
Bi-P07	750	15
Bi-P08	6,404,180	24
Bi-PL1	995	100
Bi-PL2	1,008	200
Bi-PL3	2,543	500
Bi-PL4	5,002	1,000
Bi-PL5	10,011	2,000

5.1 Problems instances

In this subsection, we describe the different benchmark problems used in our experimental study. In fact, we utilise commonly used test problems within the standards KP community that allow assessing the performance of any algorithm design with respect to different kinds of difficulties (small and large size problems) (Martello, 1990). These data tests are all based on randomly distributed profits and weights. The description properties instances are given in Table 1. The other details of Bi-P instances are extracted from the KNAPSACK_01 data-set directory (Martello, 1990). Indeed the details of Bi-PL instances are described via the following link: http://artemisa.unicauca.edu.co/johnyortega/instances_01_KP/.

5.2 Performance metrics

We aim to compare the results using three indicators value to ensure that both upper solutions and lower reactions are improved using our algorithm. In the upper level,

we use the average upper level fitness (UF) as a performance indicator to evaluate the quality of generated bi-level solutions. In fact, in the lower level Legillon et al. (2012) propose the notion of rationality to tackle this issue. Rationality is based on the proximity from the optimum of the lower-level variables corresponding to a fixed upper-level variables x_u. Rational solution is a solution where the follower reaction is rational, seeking for the optimality of its own objective function. In this context, two rationality metrics were proposed in the literature and used in this work:

1 the direct rationality (DR)

2 the weighted rationality (WR).

The first one corresponds to the difficulty of improving a solution without regarding the actual gap of improvement. In other words, we count how many times the algorithm has improved the solution. While the second metric focus on the improvement rate of the solution, i.e., how much the solution was improved. The three used measures compare the capacity of an algorithm to use their given level specific methods.

5.3 Parameter setting

This section is devoted to detail the parameter setting for each of the used algorithms under comparison for the BKP problem. To ensure fairness of comparison, we use the same number of FEs as a termination criterion. This stopping criterion is set to 2,000 FEs for all used benchmark problems. The number of generations used by CODBA-II-KP and CODBA-KP for the co-evolution step is fixed to 2. The other common used parameters setting are as follows: population size, crossover probability, mutation probability, and F_{dec} factor which are set to 100, 0.9, 0.1 and 2, respectively. The a factor used in equations (4) and (5) is set to (–10) (a competitive version of BKP). We mention here that we have used the trial-and-error method for tuning parameter values (Eiben and Smit, 2011).

5.4 Comparative results

To compare between the three considered non-deterministic algorithms, we perform 31 runs for each couple (algorithm, problem instance) as recommended in Derrac et al. (2011). Indeed, for each test problem and each pair of algorithms, we perform a (two-sided) Wilcoxon test to decide whether or not the difference between the indicated average values (for each performance indicator) for both algorithms is statistically significant on the considered problem instance. The results of the significance tests for the pairwise comparisons at level α = 0.05 are presented in the form of (–: no significance) and (+: significance) in the order on which the algorithms appear. Table 2 shows the numerical results for CODBA-II-KP, CODBA-KP and repair-KP.

As shown in the table, CODBA-II obtains very competitive results for most test problems. It achieves the best average upper level fitness value on all used instances. The second-best value is also attained by the baseline CODBA version. This result validates the effectiveness and the efficiency of the upper decomposition-based scheme of CODBA-II in generating high quality bi-level solution compared to the cooperative COBRA method and the basic nested repair approach on solving the BKP. Regarding

to the lower level, CODBA-II performs strictly better than CODBA and repair. The best lower level reactions is obtained also by CODBA-II on all benchmark problems. Additionally, we can note that the CODBA-II scheme provides a better performance compared to the baseline algorithm with statistical significance. This observation is due to the fact that a CODBA-II algorithm is able to better explore both upper and lower level space. The worst performance is always observed with the nested algorithm especially at the lower level compared to CODBA and CODBA-II on the majority of test problems. This validates the need to incorporate an efficient method to the lower level of the nested method in order to be able to produce good responses.

To more emphasise the performance of the algorithm on solving the BKP regarding to its competitors, we present in Figure 3 the average fitness value for a real data set problem. This latter was deduced from the 0–1 KP presented in Rosetta code available via the link https://rosettacode.org/wiki/Knapsack_problem/0-1. The problem consists on a trip made by a tourist with his friend who want to maximise the profit of things to bring for the trip within a knapsack capacity of 4 Kg. We have just make a small change to the task description to be compliant with the model of Chen and Zhang (2013). In this way, we add a second KP detained by his friend with a capacity of 2 Kg. We should note here that we are using a cooperative version of the BKP model presented in Figure 3.

Figure 3 Average upper level fitness values of CODBA-II, CODBA, COBRA and the nested method for the BKP real-data instance (see online version for colours)

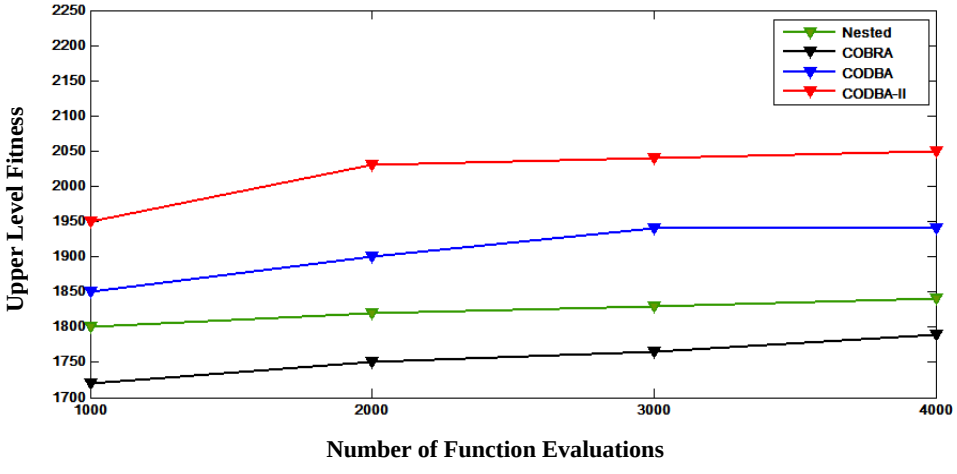


Figure 3 reports the performance of the four used algorithms on this real-world instance, in which the Y-axis denotes the average upper level fitness value while the X-axis gives the different used stopping criterion values in terms of number of function evaluations. The results suggest that best performance on the different termination criteria levels is obtained by CODBA-II (higher upper fitness) regarding its competitors.which proves that the co-evolutionary decomposition-based approach is more adapted to the bi-level aspect of the problem and was able to solve efficiently the BKP regarding other competitive approaches.

Table 2 Average upper level fitness values of CODBA-II, CODBA, COBRA and the nested method (repair) on BKP

Instances		*CODBA-II*			*CODBA*			*COBRA*			*Nested*		
		UF	*DR*	*WR*	*UF*	*DR*	*WR*	*UF*	*DR*	*WR*	*UF*	*DR*	*WR*
Small scale	Bi-P02	91	13	*865*	91	33	1,512	85	46	1,612	91	76	2,116
		(+++)	(+++)	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-P03	*265*	*81*	*3,840*	265	95	3,975	265	95	4,100	265	97	4,120
		(+++)	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-P05	*1,738*	35	145	1,713	*33*	*110*	1,700	63	160	1,710	65	170
		(+++)	(+++)	(+++)	(+++)	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
Medium scale	Bi-P01	*574*	*30*	*60*	564	33	64	515	34	75	388	42	79
		(–++)	*(– – +)*	*(+++)*	(–++)	(–++)	(+++)	(+++)	(–++)	(+++)	(+++)	(+++)	(+++)
	Bi-P06	*3,416*	*30*	*240*	3,416	31	255	3,410	35	260	3,410	36	265
		(+++)	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-P07	*2,849*	24	*150*	2,806	27	160	2,660	28	180	2,660	29	190
		(+++)	*(– – +)*	*(+++)*	(+++)	*(– – +)*	(+++)	(+++)	*(– – +)*	(+++)	(+++)	(+++)	(+++)
Large scale	P08	*2.6E+05*	*108*	*1,708*	2.60E+05	128	1,752	2.41E+05	129	1,801	2.58E+05	130	1,825
		(+++)	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-PL1	*9,260*	41	*165*	9,130	52	185	9,104	56	201	8,913	63	229
		(+++)	(+++)	*(+++)*	(+++)	*(+–+)*	(+++)	(+++)	*(+–+)*	(+++)	(+++)	(+++)	(+++)
	Bi-PL2	*11,929*	*95*	*1,562*	*11,010*	112	1,892	10,052	128	1,986	9,547	150	2,560
		(+++)	*(+++)*	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-PL3	*28,950*	*110*	*1,895*	*28,856*	*150*	2,058	28,125	160	2,185	20,012	198	2,986
		(–++)	*(+++)*	*(+++)*	*(–++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-PL4	*55,203*	*195*	*2,986*	*55,000*	210	3,698	53,256	285	3,560	48,596	410	4,523
		(+++)	*(+++)*	*(+++)*	*(+++)*	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)
	Bi-PL5	*120,589*	*526*	*5,123*	110,585	522	5,362	102,589	632	7,525	108,952	699	6,121
		(+++)	*(–++)*	*(+++)*	(+++)	(–++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)	(+++)

Note: The best values are highlighted in ital.

6 Conclusions

In this paper, we have proposed a co-evolutionary decomposition-based algorithm-II to tackle BKPs. The statistical results have proven the ability of CODBA-II to generate interesting solutions on the bi-level KP problem. Thus, it would be interesting to propose a bi-level formulation of other combinatorial bi-level optimisation problems and evaluate the performance of CODBA-II on these problems. Indeed, we plan to design a cooperative behaviour between the two levels to improve the quality of generated solutions.

References

Bard, J.F. and Falk, J.E. (1985) 'An explicit solution to the multi-level programming problem', *Computers & Operations Research*, Vol. 9, No. 1, pp.77–100.

Brotcorne, L., Hanafi, S. and Mansi, R. (2009) 'A dynamic programming algorithm for the bilevel knapsack problem', *Operations Research Letters*, Vol. 37, No. 3, pp.215–218.

Chaabani, A., Bechikh, S. and Ben Said, L. (2015a) 'A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization', *IEEE Congress on Evolutionary Computation (CEC)*, pp.1659–1666.

Chaabani, A., Bechikh, S. and Ben Said, L. (2015b) 'An improved co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization', *Annual Conference on Genetic and Evolutionary Computation (GECCO)*, ACM, pp.1363–1364.

Chaabani, A., Bechikh, S. and Ben Said, L. (2016) 'A memetic evolutionary algorithm for bi-level combinatorial optimization: a realization between Bi-MDVRP and Bi-CVRP', *IEEE Congress on Evolutionary Computation (CEC)*, pp.1666–1673.

Chaabani, A., Bechikh, S. and Ben Said, L. (2017a) 'A co-evolutionary decomposition-based chemical reaction algorithm for bi-level combinatorial optimization problems', *Knowledge-Based and Intelligent Information & Engineering Systems (KES)*, Vol. 48 , No. 9, pp.780–789.

Chaabani, A., Bechikh, S. and Ben Said, L. (2017b) 'A new co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization', *Applied Intelligence*, Vol. 48, No. 9, pp.2847–2872, Springer.

Chen, L. and Zhang, G. (2013) 'Approximation algorithms for a bi-level knapsack problem', *Theoretical Computer Science*, Vol. 497, pp.1–12.

Dempe, S. and Richter, K. (2000) *Bilevel Programming with Knapsack Constraints*, TU Bergakademie, Fakultat fur Mathematik und Informatik.

Dempe, S. (2002) *Foundations of Bilevel Programming*, Springer Science and Business Media.

Dempe, S. (2003) 'Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints'.

DeNegre, S. (2011) *Interdiction and Discrete Bilevel Linear Programming*, Thesis, Lehigh University.

Derrac, J., Garcia, S., Molina, D. and Herrera, F. (2011) 'A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms', *Swarm Evolutionary Computation*, Vol. 1, No. 1, pp.3–18.

Eiben, A.E. and Smit, S.K. (2011) 'Parameter tuning for configuring and analyzing evolutionary algorithms', *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp.19–31.

Jeroslow, R.G. (1985) 'The polynomial hierarchy and a simple model for competitive analysis', *J. Mathematical Programming*, Vol. 32, No. 2, pp.146–164.

Kuhn, H. and Tucker, A. (1951) 'Non linear programming', *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, BerThley, University of California, pp.481–492.

Legillon, F., Liefooghe, A. and Talbi, E.G. (2012) 'COBRA: a cooperative coevolutionary algorithm for bi-level optimization', *IEEE Congress on Evolutionary Computation*, pp.1–8.

Mansi, R., Alves, C., Valerio de Carvalho, J.M. and Hanafi, S. (2012) 'An exact algorithm for bilevel knapsack problems', *Mathematical Problems in Engineering*, Vol. 2012.

Marinakis, Y., Migdalas, A. and Pardalos, P.M. (2007) 'A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm', *Journal of Global Optimization*, Vol. 38, No. 4, pp.555–580.

Martello, S. (1990) *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Interscience Series in Discrete Mathematics and Optimization.

Savard, G. and Gauvin, J. (1994) 'The steepest descent direction for the nonlinear bilevel programming problem', *Operations Research Letters*, Vol. 15, No. 5, pp.265–272.

Shepherd, S. and Sumalee, A. (2004) 'A genetic algorithm based approach to optimal toll level and location problems', *Networks and Spatial Economics*, Vol. 4, No. 2, pp.161–179.

Sinha, A., Malo, P. and Deb, K. (2013) 'Efficient evolutionary algorithm for single-objective bilevel optimization', *arXiv preprint arXiv:1303.3901*.

Sinha, A., Malo, P. and Deb, K. (2017) *Evolutionary Bilevel Optimization: An Introduction and Recent Advances*, pp.71–103, Springer.

Talbi, E-G. (2013) *Metaheuristics for Bi-level Optimization*, Vol. 482, Springer, DOI: 10.1007/978-3-642-37838-6.

Wen, U.P. and Hsu, S.T. (1991) 'Linear bi-level programming problems – a review', *Journal of the Operational Research Society*, Vol. 42, No. 2, pp.125–133.