INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res 00 (2015) 1–22
DOI: 10.1111/itor.12153

A study on the use of heuristics to solve a bilevel programming problem

Jaqueline S. Angelo and Helio J. C. Barbosa

Laboratório Nacional de Computação Científica, Petrópolis-RJ, Brazil
E-mail: jsangelo@lncc.br [Angelo]; hcbm@lncc.br [Barbosa]

Received 3 January 2014; received in revised form 25 June 2014; accepted 14 December 2014

Abstract

The bilevel programming problem is characterized as an optimization problem that has another optimization problem in its constraints. The leader in the upper level and the follower in the lower level are hierarchically related where the leader's decisions affect both the follower's payoff function and allowable actions, and vice versa. One difficulty that arises in solving bilevel problems is that unless a solution is optimal for the lower level problem, it cannot be feasible for the overall problem. This suggests that approximate methods could not be used for solving the lower level problem, as they do not guarantee that the optimal solution is actually found. However, from the practical point of view near-optimal solutions are often acceptable, especially when the lower level problem is too costly to be exactly solved thus rendering the use of exact methods impractical. In this paper, we study the impact of using an approximate method in the lower level problem, discussing how near-optimal solutions on the lower level can affect the upper level objective function values. This study considers a bilevel production-distribution planning problem that is solved by two intelligent heuristics hierarchically related: ant colony optimization for solving the upper level problem, and differential evolution method to solve the lower level problem.

Keywords: bilevel optimization; ant-colony optimization; differential evolution

1. Introduction

Bilevel programming problems (BLP) are a challenging class of optimization problems which have two optimization problems hierarchically related. One can think of a leader—in the upper level—who seeks to optimize an objective function considering certain constraints and the reaction of the follower—in the lower level—who, in turn, makes his/her decision in order to optimize his/her objective function taking into account what has been imposed by the leader.

Many practical problems have been modeled as bilevel problems that appear in areas such as economics: in Stackelberg games, Cournot–Nash equilibria, principal agent problem (Dempe, 2002) and environmental economics (Sinha et al., 2013); engineering: in optimal design (Barbosa, 1997;

Friedlander and Gomes, 2011) and optimal chemical equilibria (Dempe, 2002); transportation: in network design (Costantino et al., 2013) and optimal pricing (Labbé et al., 1998; Pimentel and Fampa, 2012); management: in facility location (Huijun et al., 2008; Beresnev, 2013) and coordination of multidivisional firms (Bard, 1983); and others.

Bilevel problems are considered very difficult for most existing optimization approaches. Even bilevel problems, where all functions involved are linear, are in general difficult and were shown to be NP-hard (Ben-Ayed, 1990; Hansen et al., 1992). Moreover, even when both levels are convex programming problems, the resultant BLP can be nonconvex (Ben-Ayed, 1993). A variety of exact methods have been developed, involving the use of enumerative schemes, application of a penalty function, or gradient methods, among others. A common strategy used by researchers in handling such problems is to investigate the properties of the so-called inducible region to obtain (necessary and sufficient) conditions to replace the follower's problem with its Karush–Kunh–Tucker conditions, by appending the resultant system to the leader's problem and transforming the bilevel problem into a standard single-level programming problem. However, those approaches assume explicit knowledge of the analytical form of the objective function and the constraints of the problem and, sometimes, the transformed problem is not equivalent to the original bilevel problem.

Due to the complexity of bilevel problems, the use of intelligent heuristics methods becomes attractive because they can help overcome the many challenges of BLP, such as nonconvexity and nondifferentiability, large number of variables and/or constraints, nonunique optimal solution for the follower's problem, and large-scale problem instances. A considerable amount of work has already been published using evolutionary methods to solve different applications in bilevel optimization since they are able to provide satisfactory solutions in a reasonable time (Talbi, 2013). Ant-colony optimization (ACO; Baskan and Haldenbilen, 2011; Calvete et al., 2011), genetic algorithms (Marinakis et al., 2007; Deb and Sinha, 2009), differential evolution (DE; Segundo et al., 2012; Angelo et al., 2013), particle swarm optimization (Koh, 2009), coevolutionary methods (Legillon et al., 2012; Deb et al., 2013), and tabu search (Rajesh et al., 2003; Balakrishnan et al., 2013) are examples of metaheuristics proposed to solve bilevel problems.

In this paper, we address a bilevel transportation routing problem. The leader's problem concerns the routing process, where a fleet of vehicles is responsible for shipping products from a set of depots to a set of retailers. This problem is a multidepot vehicle routing problem, which will be solved by an ant colony based algorithm. In the lower level, the follower is responsible for producing commodities on a set of plants to meet the demands of the leader's depots. This last problem is modeled as a transportation problem (TP), and will be solved by a DE method. The bilevel algorithm proposed, named bilevel ACO + DE, makes use of those two intelligent heuristics hierarchically related to solve the bilevel transportation routing problem.

We intend to study the implications of using an approximate method, in this case the DE, to solve the lower level problem, discussing how near-optimal lower level solutions can affect the overall optimization process. The comparison will be made observing how the leader's objective function is affected by the solutions obtained from the approximate method, where the results generated by the DE are compared with the optimal solutions provided by an exact method.

It is important to mention that since the lower level problem treated here—the TP—can be solved to optimality, a direct quality measurement can be made comparing the approximate solutions, found by the DE, with the optimal solutions. When the optimal solution of the lower level problem

is not available, indicators to assess the quality of the solutions obtained have been suggested in Legillon et al. (2012).

The paper is organized as follows. Section 2 presents a brief overview of the optimization problem considered, the BLP; in Section 3, the bilevel transportation routing problem is introduced, which is used as a test problem to analyze the algorithm proposed; the bilevel method proposed, the bilevel ACO + DE, is described in Section 4, where the ant colony based algorithm and the DE method are applied to the problem to be solved. The computational experiments are described in Section 5, where the proposed algorithm is analyzed and discussed. Finally, the conclusions and suggestions for future work are given in Section 7.

2. The BLP

The BLP has its roots in two different domains. The first one is the domain of game theory that dates back to 1952, when the economist Stackelberg described oligopoly situations in economy markets. Stackelberg's model describes a competitive game between two companies—decision makers—that make decisions sequentially (Stackelberg, 1952). The other root comes from mathematical programming, described by Bracken and McGill (1973), where the problems appeared as an optimization problem that contains another optimization problem in its constraints (Bracken and McGill, 1973).

In the BLP, two decision makers, the leader in the upper level and the follower in the lower level, are hierarchically related, where the leader's decisions affect both the follower's payoff function and allowable actions, and vice versa. Each decision maker has control over a set of variables, seeking to optimize its own objective function. Once the leader chooses the value of its variables, the follower reacts by choosing the value of its own variables in order to optimize its objective function.

A BLP can be defined as

$$\begin{array}{lll} (L) & \min\limits_{x \in X} & f_1(x, y(x)) \\ & \text{subject to} & g_1(x, y(x)) \leq 0 \\ & (F) \quad y(x) \in R(x) := \arg\min\limits_{y \in Y} & f_2(x, y) \\ & \text{subject to} & g_2(x, y) \leq 0, \end{array} \tag{1}$$

where $f_1(x, y(x))/f_2(x, y)$, and $g_1(x, y(x))/g_2(x, y)$ are the upper/lower level objective function and upper/lower level constraints.

The upper level decision maker (L)—the leader—has control over the x variables, and makes his decision first, fixing x. The lower level decision maker (F)—the follower—controls the y variables. Reacting to the decision of the leader, the y variables are set in response to the given x.

In problem (1) the reaction set of the follower, namely $R(x)$ such that $y(x) \in R(x)$, defines the follower's response given an x fixed by the leader. One difficulty that arises in solving BLP is that, if $R(x)$ is not single-valued for all possible x, the leader may not achieve his minimum payoff, since the follower has multiple minimum solutions to choose. In this case, there is no guarantee that the follower's choice is the best for the leader, leading to sub-optimal solutions in the leader's problem. To overcome this situation at least two approaches can be considered, the optimistic one and the pessimistic one.

In the optimistic case, the leader assumes that the follower is willing to support him, that is, that the follower will select a solution $y(x) \in R(x)$ which is the best from the leader's point of view. This results in the so-called "optimistic" or "weak bilevel problem" (Dempe, 2002):

$$\begin{array}{rll} (L) & \min\limits_{x\in X}\ \min\limits_{y\in R(x)} & f_1(x, y(x)) \\ & \text{subject to} & g_1(x, y(x)) \leq 0 \\ & (F) & y(x) \in R(x) := \arg\min\limits_{y\in Y} f_2(x, y) \\ & & \text{subject to} \quad g_2(x, y) \leq 0. \end{array} \tag{2}$$

On the other hand, in the pessimistic case, the leader protects himself against the worst possible situation, leading to the so-called "pessimistic" or "strong bilevel problem" (Dempe, 2002):

$$\begin{array}{rll} (L) & \min\limits_{x\in X}\ \max\limits_{y\in R(x)} & f_1(x, y(x)) \\ & \text{subject to} & g_1(x, y(x)) \leq 0 \\ & (F) & y(x) \in R(x) := \arg\min\limits_{y\in Y} f_2(x, y) \\ & & \text{subject to} \quad g_2(x, y) \leq 0. \end{array} \tag{3}$$

Another challenge lies in the fact that unless a solution is optimal for the lower level problem, it cannot be feasible for the overall problem. This suggests that approximate methods could not be used for solving the lower level problem, as they are not guaranteed to reach the optimal solution. However, the complexity of many bilevel applications makes the use of exact methods impractical.

3. The bilevel transportation routing problem

The bilevel transportation routing problem was first proposed in Calvete et al. (2011). The problem was described as a production–distribution planning problem involving two distinct decision makers. The first one, in the upper level of the hierarchy, controls the distribution of a single commodity between customers and depots, which is influenced by the behavior of the second decision maker. At the lower level of the hierarchy, the second decision maker controls the production process defining the amount of commodities to produce at each plant in order to meet the demands of the leader's depots and optimize its own objective function.

This bilevel problem involves two well-known problems: the vehicle routing problem with multiple depots—in the upper level—and the linear TP—in the lower level. Those two problems will be described here separately in order to clearly associate the different algorithms with each problem to be treated.

3.1. The multiple depot vehicle routing problem

The vehicle routing problem is a combinatorial optimization problem where one seeks to service a number of customers with a fleet of vehicles allocated at a single depot. As the name indicates, in the multiple depot vehicle routing problem (MDVRP) a set of depots is used. A fleet of homogeneous vehicles based at each depot is required to visit customers in order to fulfill the customers' demands.

Each vehicle has to start from and return to the same depot, visiting each customer exactly once. The objective of the problem is to minimize the total travel cost while serving all customers.

The MDVRP can be represented in a graph $G(V, E)$ where $V = \{v_1, \ldots, v_n\} \bigcup V_0$ is the set of nodes, composed by customers and depots, given by $V_0 = \{v_{n+1}, \ldots, v_m\}$, while $E = \{(i, j) : i, j \in V\}$ is the set of arcs that fully connect the nodes, except by connections between depots. To each customer i, with $i = 1, \ldots, n$, is associated a demand q_i and a service time st_i, and each vehicle s, with $s = 1, \ldots, S$, has the same capacity Q and maximum time duration T allowed for traversing a complete route. Each route R_s associated to a vehicle s, is defined by $R_s = \{d, p_1, \ldots, p_w, d\}$, with $d \in V_0$ and $p_1, \ldots, p_w \in V/V_0$. The cost information c_{ij} denotes the cost—distance—between customer i and j, or between a depot i and a customer j.

Defining the decision variables as

$$x_{ij}^s = \begin{cases} 1, & \text{if the vehicle's route } R_s \text{ uses the arc } (i, j) \text{ and} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

the MDVRP can be written as

$$\min \quad \sum_{s=1}^{S} \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij}^s \tag{5}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}^s t_i \leq T, \qquad s = 1, \ldots, S \tag{6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}^s q_i \leq Q, \qquad s = 1, \ldots, S \tag{7}$$

$$\sum_{i=1}^{m} x_{il}^s - \sum_{j=1}^{m} x_{lj}^s = 0, \quad s = 1, \ldots, S, l = 1, \ldots, m \tag{8}$$

$$x_{ij}^s \in \{0, 1\}, \qquad (i, j) \in E. \tag{9}$$

Hence, a feasible solution is composed by a set of routes $R_1, \ldots, R_S$, specifying the order of the visited customers. A route R_s is feasible if the vehicle does not exceed the maximum time duration of a route and its total capacity, as presented by constraints (6) and (7), respectively, while visiting each customer exactly once, as specified by constraint (8).

3.2. The transportation problem

In the TP, one seeks to determine a minimum cost of transportation and allocation of resources from a number of sources (plants) to a number of destinations (depots). It is assumed that the products can be transported conveniently from all sources to all destinations and the cost of transportation on a given route is directly proportional to the number of units shipped on that route.

The TP can be written as

$$\min \quad \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \tag{10}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} y_{ij} = S_i \quad \text{for } i = 1, \ldots, N \tag{11}$$

$$\sum_{i=1}^{N} y_{ij} = D_j \quad \text{for } j = 1, \ldots, M \tag{12}$$

$$y_{ij} \in \mathsf{N}, \tag{13}$$

where constraint (11) specifies that the sum of the shipments from a plant must be equal to its total supply, constraint (12) stipulates that the sum of the shipments to a destination must exactly satisfy its demand, and (13) assumes that the decision variables y_{ij} take on integer and nonnegative values. The transportation cost c_{ij} is associated to the cost of manufacturing an item at plant i for depot j. Here, we assume that the total supply is equal to the total demand, that is

$$\sum_{i=1}^{N} S_i = \sum_{i=1}^{M} D_j. \tag{14}$$

Hence, a feasible solution that satisfies all constraints means that the transportation plan exactly meets all demands at each depot and uses all of the supplies at each plant.

Although the TP considered here is linear and an exact solver is available, the DE adopted can also be used for solving a nonlinear version of the problem.

3.3. The bilevel problem formulation

The bilevel formulation of the transportation routing problem is described as follows:

$$\begin{aligned}
(L) \quad & \min_{x_{ij}^s, y_{nl}} \quad \sum_{s \in S} \sum_{(i,j) \in E} c_{ij}^1 x_{ij}^s + \sum_{n \in N} \sum_{l \in M} c_{nl}^3 y_{nl} \\
& \text{s.t.} \quad x_{ij}^s \text{ verifies the MDVRP constraints} \\
& \qquad y_{nl} \text{ solves } (F) \\
(F) \quad & \min_{y_{nl}} \sum_{n \in N} \sum_{l \in M} c_{nl}^2 y_{nl} \\
& \text{s.t} \quad y_{nl} \text{ verifies the TP constraints.}
\end{aligned} \tag{15}$$

In this problem, a distribution company—at the upper level—owns a fleet of homogeneous vehicles and a set of depots, in order to serve its customers while minimizing its operational costs. Each vehicle, based at a different depot, will serve a set of customers in order to satisfy their demands. The cost of the distribution company involves the cost of sending an item from customer i to customer

j using vehicle s ($c^1_{ij}x^s_{ij}$) plus the cost of acquiring and shipping an item manufactured in plant n to supply the depot l ($c^3_{nl}y_{nl}$).

A production company—at the lower level—receives the orders from the distribution company informing the demand of each depot. Knowing the demands of each depot, the production company, which owns a set of plants, needs to produce commodities to satisfy the depots' demands aiming at minimizing its own operational costs. The cost involved in producing a commodity required for depot l depends on the allocation of production to plants ($c^2_{nl}y_{nl}$). In this way, the distribution company can influence, but cannot control, the decisions of the production company. Hence, each candidate solution x^s_{ij} generated for the MDVRP will provide a demand order $D = \{d_1, \ldots, d_M\}$ corresponding to the demands of each depot. Those orders are passed to the production company so as to provide the best possible arrangement of the amount of commodities necessary to be manufactured to supply all the depots' demands.

4. Description of the solution method proposed

The bilevel algorithm proposed uses two nested heuristics to solve the bilevel transportation routing problem. The method uses an ant colony algorithm to solve the upper level problem, which is associated with the MDVRP, while the lower level problem is solved by a DE algorithm addressed to the TP. Both algorithms will be separately described as they are applied to the problems considered here, and then the bilevel ACO + DE algorithm proposed will be presented.

4.1. Upper level optimization—ACO

ACO is a metaheuristic inspired by the collective behavior of real ant colonies that is used for solving optimization problems in general and, in particular, those that can be formulated as finding good paths through graphs. In ACO, a set of agents (artificial ants) cooperate in trying to find good solutions to the problem at hand (Dorigo and Stützle, 2004).

In ACO algorithms, artificial ants cooperate while exploring the search space, searching good solutions for the problem through a communication mediated by artificial pheromone trails. The solution construction process is incremental, where a solution is built by adding solution components to an initially empty candidate solution. The ant's heuristic rule probabilistically decides the next solution component guided by (a) the heuristic information (η), representing *a priori* information about the problem instance to be solved; and (b) the pheromone trail (τ), which encodes a memory about the ant colony search process that is continuously updated by the ants according to the quality of the solutions constructed.

The proposed ACO algorithm designed to solve the MDVRP is based on the ant colony system (Dorigo and Gambardella, 1997). Each ant has to construct a complete solution to the problem, that is, a set of routes to visit customers exactly once, starting from a depot and ending at the same depot. Hence, beginning at a random depot, the action choice rule adopted by each ant to choose the next node j—customer—to be visited is given by

$$j = \begin{cases} \arg\max\limits_{j \in \mathcal{N}_i^h} \left[\tau_{ij}^{\alpha}\right]\left[\eta_{ij}^{\beta}\right], & \text{if } q \leq q_0 \\ p_{ij}, & \text{otherwise,} \end{cases} \tag{16}$$

where

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i^h}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, & \text{if } j \in \mathcal{N}_i^h \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The pheromone information τ_{ij} indicates the attractiveness of choosing customer j, when the ant is at node i, which can be a depot or a customer, while the heuristic information $\eta_{ij} = 1/c_{ij}^1$ indicates that the heuristic desirability of going from node i to node j is inversely proportional to the distance between the two nodes. The parameter $q_0 \in [0, 1]$ chosen by the user, and a random variable $q \in [0, 1]$, determine if the next node to be visited is chosen deterministically or probabilistically, as given in (16) and (17), respectively. Note that an ant chooses with probability q_0 the best value as indicated in the pheromone trails and heuristic information, intensifying the search, and performs a diversification in the search space with probability $(1 - q_0)$. Parameters α and β define the relative importance of the pheromone trail and heuristic information, respectively, and $\mathcal{N}_i^h$ is the set of customers that have not yet been visited by ant h. This construction process continues until all customers are visited or until any constraint—(6) or (7)—is violated, in which case the ant returns to the depot it came from.

Every time an ant moves from one node to another a local pheromone update is performed to increase the exploration of alternative paths, as follows:

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0, \tag{18}$$

where $\xi \in (0, 1)$ is the local pheromone decay coefficient, and τ_0 is the initial value of the pheromone trails.

The global pheromone update is performed at the end of each iteration, that is, after all ants construct a complete solution to the problem. Only the best-so-far ant is allowed to update the global pheromone. This is done by both evaporation and new pheromone deposit on the arcs crossed by the best-so-far ant:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{\text{best}}, \tag{19}$$

where $\rho \in (0, 1]$ is the pheromone evaporation rate and $\Delta\tau_{ij}^{\text{best}}$ is the amount of pheromone deposited by the ant that generates the best solution—best-so-far ant—according to the upper level objective function value.

4.2. Lower level optimization—DE

DE is a simple stochastic population based algorithm shown to be an effective method for global continuous optimization (Storn and Price, 1997). Another reason for its popularity is the reduced number of control parameters required.

The basic DE algorithm begins with a random generation of candidate solutions or individuals, represented by vectors, where the set of these individuals represent a population. The evolution of the individuals is given by successive generations, where DE performs mutations and recombinations in the population in order to generate a new population, that is, the next generation of individuals.

The basic operation performed is the addition to each design variable in a given candidate solution of a term, which is the scaled difference between the values of such variable and other candidate solution in the population. The number of differences applied, the way in which the individuals are selected, and the distribution of recombination determine the DE variant. Like other evolutionary algorithms, DE performance is influenced by its particular parameters, namely, the chosen variant.

Here, the DE variant DE/target-to-rand/1/ was adopted, with no crossover operation. This variant uses three distinct randomly selected individuals ($y^{r_1}, y^{r_2}, y^{r_3}$) and the target individual y^k of the population (the one that will be used in the comparison after the mutation, also called target individual), leading to

$$u_{ij} = y_{ij}^k + F.\left(y_{ij}^{r_1} - y_{ij}^k\right) + F.\left(y_{ij}^{r_2} - y_{ij}^{r_3}\right), \tag{20}$$

where u is the new individual, with $i = 1, \ldots, N$ and $j = 1, \ldots, M$ and F is the scale factor applied. If the new individual generated u is better than the target individual y^k, it replaces the target individual in the next generation; otherwise the target retains its place in the population.

The initial population is composed of solutions that satisfy all TP constraints, as proposed in Michalewicz (1996). With this initialization procedure the DE maintains the linear constraints (11) and (12) for every value of $F \in (0, 1]$. However, when real values are set to F, the integrality constraint (13) is violated. Hence, we adopt $F = 1$ to keep all design variables integer.

In addition, for each design variable, lower and upper bounds are assumed—as defined by constraints (11)–(13)—and whenever a given component u_{ij} of a new candidate solution u is generated outside this prescribed range, a standard projection operation is performed:

$$\text{If } u_{ij} > S_i \text{ or } u_{ij} > D_j \text{ or } u_{ij} < 0 \text{ then } u_{ij} = 0. \tag{21}$$

As a consequence, this projection operation makes the candidate solution infeasible as

$$\sum_{j=1}^{M} u_{ij} < S_i \quad \text{for } i = 1, \ldots, N \quad \text{and/or} \tag{22}$$

$$\sum_{i=1}^{N} u_{ij} < D_j \quad \text{for } j = 1, \ldots, M, \tag{23}$$

and so it is necessary to make changes in the individual in order to complete the solution with the missing values so as to satisfy all constraints.

4.3. Termination criteria

The ACO algorithm—upper level optimization—stops when a maximum number of iterations $it_{\max}$ is reached. The DE—lower level optimization—stops when a maximum number of generations $gen_{\max}$ is reached or until no better solution is found for a certain number of generations *gen*. The *gen* value varies according to the number of iterations of the upper level algorithm, where *gen* increases as the number of ACO iterations increase.

In the first ACO iteration, $l_{\max}$ and $l_{\min}$ are set, which are the maximum and the minimum number of generations, respectively, the DE continues until no better solution is found. Then, an interval is

calculated as $\mu = (l_{\max} - l_{\min})/it_{\max}$. At each ACO iteration, an auxiliary variable is calculated as $aux^{(t+1)} = aux^{(t)} + \mu$, with $aux^{(0)} = 0$. Then, the *gen* value is computed as

$$gen = l_{\min} + \lfloor aux \rfloor. \tag{24}$$

This procedure makes the *gen* value to vary from $l_{\min}$ to $l_{\max}$ as the number of ACO iterations increases.

4.4. The bilevel ACO + DE method

A pseudo-code of the proposed bilevel method is described in Algorithm 1. The main steps of the algorithm are summarized as follows.

Initiate data: The algorithm starts by initializing the data, where the bilevel instance is read and the parameters from both algorithms are set. In the initialization phase, a feasible initial solution of problem (15) is generated.

Generate initial solution: A initial feasible solution $f_{initial}$ is constructed for the bilevel problem, where a greedy heuristic is applied to construct a solution to the MDVRP, while the DE algorithm is used for solving the TP, corresponding to the routes obtained.

Initiate pheromone values: Hereafter, the pheromone trails τ_{ij} are initialized with the value τ_0, which is associated with the upper level function value of the initial solution $f_{initial}$.

Construct bilevel solutions: The bilevel solution construction starts by iteratively solving the upper and the lower level problems in this order, where for each solution generated in the upper level there is an associated solution in the lower level. Each ant will construct a set of routes that solves the MDVRP. Once the routes are constructed, they are improved by a 2-opt local search procedure (Hoos and Stutzle, 2004).

In addition to the global pheromone trail updating rule, a local pheromone update is applied so as to increase the exploration of new routes. The routes constructed provide the required amount of commodities in each depot according to the customers' demands. This information is passed to the lower level problem where the DE algorithm solves the TP. For each solution generated by an ant for the MDVRP, there is an associated solution for the TP. At this point, a feasible solution $\{x_{ij}, y_{nl}\}$ to the bilevel problem is obtained, and the leader's objective function value is computed.

Apply global pheromone update: After all ants have constructed a feasible solution to the bilevel problem (15), an elitist procedure is performed to select the best solution obtained according the leader's objective function value. Only this solution is allowed to update the global pheromone trails, that is, only the solution components of the best-so-far ant will receive an increment in the pheromone trails.

Finally, when the termination criteria are met, the algorithm returns the best solution found according to the upper level function value.

Algorithm 1: Bilevel ACO+DE Pseudo-code

```
// Initiate data
Read bilevel instance;
Initialize parameters for both ACO and DE algorithms;

// Generate initial solution
Solve the MDVRP, using a greedy heuristic;
Solve the TP, using DE, corresponding to the routes obtained;
Compute the leader objective function value f_initial;

// Initiate pheromone values
Set initial pheromone value τ_0, according to f_initial;
Initialize pheromone trails with τ_0;

while (not terminate) do
    // Construct Bilevel Solutions
    for a ← 0 to A_ants do
        Solve the MDVRP, using ACO, generating routes for the a-th ant;
        Apply local search procedure;
        Update local pheromone trails;
        Solve the TP, using DE, corresponding to the routes of the a-th ant;
        Compute the leader's objective function value of the a-th ant;

    // Apply global pheromone update
    Select the best-so-far ant a_bsf, according to the leader's objective function value;
    Update the global pheromone trail, associated with the a_bsf solution;

Return the best solution value from the leader;
```

5. Computational experiments

A computational study has been conducted in order to analyze how an approximate optimizer on the lower level problem can influence the optimization process of a bilevel problem. Hence, the tests are performed using an ant colony based algorithm to solve the upper level problem, and the DE to solve the lower level problem, where the DE results will be compared with an exact method named stepping stone method (SSM).[1]

The experiments tested the proposed algorithm on a set of bilevel transportation routing problems created based on MDVRP benchmark instances available at http://neo.lcc.uma.es/vrp/. The 10 original MDVRP instances were adapted in the same way as done in Calvete et al. (2011), in order to provide data for the TP. The number of customers in the MDVRP problems varies from 48 to 288 with four or six depots. The number of plants is set as the number of depots and the sum of the production availability of plants is equal to the sum of the total demands of the depots. The costs c^2_{nl} and c^3_{nl}—from Equation (15)—follow the same description given in Calvete et al. (2011). Table 1 describes the bilevel instances used for the computational experiments.

[1]The description of the SSM is available at http://orms.pef.czu.cz/text/transProblem.html. The Northwest Corner method was used for obtaining the initial feasible solution.

Table 1
Description of the bilevel instances

Problem	Depots	Vehicles	Plants	Customers	T	Q
BiPR01	4	1	4	48	500	200
BiPR02	4	2	4	96	480	195
BiPR03	4	3	4	144	460	190
BiPR04	4	4	4	192	440	185
BiPR05	4	5	4	240	420	180
BiPR06	4	6	4	288	400	175
BiPR07	6	1	6	72	500	200
BiPR08	6	2	6	144	475	190
BiPR09	6	3	6	216	450	180
BiPR10	6	4	6	288	425	170

T is the maximum duration of a route and Q is the capacity of a vehicle.

5.1. Parameter settings

The bilevel ACO + DE algorithm was executed 10 times for each bilevel instance. Some preliminary experiments were performed to find reasonable values for the user defined parameters. The parameters were set as follows:

- For ACO:
  - α and β: The relative importance of the pheromone trail and heuristic information are set to 1 and 2, respectively.
  - ρ and ξ: The global and local pheromone evaporation rates are both set to 0.1.
  - q_0: The parameter that regulate exploration and intensification of the search is set to 0.9.
  - A_{ants}: The number of ants is set to 40.
  - $it_{\max}$: The maximum number of iterations is set to 4000.
- For DE:
  - POP: The population size is set to 40.
  - F: The scale factor that controls the magnitude of the perturbation applied is set to 1.
  - $gen_{\max}$: The maximum number of generations is set to 1000.
  - gen, $l_{\max}$, and $l_{\min}$: The gen value indicates the number of generations DE continues until no better solution is found, which is set to a variable number going from $l_{\min} = 10$ to $l_{\max} = 100$ as the number of ACO iterations increases. The gen value is calculated according to Equation (24).

The experiments were conducted on a machine with 2 Intel Xeon E5440 Quad Core processors and 16 GB RAM.

5.2. Analysis of the results

The experiments analyze the results obtained by the proposed algorithm in order to compare the quality of the lower level solutions obtained by DE method against the optimal solutions obtained by SSM. We are interested in checking whether the DE method is able to generate good solutions to the follower's problem and how these solutions affect the objective function of the leader's problem.

The results obtained are compared with respect to the lower and the upper level objective function values, in order to verify:

- if the DE method was capable of generating optimal solutions;
- when DE is not capable to find the optimal solution, how close the solutions obtained are from the optimal ones;
- how many times DE finds optimal solutions;
- the number of lower level function evaluations;
- how the solutions obtained by DE affect the upper level solution value; and
- how the final solution generated is affected when DE is used for solving the lower level problem.

The plots in Figs. 1–10 present the results of one execution of the proposed method, which give information about how the solutions obtained by DE in the lower level affect the upper level solution values.

The plots at the left side of the figures refer to the follower's problem, presenting the percentage difference between the solution values obtained by DE against the optimal solutions obtained by SSM. The plots at the right side of the figures give information about how the leader's problem was affected by the solutions generated in the follower's problem using DE.

When DE does not find the optimal solution, the leader's objective function can be affected in two alternative ways: the objective function can be decreased, "improving" the solution value (as indicated by the negative values) or the objective function can be increased, worsening the solution

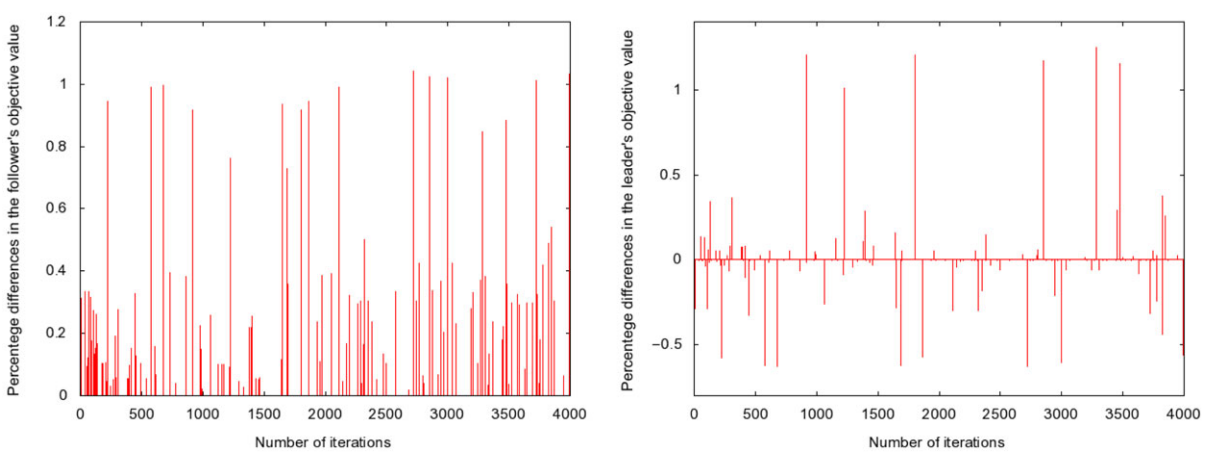

Fig. 1. Results from one execution, for instance BiPR01.

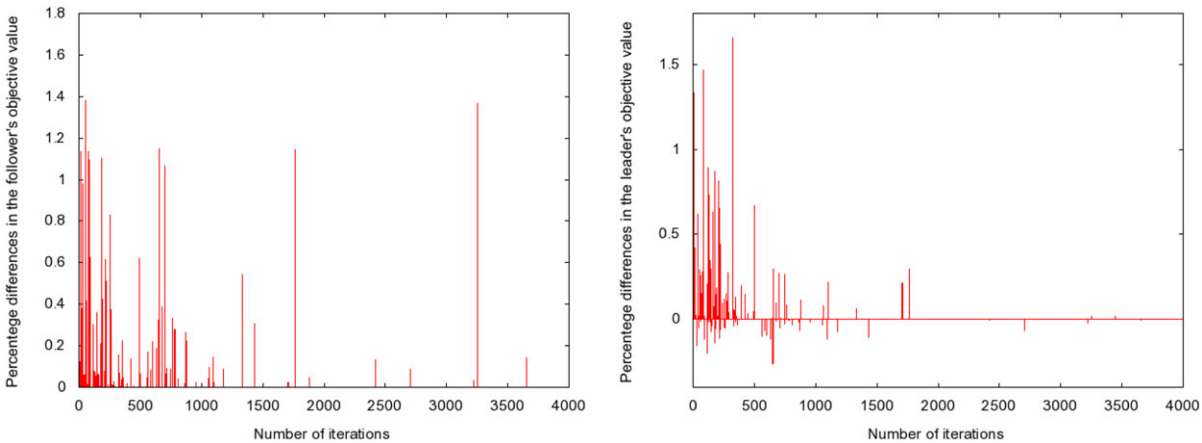

Fig. 2. Results from one execution, for instance BiPR02.

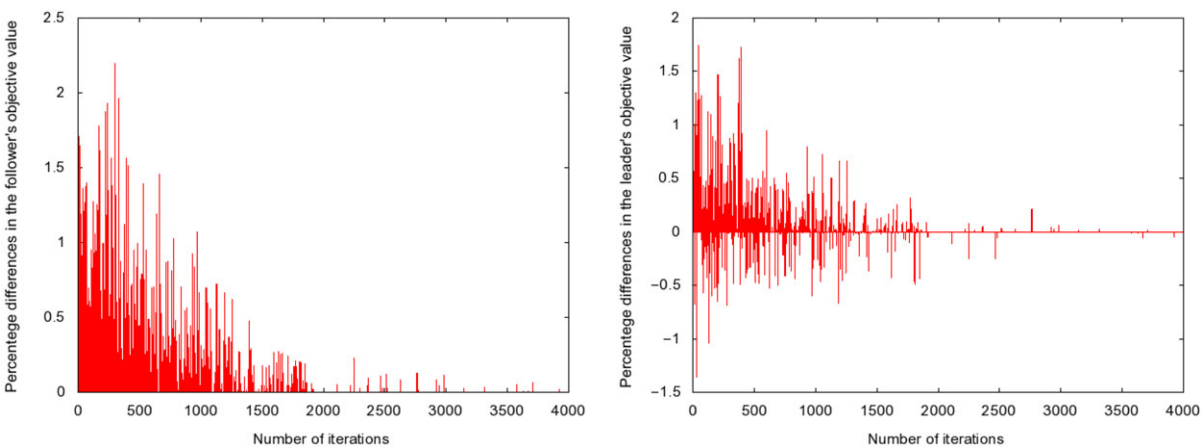

Fig. 3. Results from one execution, for instance BiPR03.

value (as indicated by the positive values). Clearly, when DE finds the optimal solution the leader's objective is not affected and no vertical bar is shown.

From the results presented in Figs. 1–10, it is possible to observe that for problems BiPR01–BiPR06 DE was capable of finding optimal solutions in many iterations of the algorithm. For problems BiPR07–BiPR10 the DE has some difficulty in finding optimal solutions. This observation can be validated by the values on Table 2, which inform the percentage of optimal solutions found by DE when all executions are considered.

For all instances, as the number of iterations increases, the DE can increasingly generate solutions closer to the optimal ones, as presented on the left side of Figs. 1–10. For example, on problem BiPR10 in the first iterations the solutions value obtained by DE have a percentage difference from the optimal solution of about 8%, decreasing this percentage to about 5% in the last iterations.

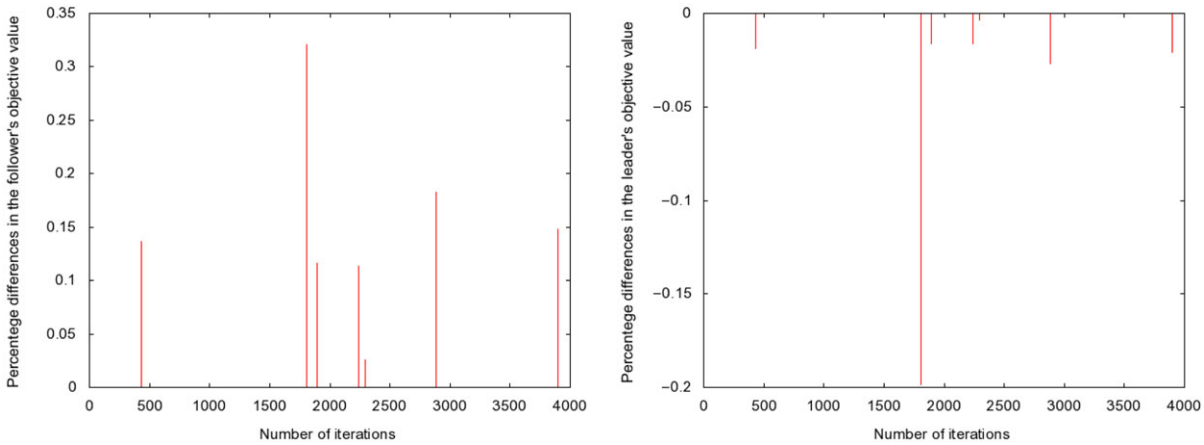

Fig. 4. Results from one execution, for instance BiPR04.

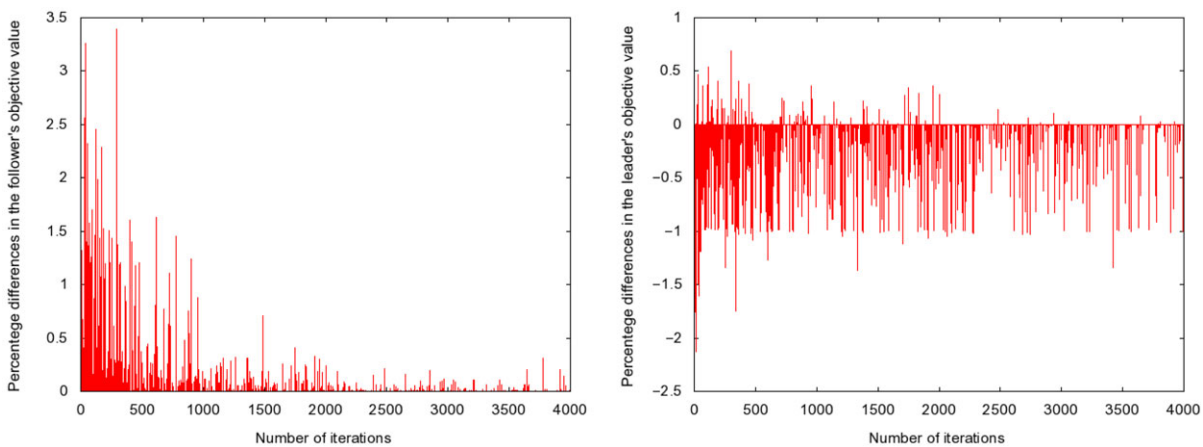

Fig. 5. Results from one execution, for instance BiPR05.

As shown in Table 2, from 10 runs, DE finds more than 90% of optimal solutions for problems BiPR01, BiPR02, and BiPR04; more than 80% of optimal solutions for problems BiPR03 and BiPR05; about 60% of optimal solutions for problem BiPR06; and less than 12% of optimal solutions for problems BiPR07–BiPR10. The average number of lower level function evaluations needed to obtain those results are shown in Fig. 11, and the average time consumed by the algorithm from 10 runs is shown in Fig. 12.

Table 3 presents the best and worst upper level solution values found by the proposed algorithm in 10 runs as well as the percentage differences in the objective functions in the lower level (LL) and upper level (UL).

In Table 3, for the test problems BiPR02 to BiPR06 the bilevel ACO + DE was capable of generating the best solution with the optimal lower level function value. For problems BiPR01 and BiPR7–BiPR10, DE was not capable to find the best solution with optimal lower level function value.

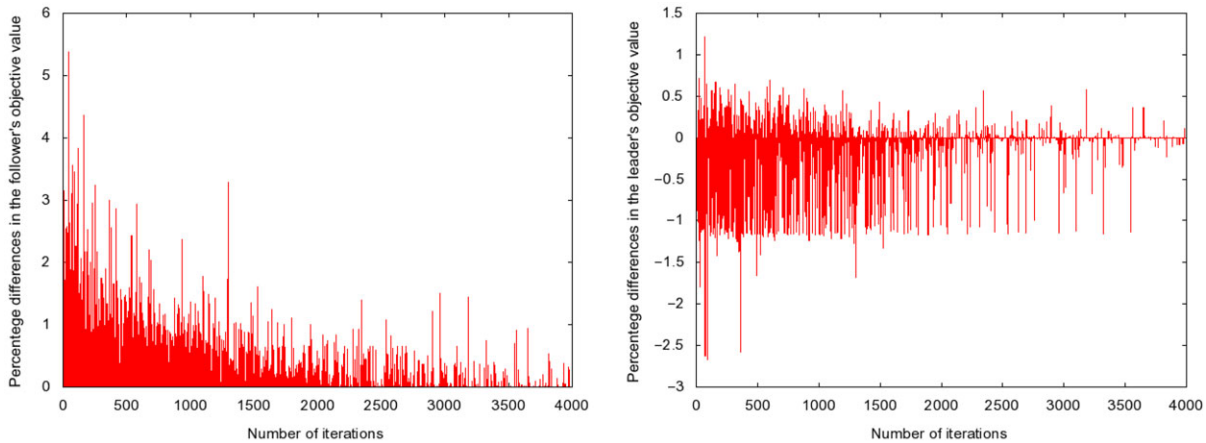

Fig. 6. Results from one execution, for instance BiPR06.

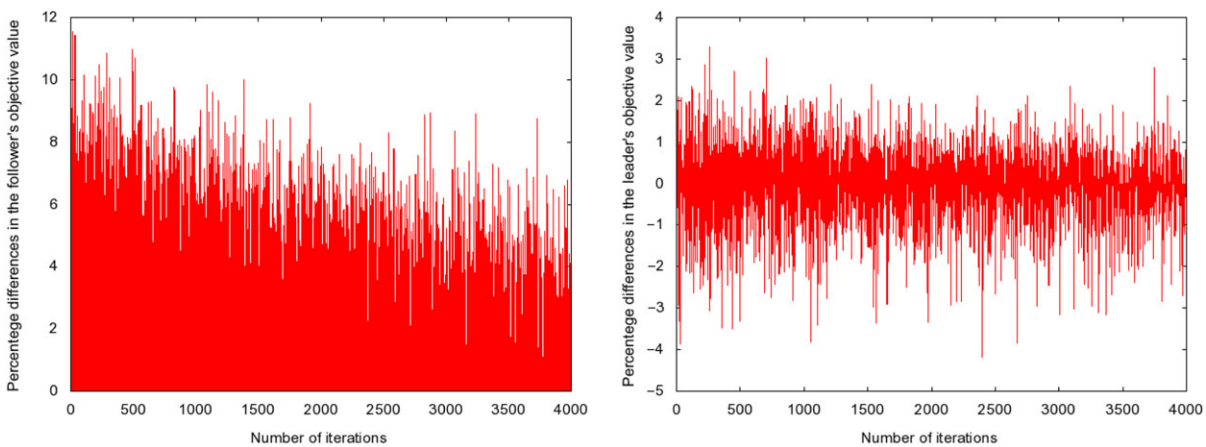

Fig. 7. Results from one execution, for instance BiPR07.

As a consequence, for those problems, the lower level problem was not solved to optimality, hence generating infeasible final solutions. A similar behavior occurs with the worst solutions obtained, where for problems BiPR02, BiPR04, and BiPR05 the algorithm was capable of generating feasible solutions.

The final results obtained by the proposed method were compared with the same ACO algorithm, used for solving the leader's problem, with the exact SSM to solve the follower's problem, as presented in Table 4. The values in bold indicate the best solutions obtained between the two methods, and values with an asterisk indicate that the solution has the optimal value for the lower level problem.

From Table 4, we conclude that bilevel ACO + DE finds better solutions for problems BiPR01, BiPR04, BiPR05, BiPR07, BiPR08, and BiPR09 when compared to the ACO algorithm using the SSM to solve the follower's problem. However, from those solutions only the solutions obtained for problems BiPR04 and BiPR05 generate the best solution with the optimal value in the lower level

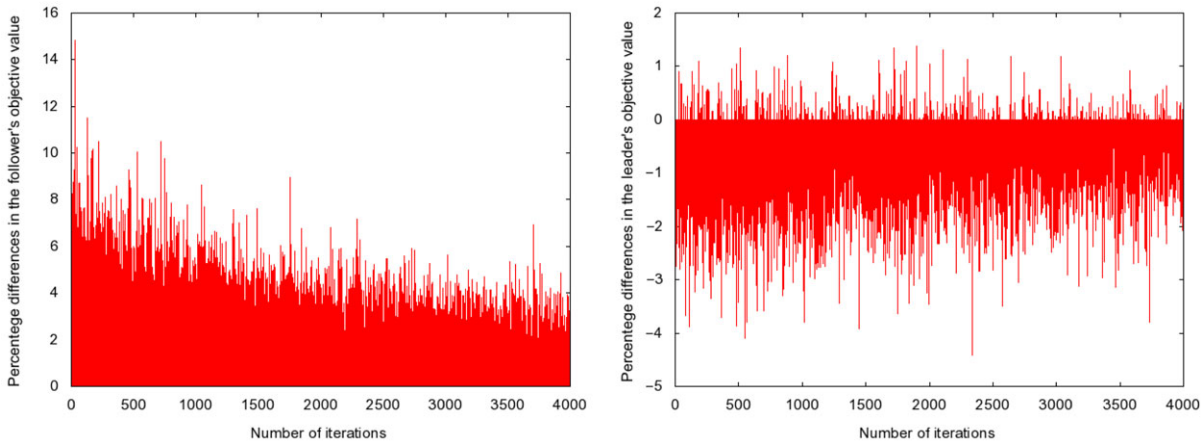

Fig. 8. Results from one execution, for instance BiPR08.

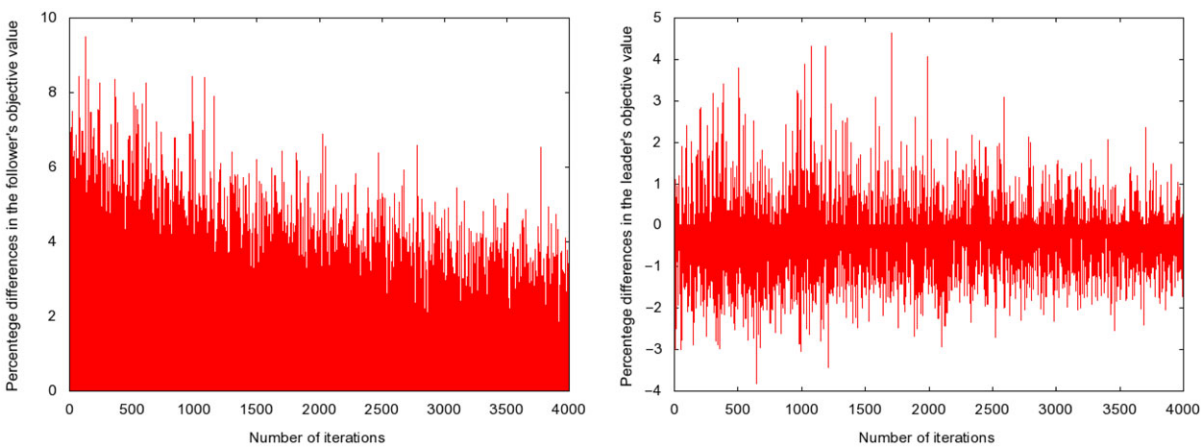

Fig. 9. Results from one execution, for instance BiPR09.

problem, as can be seen in Table 3. It is of interest to note that, although the DE solutions were not guaranteed to be optimal, bilevel ACO + DE finds better solutions for two problems (BiPR04 and BiPR05) with the optimal lower level value, when compared with the bilevel ACO with SSM.

6. Discussion

The computational results show how the proposed hybrid ACO + DE algorithm was capable of finding optimal solutions when solving the bilevel transportation routing problem, indicating how the near-optimal solutions obtained by the DE on the follower's problem affected the overall optimization process.

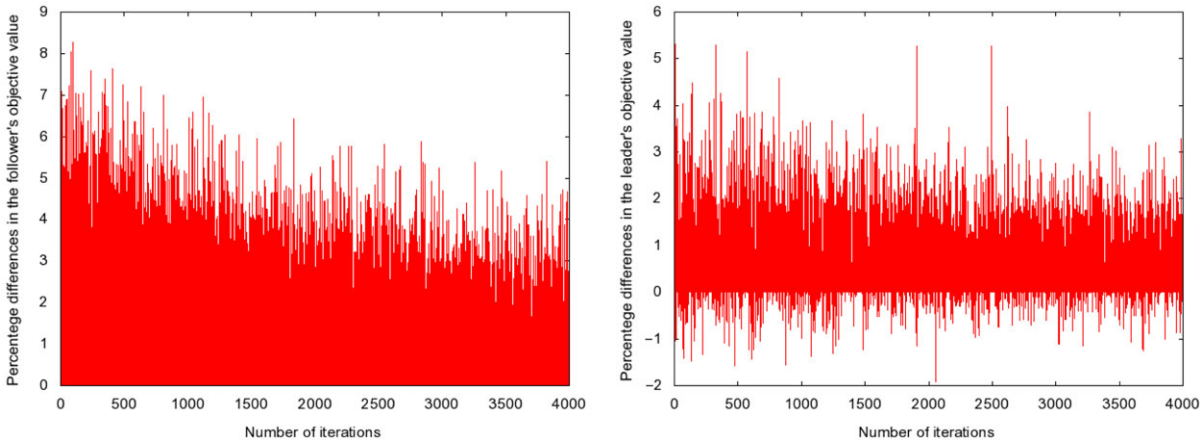

Fig. 10. Results from one execution, for instance BiPR10.

Table 2
Best, median, and worst percentage of optimal solutions found by DE in 10 runs

Problem	Best	Median	Worst
BiPR01	96.50	95.30	94.55
BiPR02	97.08	95.95	95.13
BiPR03	82.75	81.31	80.35
BiPR04	99.98	99.86	99.70
BiPR05	83.03	81.45	80.08
BiPR06	60.05	58.94	57.88
BiPR07	11.45	7.96	6.55
BiPR08	3.10	2.56	1.73
BiPR09	1.93	1.55	1.28
BiPR10	2.10	1.78	1.53

In our first analysis, the results obtained by DE were analyzed on the view of the upper and lower level objective functions. The number of generations allowed to the DE to solve the lower level problem was increased as the generation counter of the upper level optimization increased. By doing so, it was possible to see that as the number of generations increased, DE increasingly found better solutions, closer to the optimal ones, obtaining more than 80% of optimal solutions in problems BiPR01–BiPR05.

This experiment indicates that if an approximate method is used for solving the follower's problems, a large number of generations are required in order to obtain optimal solutions or at least "good" solutions. On the other hand, as a result, a large amount of lower level function evaluations is performed, as presented in Fig. 11. However, if an approximate algorithm in the follower's problem is executed for only a few generations to save computation time, that may cause the resultant solutions to be infeasible, as occurred in problems BiPR01, and BiPR07–BiPR10, indicated in Table 3.

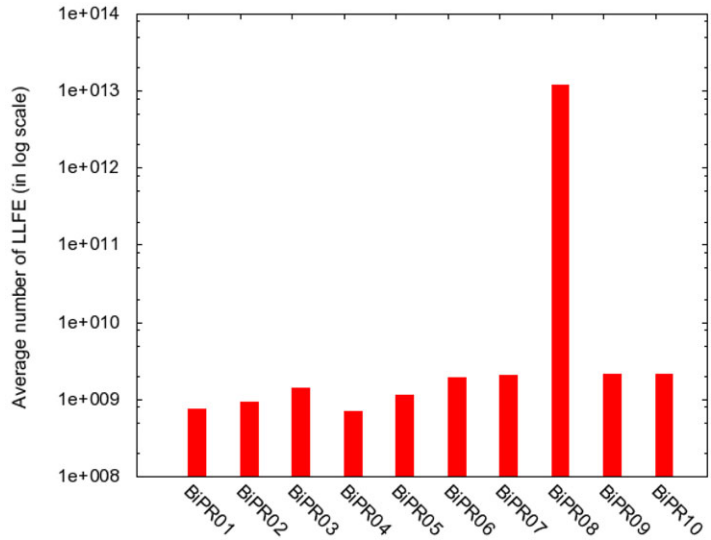

Fig. 11. Average number of lower level function evaluations in 10 runs required by bilevel ACO + DE when solving the bilevel problems.

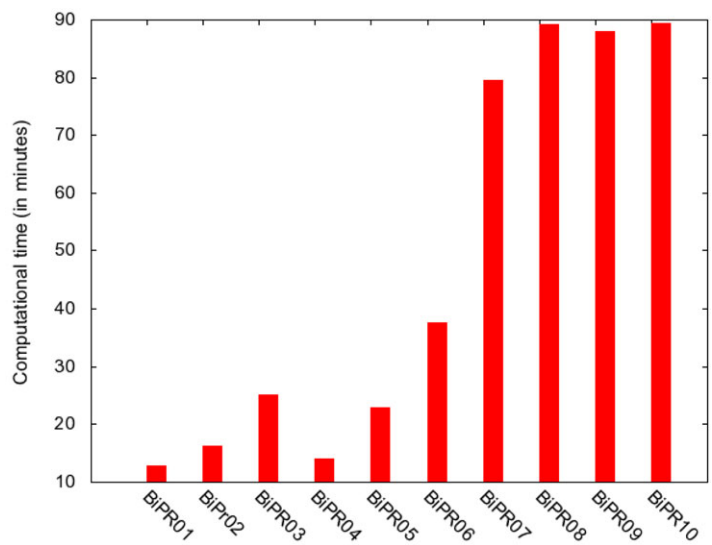

Fig. 12. Average computational time from 10 runs required by bilevel ACO + DE when solving the bilevel problems.

Hence, when solving a BLP with an approximate algorithm on the lower level, it is strongly recommended that the algorithm should evolve for a sufficient number of generations in order to obtain a good, thus nearly feasible, solution to the bilevel problem, although this may substantially increase the computational effort on calculating the lower level solution.

It is important to mention that, from pilot testing, bilevel ACO + DE was capable of generating the same final results using much less iterations than those reported. In fact, for some test problems,

Table 3
Best and worst upper level solution value found by bilevel ACO + DE in 10 runs, and the percentage differences in the lower (LL) and upper level (UL) objective functions

Problem	Best solution	Percentage in LL	Percentage in UL	Worst solution	Percentage in LL	Percentage in UL
BiPR01	1321.319	0.36	−0.01	1405.311	0.18	−0.19
BiPR02	2352.270	0	0	2440.190	0	0
BiPR03	3214.667	0	0	3428.235	0.20	−0.44
BiPR04	4325.061	0	0	4546.747	0	0
BiPR05	5209.667	0	0	5485.823	0	0
BiPR06	6452.972	0	0	6664.004	0.66	−1.17
BiPR07	1817.649	5.95	−3.88	1975.490	7.37	−3.20
BiPR08	3429.619	3.41	−1.32	3656.493	3.45	−2.18
BiPR09	4757.578	4.64	−2.03	5010.308	2.51	−1.26
BiPR10	6869.727	1.41	−0.03	7038.966	3.00	0.63

Table 4
Best, average, and worst upper level solution values found by ACO algorithm using the exact SSM at the lower level problem against the proposed bilevel ACO + DE algorithm from 10 runs

	Bilevel ACO with SSM			Bilevel ACO + DE		
Problem	Best	Average	Worst	Best	Average	Worst
BiPR01	1322.165	1356.549	1375.794	**1321.319**	1350.867	1405.311
BiPR02	**2287.229**	2358.057	2454.139	2352.270	2387.737	2440.190
BiPR03	**3188.317**	3294.063	3394.899	3214.667	3317.790	3428.235
BiPR04	4410.933	4500.867	4575.884	**4325.061***	4466.976	4546.747
BiPR05	5259.594	5353.814	5398.114	**5209.667***	5354.867	5485.823
BiPR06	**6386.031**	6550.050	6708.208	6452.972	6547.268	6664.004
BiPR07	1862.238	1919.826	1977.011	**1817.649**	1903.365	1975.490
BiPR08	3431.155	3602.554	3681.858	**3429.619**	3572.512	3656.493
BiPR09	4826.094	4946.248	5010.948	**4757.578**	4905.053	5010.308
BiPR10	**6675.745**	6884.343	7026.912	6869.727	6953.314	7038.966

with 2000 or 3000 iterations on the upper level optimization, the algorithm obtained the same final results.

Furthermore, since the algorithmic idea proposed used two algorithms independently, other evolutionary methods could be incorporated in the different levels of the BLP. The implementation of different methods in different levels of the BLP could be interesting when, for instance, the follower's problem can be efficiently solved by an existing problem-specific technique.

Nevertheless, there are some alternatives that can potentially decrease the number of function evaluations, consequently reducing the computational cost, such as the use of surrogate models, which replace the computationally intensive original objective function and constraint checking evaluation by a relatively inexpensive approximation; the selection of a small set of potentially "good" solutions on the upper level to submit to the lower level; the use of coevolution; and also adapting the algorithm parameters and closely monitoring the convergence process in the population in order to save function evaluations by stopping the iterations earlier.

7. Conclusions

Here we propose a hybrid ACO + DE technique for solving the bilevel transportation routing problem, and show that the methodology used for solving BLPs needs to be carefully designed so as not to produce misleading results.

We are particularly concerned with practical situations where metaheuristics are the only alternative for solving one or both levels of a BLP problem. When applied to the lower level problem in a nested procedure, good approximate solutions can be generated. However, such solutions, not being exact, are still rigorously infeasible with respect to the BLP, affecting the leader's solution. The termination criteria for the lower level problem in this case must be carefully designed in order to achieve an effective compromise between a good approximate solution for the lower level, and an acceptable total computer time for the BLP.

For future work, it is intended to extend the research to the case of multiple followers, and also when multiple objectives appear at the upper and/or lower level problems, where the design of effective and efficient techniques for such BLP problems using metaheuristics is an even greater challenge.

Acknowledgments

The authors thank the reviewers for their comments and suggestions, and acknowledge the support from CNPq (grants 141519/2010-0 and 310778/2013-1).

References

Angelo, J.S., Krempser, E., Barbosa, H.J.C., 2013. Differential evolution for bilevel programming. 2013 IEEE Congress on Evolutionary Computation, Cancún, México, pp. 470–477.

Balakrishnan, A., Banciu, M., Glowacka, K., Mirchandani, P., 2013. Hierarchical approach for survivable network design. *European Journal of Operational Research* 225, 223–235.

Barbosa, H.J.C., 1997. A coevolutionary genetic algorithm for a game approach to structural optimization. International Conference on Genetic Algorithms, ICGA, East Lansing, Michigan, pp. 545–552.

Bard, J.F., 1983. Coordination of a multidivisional organization through two levels of management. *Omega* 11, 5, 457–468.

Baskan, O., Haldenbilen, S., 2011. Ant colony optimization: methods and applications. In Ostfeld, A. (ed.) *Ant Colony Optimization Approach for Optimizing Traffic Signal Timings*. InTech, Rijeka, Croatia, pp. 205–220.

Ben-Ayed, O., 1990. Computational difficulties of bilevel linear programming. *Operations Research* 38, 3, 556–560.

Ben-Ayed, O., 1993. Bilevel linear programming. *Computers & Operations Research* 20, 5, 485–501.

Beresnev, V., 2013. Branch-and-bound algorithm for a competitive facility location problem. *Computers & Operations Research* 40, 2062–2070.

Bracken, J., McGill, J.T., 1973. Mathematical programs with optimization problems in the constraints. *Operations Research* 21, 1, 37–44.

Calvete, H.I., Galé, C., Oliveros, M., 2011. Bilevel model for production-distribution planning solved by using ant colony optimization. *Computers & Operations Research* 38, 1, 320–327.

Costantino, N., Dotoli, M., Falagario, M., Fanti, M.P., Mangini, A.M., Sciancalepore, F.B., Ukovich, W., 2013. A hierarchical optimization technique for the strategic design of distribution networks. *Computers & Industrial Engineering* 66, 849–864.

Deb, K., Gupta, S., Dutta, J., Ranjan, B., 2013. Solving dual problems using a coevolutionary optimization algorithm. *Journal of Global Optimization* 57, 3, 891–933.

Deb, K., Sinha, A., 2009. Solving bilevel multi-objective optimization problems using evolutionary algorithms. Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, EMO'09, Springer-Verlag, Berlin, Heidelberg, pp. 110–124.

Dempe, S., 2002. *Foundations of Bilevel Programming*. Kluwer Academic, Dordrecht.

Dorigo, M., Gambardella, L.M., 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1, 1, 53–66.

Dorigo, M., Stützle, T., 2004. *Ant Colony Optimization*. MIT Press, Cambridge, MA.

Friedlander, A., Gomes, F.A.M., 2011. Solution of a truss topology bilevel programming problem by means of an inexact restoration method. *Computational & Applied Mathematics* 30, 109–125.

Hansen, P., Jaumard, B., Savard, G., 1992. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* 13, 5, 1194–1217.

Hoos, H., Stutzle, T., 2004. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann, San Francisco, CA.

Huijun, S., Ziyou, G., Jianjun, W., 2008. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied Mathematical Modelling* 32, 610–616.

Koh, A., 2009. A coevolutionary particle swarm algorithm for bi-level variational inequalities: applications to competition in highway transportation networks. In Chiong, R., Dhakal, S. (eds) *Natural Intelligence for Scheduling, Planning and Packing Problems. Studies in Computational Intelligence*, Vol. 250. Springer, Berlin, pp. 195–217.

Labbé, M., Marcotte, P., Savard, G., 1998. A bilevel model of taxation and its application to optimal highway pricing. *Management Science* 44, 1608–1622.

Legillon, F., Liefooghe, A., Talbi, E., 2012. Cobra: a cooperative coevolutionary algorithm for bi-level optimization. IEEE Congress on Evolutionary Computation, Brisbane, Australia, pp. 1–8.

Marinakis, Y., Migdalas, A., 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* 38, 555–580.

Michalewicz, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, New York.

Pimentel, W., Fampa, M., 2012. A genetic algorithm to the strategic pricing problem in competitive electricity markets. Simpósio Brasileiro de Pesquisa Operacional, SBPO, Rio de Janeiro, Brazil, pp. 3684–3692.

Rajesh, J., Gupta, K., Kusumakar, H., Jayaraman, V.K., Kulkarni, B.D., 2003. A tabu search based approach for solving a class of bilevel programming problems in chemical engineering. *Journal of Heuristics* 9, 307–319.

Segundo, G.A.S., Krohling, R.A., Cosme, R.C., 2012. A differential evolution approach for solving constrained min-max optimization problems. *Expert Systems with Applications* 39, 18, 13440–13450.

Sinha, A., Malo, P., Frantsev, A., Deb, K., 2013. Multi-objective Stackelberg game between a regulating authority and a mining company: a case study in environmental economics. IEEE Congress on Evolutionary Computation, Cancún, México, pp. 478–485.

Stackelberg, H.V., 1952. *The Theory of the Market Economy*. Oxford University Press, New York.

Storn, R., Price, K.V., 1997. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 341–359.

Talbi, E. (ed.), 2013. *Metaheuristics for Bilevel Optimization*, Vol. 482 of *Studies in Computational Intelligence*, trans. Peacock, A.T. Springer, New York.