Computers and Mathematics with Applications 59 (2010) 706–715

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Solving a type of biobjective bilevel programming problem using NSGA-II

Minqiang Li [a], Dan Lin [b,*], Shouyang Wang [c]

[a] School of Management, Tianjin University, Tianjin 300072, PR China

[b] Department of Mathematics, Tianjin University, Tianjin 300072, PR China

[c] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, PR China

ARTICLE INFO

Article history:
Received 27 November 2007
Received in revised form 24 August 2009
Accepted 25 September 2009

Keywords:
Bilevel programming
Two-level optimization
Multiobjective evolutionary algorithms
Decision-making
Bargaining

ABSTRACT

This paper considers a type of biobjective bilevel programming problem, which is derived from a single objective bilevel programming problem via lifting the objective function at the lower level up to the upper level. The efficient solutions to such a model can be considered as candidates for the after optimization bargaining between the decision-makers at both levels who retain the original bilevel decision-making structure. We use a popular multiobjective evolutionary algorithm, NSGA-II, to solve this type of problem. The algorithm is tested on some small-dimensional benchmark problems from the literature. Computational results show that the NSGA-II algorithm is capable of solving the problems efficiently and effectively. Hence, it provides a promising visualization tool to help the decision-makers find the best trade-off in bargaining.

1. Introduction

The bilevel programming problem (BLPP) [1,2] is an optimization problem arising from hierarchical decision-making. It is equivalent to a static two-person Stackelberg game where the upper level decision-maker (the leader) and the lower level decision-maker (the follower) pursue individual objectives in a non-cooperative manner and subject to a set of interdependent constraints. In a BLPP, the set of decision variables is partitioned between two vectors x and y, and the decisions are taken sequentially. The leader goes first by selecting x and then the follower replies on this decision by solving an optimization problem parameterized in x. The optimal solution $y(x)$ is then returned to the leader as the follower's rational reaction on x. On the basis of the reactions of the follower, the leader has now to optimize his single or multiple objective functions over his decision vector x. A BLPP is called single objective if the leader and the follower both have a single objective; otherwise, it is called multiobjective. A BLPP is linear if all functions involved are linear.

A BLPP is generally non-convex and non-differentiable and thus intrinsically hard to solve. Due to the intractability of the BLPPs, evolutionary algorithms (EAs) [3,4] have found great interest in the past years as approximation methods for BLPPs. Evolutionary algorithms represent a large class of heuristic techniques based on the natural selection principle, with genetic algorithms (GAs) being amongst the most widely known. In most EAs, a population of solutions evolves from one generation to the next through the application of selection and variation, namely mutation and/or crossover, operators. Since EAs do not need specific information to guide the search, they can be applied in a wide variety of domains. Moreover, being population-based search strategies, they are naturally suited for the extension into the multiobjective optimization problem domain, and allow finding an entire set of approximate Pareto-optimal (efficient) solutions in a single execution of

* Corresponding author.
E-mail addresses: mqli@tju.edu.cn (M. Li), dlin@tju.edu.cn (D. Lin), swang@iss.ac.cn (S. Wang).

0898-1221/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2009.10.022

the algorithms. To date, many multiobjective evolutionary algorithms (MOEAs) have been proposed [5,6], and among them is NSGA-II, which has been successfully applied to a wide range of problems [5,7].

Applications of EAs to single objective BLPPs have been seen in the last decade. The existing evolutionary approaches for solving BLPPs can be roughly subdivided into three classes: Kurash–Kuhn–Tucker (KKT) methods, direct methods and vertex enumeration methods (for linear BLPPs only). The KKT approaches replace the lower level problem of a BLPP with its KKT optimality conditions, thus yielding a usual single level problem with complementary conditions in the constraints. This problem is then solved by developed EA-based algorithms. [8] further introduced 0–1 variables with respect to the complementary conditions. A GA was developed to solve the resultant single level linear mixed 0–1 programming problem. For nonlinear BLPPs, [9] designed an EA to solve the derived single level problem by incorporating a new constraint-handling method and a problem-specific crossover operator.

The second class of approaches directly solve the lower level programming problems with conventional optimization methods, or even also by using EAs if the problem is complicated enough. [10] developed a GA for linear BLPPs in which only mutation is used. [11] considered a BLPP with multiple followers, instead of the one-leader one-follower type of BLPP studied in this paper. For the lower level problem, a response solution known as the Nash equilibrium was solved for all followers by an iterative method or a designed GA. The GA was also used for solving the Stackelberg–Nash equilibrium for the type of BLPP considered. This model was extended to a fuzzy BLPP with fuzzy parameters in objective functions and constraints, and the new model was solved by a GA combined with fuzzy simulation and a neural network, where the GA was also applied for finding the followers' Nash equilibrium [12]. [13] developed a genetic approach called GAB to solve a BLPP describing a decision-making problem for transportation system planning and management. [14] proposed a co-evolutionary algorithm with the use of a BLPP, where dual populations were used to solve the upper and lower problems iteratively.

The vertex enumeration methods developed for linear BLPPs utilize the fact that an extreme point of the constraint region of a BLPP, a polyhedron, is an optimal solution to that BLPP. [15] developed a GA in which each solution was coded as a string representing a bilevel feasible extreme point. Aiming at combining classical extreme point enumeration techniques with genetic search methods, [16] designed a genetic procedure by associating solution codings with extreme points of the constraint region.

In a BLPP from the transportation decision-making domain, the leader usually needs to simultaneously consider multiple objectives since she has several aims and social concerns. MOEAs have been applied to solve this kind of multiobjective BLPP in which multiple objectives are considered at the upper level by the leader. [17] is concerned with general multiobjective BLPPs in transportation systems, and a numerical experiment was conducted on a biobjective BLPP arising from transportation network design. [18] formulated a type of network design problem with multiple objectives under demand uncertainty as a stochastic BLPP. In particular, two mean–variance biobjective models were derived, for which the stochastic simulation to simulate the uncertainty of demands was needed. In all these biobjective BLPPs concerning network design, the lower level problem is a traffic assignment problem solvable by standard methods. Both MOEAs developed in these two works combined a distance-based method for handling multiple objectives [19] with some single objective GA, in particular GAB for the algorithm developed in [13]. Case studies showed that these two MOEAs were efficient in simultaneously searching the efficient solutions, and that they performed better than some conventional methods.

The hierarchical characteristics in bilevel decision-making allow the leader to partially dominate the follower; thereby the leader would pursue her optimal solution regardless of, and often at the cost of, the follower's payoff during the sequential play in the BLPP. Thus the follower may often be unsatisfied with the final solution achieved on behalf of the leader, and some type of bargaining may take place between both sides after the optimization process. In this study, we assume that the original bilevel decision-making structure is retained: both sides have no motivation to cooperate with each other, and the hierarchical framework is strictly enforced. In such a situation, the candidate solutions for bargaining can be regarded as the efficient solutions to a specific biobjective BLPP derived from the BLPP under consideration. The model is formed simply through lifting the follower's objective function up to the upper level to constitute a biobjective problem for the leader. This model corresponds to the scenario that the leader takes into account the follower's profit while seeking to optimize her own objective, and the efficient trade-offs of both sides' objectives are considered as candidate solutions for the bargaining which ends at a compromise solution that is satisfactory to both sides.

In this study, we applied NSGA-II to solve the proposed type of biobjective BLPP. The obtained approximation to the efficient set can provide a sound foundation for non-cooperative posterior bargaining.

The rest of this paper is organized as follows. In Section 2 we give the formulation of the models and related definitions. Section 3 presents a description of NSGA-II applied to the proposed models. Computational results are given in Section 4. We conclude in Section 5.

2. Problem formulation

2.1. General multiobjective programming problem

A general multiobjective programming problem (MOPP) can be written as

$$\begin{aligned} &\min f(z) = (f_1(z), \ldots, f_m(z)) \\ &\text{s.t. } z \in X, \end{aligned} \tag{1}$$

where $z = (z_1, \ldots, z_n)$ is the decision (variable) vector, $f_k(z)$ $(1 \leq k \leq m)$ is the objective function, and $f(z) = (f_1(z), \ldots, f_m(z))$ is called the objective vector. $X \subseteq \mathbb{R}^n$ is the feasible set of the problem. Any decision vector in X is called feasible.

Definition 1. For two decision vectors $z_1, z_2 \in X$, z_1 is said to dominate z_2, denoted by $z_1 \preceq z_2$, if $f_k(z_1) \leq f_k(z_2)$ for any $k \in \{1, \ldots, m\}$ and $f(z_1) \neq f(z_2)$.

Definition 2. The decision vector $z^* \in X$ is said to be nondominated regarding a set $X' \subseteq X$ if any $z \in X'$ does not dominate z^*. When $X' = X$, z^* is called Pareto-optimal or efficient. The set N of all nondominated (resp. efficient) decision vectors regarding X' (resp. X) is called the nondominated (resp. efficient) set, and its image in the objective space $f(N) = \{f(z)|z \in N\}$ is called the nondominated (efficient) front or in short NDF (EF).

2.2. Bilevel models

A general (continuous) BLPP can be expressed mathematically as follows:

$$\begin{aligned} \min_{x,y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & \min_{y} \quad f(x, y) \\ & \text{s.t.} \quad g(x, y) \leq 0, \end{aligned} \tag{2}$$

with $x \in \mathbb{R}^{n_x}$, $y \in \mathbb{R}^{n_y}$, $F, f : \mathbb{R}^{n_x+n_y} \to \mathbb{R}$, $G : \mathbb{R}^{n_x+n_y} \to \mathbb{R}^p$, $g : \mathbb{R}^{n_x+n_y} \to \mathbb{R}^q$. x (resp. y) is called the vector of the upper (resp. lower) level decision variables, F (f) the upper (lower) level objective function, and $G \leq 0$ ($g \leq 0$) the upper (lower) level constraints.

For each x taken by the leader, let $Y(x)$ denote the solution set of the lower level problem or the follower's problem:

$$\begin{aligned} \min_{y} \quad & f(x, y) \\ \text{s.t.} \quad & g(x, y) \leq 0. \end{aligned} \tag{3}$$

$Y(x)$ is also known as the rational reaction set of the follower for fixed x. If $Y(x)$ is not a singleton, i.e. the minimal solution of (3) is not unique, a common procedure is the so-called optimistic approach in which the minimal solution minimizing $F(x, y)$ (for a fixed value of x) is assumed to be returned to the upper level as the response of the lower level to the given x [2].

Denote the constraint region of (2) by $S = \{(x, y) : G(x, y) \leq 0, g(x, y) \leq 0\}$. The feasible set, also known as the induced region, of (2) is defined as $IR = \{(x, y) : (x, y) \in S, y \in Y(x)\}$. Here it is always assumed that a unique $y \in Y(x)$ is determined with the optimistic approach when $Y(x)$ is not a singleton.

Definition 3. A point (x, y) is said to be feasible to (2) if $(x, y) \in IR$. A feasible solution (x^*, y^*) is said to be an optimal solution to (2) if $F(x^*, y^*) \leq F(x, y)$ for any $(x, y) \in IR$. $F(x^*, y^*)$ is called the optimal value for (2).

The general biobjective model associated with (2) can be used to describe the decision-making process if the sequential decision-making structure in (2) is abandoned, and both sides have a desire to cooperate:

$$\begin{aligned} \min_{x,y} \quad & (F(x, y),\ f(x, y)) \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & g(x, y) \leq 0. \end{aligned} \tag{4}$$

S is now the feasible set of the biobjective problem (4).

No simple relationship between the solutions to (2) and those for (4) exists even in the simple case of linear bilevel programming [20]. [21] has introduced an order relation such that the optimal solutions to (2) are the nondominated points with respect to the order relation and reposed (2) as a multiobjective programming problem with four criteria, two of which are the objective functions of the upper and lower level, respectively.

For the after optimization bargaining in which the hierarchical decision-making structure is retained and both sides act uncooperatively, the efficient solutions to the next biobjective model (5) would serve as good candidates. The model can be derived from (2) by adding the lower level objective function f up to the upper level:

$$\begin{aligned} \min_{x,y} \quad & (F(x, y),\ f(x, y)) \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & \min_{y} \quad f(x, y) \\ & \text{s.t.} \quad g(x, y) \leq 0, \end{aligned} \tag{5}$$

and the general Definitions 1–3 apply to model (5) when taking $m = 2$ and $X = IR$.

Several simple conclusions follow from the definitions and model formulations: (1) any solution in the efficient set of (5) is dominated by or equal to at least one efficient solution to (4), while any efficient solution to (4) cannot be dominated

by any solution to (5); (2) the optimal solution to (2), (x^*, y^*), is also an efficient solution to (5) with minimum F-value $F(x^*, y^*)$ among all efficient solutions; thus $(F(x^*, y^*), f(x^*, y^*))$ is identical to an extreme point of the EF; and (3) in such a non-cooperative bargaining process, the follower can only improve her payoff while the leader loses during bargaining.

Some cooperative bargaining models have been proposed [22,23]. These models aimed at finding a class of efficient solutions to model (4) from the currently obtained solution to BLPP (2). In contrast to the situation of non-cooperative bargaining, both sides could then benefit from the bargaining. However, this model may not apply to at least the cases when cooperation is not allowed or impossible.

Because preferences of the decision-makers are generally unknown in advance, it would be more advantageous to show the decision-makers a good representative sample of the whole EF from which they can choose the final compromise solution. The weighted sum method, one of the most common approaches for solving general MOPPs, can trace out part of the EF of model (5) by solving the parametric problem (6) with varying weight $\lambda \in [0, 1]$:

$$\begin{aligned} \min_{x,y} \quad & \lambda \cdot F(x, y) + (1 - \lambda) \cdot f(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0, \\ & \min_{y} \quad f(x, y) \\ & \text{s.t.} \quad g(x, y) \leq 0. \end{aligned} \tag{6}$$

Two recognized drawbacks of this method, to be illustrated in Section 4, are that, if the EF is not convex, there does not exist any weight λ for which the solution to problem (5) lies in the non-convex part, and that, even if the EF is convex, an even spread of weight λ does not produce an even spread of points on the EF [24].

Since the NSGA-II algorithm proves an efficient optimizer for MOPPs, we applied NSGA-II to solve model (5). In addition, it was also used for solving (4) to obtain a nondominated solution set comprising a large number of points, contrary to the solution set given in [22,23] with at most several points. All the results associated with both models can supply the decision-makers with sufficient trade-off information for posterior bargaining. Both sides could then choose a point on the EF of (5) as the final compromise solution if they insist on the original bilevel decision-making structure. Instead, if they decide to act cooperatively, they would attain improvements for both sides by locating the bargaining solution along the EF of (4).

3. The multiobjective evolutionary algorithm: NSGA-II

3.1. Introduction to NSGA-II

For a general MOPP, a powerful MOEA aims at finding a nondominated solution set regarding all solutions examined in a single run such that the associated NDF is a good approximation of the true EF. The goal is two-sided: the NDF obtained should be as close as possible to the true EF, while being as diverse as possible. To achieve this goal, NSGA-II introduces some components to be briefly described next. The reader can refer to [5,7] for details.

The partial relation $\prec_c$ is defined for the general situation when a constrained MOPP is solved by NSGA-II. A problem-specific overall constraint violation is defined to measure the degree of constraint violation for any solution in the search space. Then a decision vector x is said to constraint dominate vector y, written as $x \prec_c y$, if any of the following conditions is true: x is feasible and y is not; both solutions are infeasible, but x has smaller overall constraint violation; or, both solutions are feasible, and $x \preceq y$. The $\prec_c$ relation also serves as a simple yet efficient constraint handling method by always favoring feasible solutions over infeasible ones, or less constraint violated solutions over more constraint violated ones [5,7].

Based on $\prec_c$, a nondominated sorting process first successively partitions the current solution set (population) P into different nondominated fronts F_k, $k = 1, \ldots, n_k$. Each individual i in F_k is assigned a nondomination rank value $i_{rank} = k$. In NSGA-II, an efficient sorting process called the *fast nondominated sorting approach* is implemented. Second, the crowding distance ($i_{distance}$) is assigned for any individual i in P front by front, estimating the density of solutions in the same nondominated front F_k surrounding i. The smaller the crowding distance of an individual, the more crowded by other solutions it is regarded as being.

On the basis of these two attribute values assigned to each individual, the crowded comparison operator $\prec_n$ can then be defined to compare any pair of individuals i and j: i is said to $\prec_n$-dominate j, denoted by $i \prec_n j$, if either $i_{rank} < j_{rank}$, or $(i_{rank} = j_{rank} \wedge i_{distance} < j_{distance})$. By favoring the solutions with lower domination rank, the algorithm gradually drives the population towards the true EF. Meanwhile, the diversity in population is preserved to ensure a good spread of solutions by preferring the solutions with smaller crowding distance.

The framework of NSGA-II is given below. An initial population P_0 of size np is created randomly and for each individual $i \in P_0$, i_{rank} and $i_{distance}$ are calculated. P_0 is considered as the current parent population, and each iteration of the main loop starts. In each generation $t \geq 0$, for the current parent population P_t, the binary tournament selection strategy repeatedly selects one winner (parent) regarding $\prec_n$ from each pair of two randomly selected individuals until all np parents have been selected. These parent individuals are then paired randomly. With crossover probability p_c, each pair is recombined by the crossover operator to create two child individuals: the newly created child individuals enter the offspring population Q_t if crossover happens; otherwise the two parents are retained in Q_t. Then each individual in Q_t undergoes a mutation by the mutation operator with mutation probability p_m. Afterwards, P_t and Q_t are combined to form the population R_t with

size $2np$. R_t is then sorted into nondominated fronts and the crowding distance is computed for each individual. The t-th generation ends by selecting the better half from R_t to create the parent population P_{t+1} for the next generation. Based on $\prec_n$, this population reduction can be carried out by selecting the F_ks from $k = 1$ in sequence, until the last selected front F_{last} causes no fewer than np solutions included in all selected fronts. If the population size is more than np, F_{last} is sorted by $\prec_n$ and then truncated by keeping the better solutions to ensure the size np for the new population. The whole evolutionary process is repeated until some predefined stopping criterion is satisfied.

NSGA-II framework

1: Set all algorithmic parameters and set $t = 0$
2: Generate initial population P_0 randomly
3: Evaluate population P_0
4: Calculate $(i_{rank}, i_{distance})$ for each individual i in P_0
5: **Repeat**
6: Select parent individuals from P_t
7: Apply crossover operator to parents and create population Q_t
8: Apply mutation operator to each individual in Q_t
9: Evaluate Q_t
10: Combine Q_t and P_t to create R_t
11: Calculate $(i_{rank}, i_{distance})$ for each individual i in R_t
12: Select better half of R_t to generate P_{t+1}
13: $t = t + 1$
14: **Until** the stopping criterion is met

3.2. Algorithmic components

Some algorithmic components for the adaptation of NSGA-II to solve (5) are listed below. The detailed description of applying NSGA-II to the MOPP (4) can be found in [5,7].

Solution representation and initialization: Each individual is coded as a real vector x of dimension n, naturally representing a decision vector of the leader. The initial population P_0 is generated by random initialization of every individual x in which its i-th component x_i is drawn uniformly at random from $[a_i, b_i]$, where a_i and b_i are, respectively, upper and lower bound of x_i.

Crossover and mutation operators: The SBX operator is used for recombination and a polynomial distribution for mutation. These two operators are efficient in solving continuous optimization problems [5].

Fitness evaluation: For any given individual x, solve the follower's problem (3) to obtain the solution $y(x)$. The pair $(F(x, y(x)), f(x, y(x)))$ is then evaluated as the vector of fitness values at x.

Overall constraint violation: Suppose $G(x, y) = (G_1(x, y), \ldots, G_p(x, y))$, where G_i is a linear or nonlinear function. The overall constraint violation value of an individual x is then defined as $\sum_{i=1}^{p} \max\{G_i(x, y(x)), 0\}$.

4. Computational experiments

4.1. Experimental settings

In this section, we present numerical results obtained on a set of test problems taken from [9]. [9] collected some single objective nonlinear benchmark BLPPs published in the literature, along with best-known solutions. The problems tested here, i.e. test problems 1–9, 11–14 as given in [9], include all the one-leader one-follower BLPPs with linear or quadratic programming (QP) problems at the lower level. In addition, we also used a linear BLPP considered in [22,23], and call it problem 0 herein. Due to space limitation, only problems 0 and 1 are described below. Readers can refer to [9] for more information about the problems and the best-known solutions.

Problem 0:

$$\begin{aligned} \min_{x,y} \quad & F(x, y) = 2x - 11y \\ \text{s.t.} \quad & x \geq 0 \\ & \min_{y} \quad f(x, y) = x + 3y \\ & \text{s.t.} \quad x - 2y \leq 4,\ 2x - y \leq 24,\ 3x + 4y \leq 96, \\ & \qquad x + 7y \leq 126,\ -4x + 5y \leq 65,\ x + 4y \geq 8,\ y \geq 0. \end{aligned}$$

Problem 1:

$$\begin{aligned} \min_{x,y} \quad & F(x, y) = (x_1 - 30)^2 + (x_2 - 30)^2 - 20y_1 + 20y_2 \\ \text{s.t.} \quad & x_1 + 2x_2 \geq 30,\ x_1 + x_2 \leq 25,\ x_2 \leq 15 \\ & \min_{y} \quad f(x, y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 \\ & \text{s.t.} \quad 0 \leq y_1 \leq 10,\ 0 \leq y_2 \leq 10. \end{aligned}$$

Table 1
Computational results of running NSGA-II for model (2).

Problem	Best	Worst	Average	Standard	Best-known	CPU time
0	−85.090909	−85.085072	−85.090176	0.001323	−85.09	1.26
1	225.000763	225.000769	225.000764	0.000012	225	3.15
2	0.000001	0.000258	0.000007	0.000079	0	1.95
3	−12.678845	−12.671539	−12.678704	0.001033	−12.68	3.56
4	−29.200000	−29.198675	−29.199793	0.000317	−29.2	1.27
5	−8.917289	−8.917289	−8.917289	0.000000	−8.92	1.65
6	−7.578566	−7.577840	−7.578460	0.000164	−7.58	2.84
7	−11.998518	−11.998501	−11.998514	0.000006	−11.999	3.76
8	−3.600028	−3.599793	−3.599916	0.000087	−3.6	8.34
9	−3.920006	−3.919564	−3.919926	0.000000	−3.92	5.95
11	1000.0000	1000.0000	1000.0000	0.000000	1000	0.17
12	81.327853	81.327853	81.327853	0.000000	81.3279	0.58
13	100.00000	100.00234	100.00045	0.000634	100.0001	0.88
14	−1.209877	−1.209808	−1.209863	0.000018	−1.2098	1.48

In order to assess the results of multiple runs of NSGA-II, we used two performance indicators, namely the hypervolume indicator I_H and the unary epsilon indicator I_ϵ^1, which are recommended as proper performance measures of MOEAs in practice [25]. Performance assessing is usually done on the NDFs generated by MOEAs. The hypervolume indicator of an NDF Z, $I_H(Z)$, for a biobjective problem, is equal to the summation of all the areas of rectangles enclosed by each of the points in Z and a chosen reference point p. Here $z < p$ for any $z \in Z$. We further use the hypervolume ratio (HR) as an indicator to assess the performance: for a given reference efficient front (REF) R which is considered as a good approximation set of the true EF, define $HR(Z) = \frac{I_H(Z)}{I_H(R)}$. Thus, larger HR values generally indicate better NDFs.

For a given REF R and an NDF Z, the unary epsilon indicator value $I_\epsilon^1(Z)$ is defined as

$$I_\epsilon^1(Z) = \inf_{\epsilon \in \mathbb{R}} \{\forall v \in R\, \exists\, u \in Z : \forall i \in \{1, \ldots, m\} u_i \leq \epsilon \cdot v_i\}. \tag{7}$$

It gives the minimum factor ϵ such that, for each point in R, when multiplied by ϵ, the resulting transformed point is dominated by at least one point in Z. By definition, a lower ϵ is preferred.

We created the REF for each problem instance as follows [25]: running NSGA-II with larger population size ($np = 400$) and longer evolution process ($gen = 800$) independently for 5 times; merging nondominated sets from all runs and removing all dominated solutions. The image of the remaining nondominated solutions is then considered as the REF used in the evaluation of HR and I_ϵ^1 indicators. It is seen in Figs. 2 and 3 that such an REF can be regarded as a good approximation of the true EF.

Normalized objective values were used. After scaling, the objective values all lie in the interval [0, 1], and $p = (2, 2)$ is always chosen as the reference point [25]. With respect to the calculation of indicator values, including the scaling and normalizing of objective values, we used corresponding tools taken from the tool environment based on the *Platform and Programming Language Independent Interface for Search Algorithms PISA* [26], which supplies all necessary tools for performance assessment of multiobjective optimizers.

The algorithm was coded in C and run on an Intel Pentium IV 2.93 MHz PC with 1 GB RAM under Windows XP. The development of programs was based on the NSGA-II C code available from Deb's KANGAL web page. The standard simplex algorithm and an approach based on the Frank–Wolfe method [27] were used to solve the follower's linear and QP problems, respectively. These two approaches were also coded in C and embedded in the NSGA-II program. In all experimental simulations, we set $p_c = 0.95$ and $p_m = 0.2$.

4.2. Computational results

To demonstrate the efficiency of NSGA-II, we first used it to solve the original single objective BLPP model (2), which can be considered as a reduced case of an MOPP. The codes developed could be implemented without modification, except for setting the single objective function. The algorithm was run 50 times independently on each test instance with $np = 100$ and $gen = 200$, using different random number seeds. Table 1 presents the statistical results of the 50 best F-values obtained in all runs for a problem. In addition, the best-known F-value and the average CPU time in seconds for each instance are also displayed. All these results indicate that NSGA-II is able to generate competitive solutions to best-known results on *all* test problems, in a reasonable time of computation. Moreover, its performance is robust since all the observed standard deviations are quite small.

In the same way, NSGA-II was applied to trace out the EFs of model (4) by running it 1000 times to solve the weighted single objective model (6) with different $\lambda = \frac{k}{1000}, k = 0, \ldots, 1000$. Because of the steady and good performance of NSGA-II demonstrated in Table 1, the solutions obtained can be seen as good candidates for true efficient solutions.

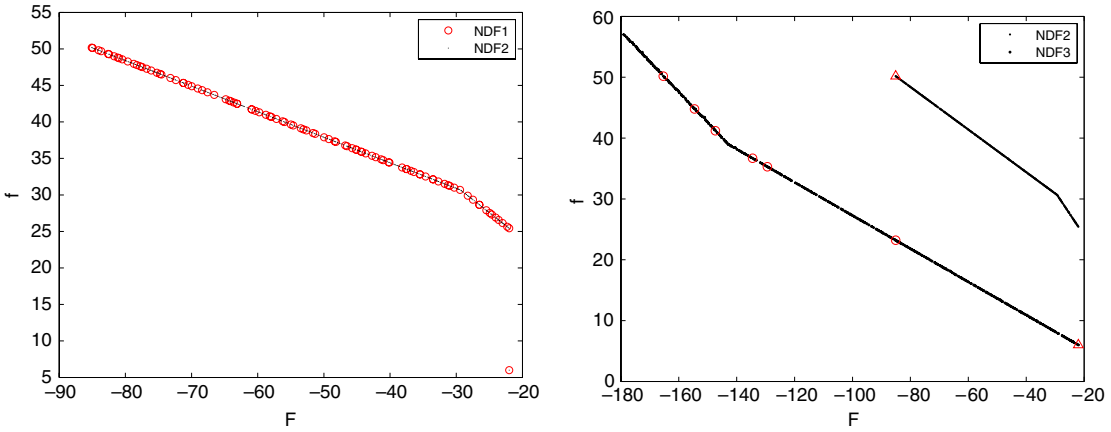
Table 2 gives a summary of the results of applying the indicators HR and I_ϵ^1 to the 50 independent runs of NSGA-II for model (5) for each test instance. The algorithmic setting used here is $np = 100$ and $gen = 200$. For each of problems 3, 7, 11 and 13, there was one single efficient solution found. This indicates that the objectives at both levels are not conflicting for

Table 2
Computational results of running NSGA-II for model (5).

Problem	*HR*		I_ϵ^1		CPU time
	Avg.	Std.	Avg.	Std.	
0	0.992274	0.000358	1.006585	0.001101	1.43
1	0.988194	0.010667	1.030119	0.032766	1.75
2	0.999594	0.000088	1.002913	0.000692	12.43
4	0.991241	0.000453	1.009477	0.001762	1.14
5	0.991684	0.000416	1.010850	0.001973	1.20
6	0.991432	0.000428	1.010738	0.002147	2.33
8	0.992865	0.000639	1.006345	0.001883	4.76
9	0.975328	0.005398	1.059657	0.015966	8.05
12	0.991697	0.000415	1.010752	0.002070	0.84
14	0.983236	0.000927	1.021235	0.004396	16.56

Table 3
Results of minimum F-values found in solving model (5) with NSGA-II.

Problem	Best	Worst	Avg.	Std.	Best-known
0	−85.090813	−85.073409	−85.085193	0.004850	−85.09
1	225.045684	231.219198	227.285323	4.036734	225
2	0.000012	0.000088	0.000036	0.000054	0
4	−29.199894	−29.185309	−29.197406	0.002955	−29.2
5	−8.917289	−8.917209	−8.917264	0.000057	−8.92
6	−7.578557	−7.574339	−7.577402	0.001052	−7.58
8	−3.596280	−3.591434	−3.594367	0.003028	−3.6
9	−3.914242	−3.891611	−3.904327	0.005192	−3.92
12	81.327853	81.329366	81.327945	0.000228	81.3279
14	−1.209877	−1.209765	−1.209804	0.000787	−1.2098



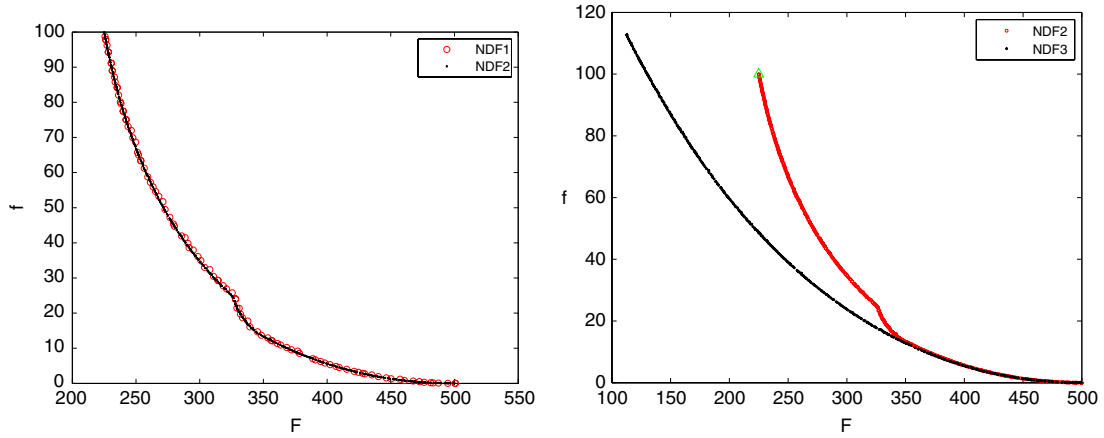
(a) NDFs for model (5). (b) NDF for model (4).

Fig. 1. NDFs obtained with NSGA-II on problem 0.

these problems, and they are omitted from the table. Very good performance in terms of both indicators can be observed from the table, and it is evident that both indicators have small variation, suggesting a reliable performance of the algorithm. Moreover, the mean CPU time is less than 20 s in all instances, while being less than 3 s for 6 out of 10, for a single run of NSGA-II to compute a good approximation EF.

Table 3 summarizes the results of those minimum F-values found in 50 runs of NSGA-II for biobjective model (5). The results prove again that NSGA-II is competitive to single objective optimizers regarding quality of solutions and computational efficiency. They also indicate the robustness of NSGA-II, since except for problem 1, the standard deviations are all rather small.

Figs. 1 and 2 show the NDFs obtained for problems 0 and 1. The left box in each figure shows the NDFs resulting from two typical runs of NSGA-II for model (5), with different settings: $np = 100$, $gen = 200$ and $np = 400$, $gen = 800$. They are denoted by NDF1 and NDF2, respectively. The point corresponding to the best-known solution is also depicted (with the symbol "△"). The NDF obtained for model (4) with setting $ps = 400$, $gen = 800$, denoted by NDF3, is shown in the right box, with NDF2 alongside for comparison. In addition, in Fig. 1, for problem 0, the approximate solutions to (4) given in [22] are also plotted in the objective space (with the symbol circle).



(a) NDFs for model (5). (b) NDF for model (4).

Fig. 2. NDFs obtained with NSGA-II on problem 1.

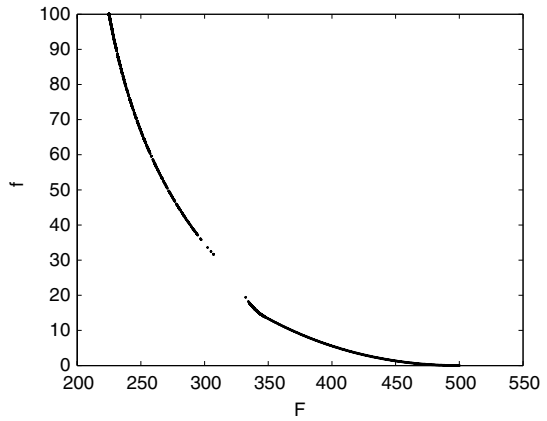


Fig. 3. Approximate EF obtained for model (5) with the weighted sum method on problem 1.

As indicated by the results of indicators in Table 2, every NDF1 is found to be a good representation of NDF2 in terms of both closeness and extension properties, meaning that we can get a good approximation of the EF in a shorter time. It is also easy to see that all the points from [22] lie exactly on the obtained front NDF3. Thus, NSGA-II is more advantageous to the decision-makers, since they can now make the choice from the approximate set to the whole EF, rather than being content with just some points thereon.

It can be observed that NDF2 is concave in the middle part for problem 1, and entirely concave for problem 0 with an isolated rightmost point. Due to the good performance of NSGA-II demonstrated in the results above, we believe that the true EFs would take on the same shape as NDF2. When applied to model (6), the weighted sum method was found to be highly susceptible to the shape of EF in that it could not trace out the concave part of the EF, as illustrated in Fig. 3, which shows the approximate EF for problem 1 obtained after 1000 runs of the algorithm with different λ. For problem 0, only the leftmost and rightmost points of the EF were found in all runs. Compared to the weighted sum method, NSGA-II is much less sensitive to the EF's shape and it is able to approximate the EF completely in a single run, a fact reflected by the plots of NDFs in Figs. 1 and 2. The approximate EF in Fig. 3 overlaps so highly with the front NDF2 in Fig. 2 that we plot them separately in different figures; however, to achieve the results shown here, it took the weighted sum method over 1500 s of CPU time while NSGA-II spent only less than 50 s. This fact strongly suggests the effectiveness and efficiency of NSGA-II.

It is interesting to note that, in Fig. 1, the isolated rightmost point of NDF2 locates exactly on NDF3. Fig. 2 shows that some points in NDF3 are approximated by mostly identical or at least very close points of NDF2. Extensive computational experiments showed that this phenomenon can also be seen for some other problems. For example, for problem 9, the majority of NDF2 coincides with NDF3, as shown in Fig. 4. Based on all such results we guess that for a broad class of problems, some efficient solutions to model (5), in particular those with relatively larger F-values, also belong to the efficient set of model (4). This is worth further theoretical research.

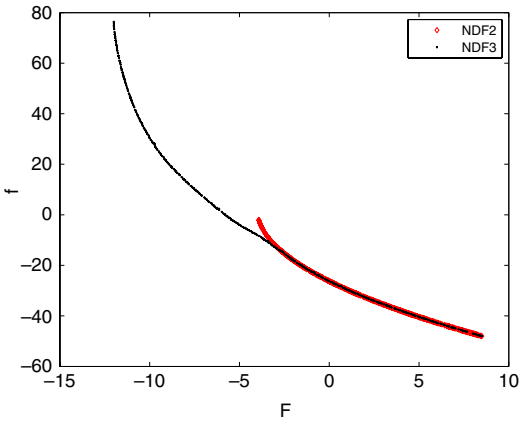


Fig. 4. NDFs obtained for models (4) and (5) with NSGA-II on problem 9.

5. Conclusion

We have considered a biobjective bilevel model derived from the original single objective BLPP by adding the follower's objective function up to the upper level. The efficient solution set of this model can be supplied to the decision-makers at both levels, who agree to bargain after optimization while keeping the hierarchical decision structure of the BLPP and acting uncooperatively, as candidates for the final compromise solution to choose from. NSGA-II has been used to solve this model. Computational results on some published low-dimensional test problems show that NSGA-II can effectively generate a good approximation efficient set in each run.

Although we have mainly focused on the special class of biobjective BLPPs, it is clear that NSGA-II (and other MOEAs) can be applied without modification to solve a general BLPP with multiple objectives considered by the leader at the upper level. Moreover, the algorithm can be extended to solve other types of BLPP, such as integer, 0–1, or mixed-integer BLPPs, through adopting a problem-specific representation scheme. These will leave as future research.

Acknowledgements

The authors thank the two anonymous reviewers for their valuable comments. The first author's research is supported by The National Science Foundation of China for Distinguished Young Scholars (Grant No. 70925005), and the second author's research is supported by the Project sponsored by SRF for ROCS, SEM.

References

[1] J.F. Bard, Practical Bilevel Optimization: Algorithms and Applications, Kluwer Academic Publishers, Boston, 1998.
[2] S. Dempe, Foundations of Bilevel Programming, Kluwer Academic Publishers, Dordrecht, 2002.
[3] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed, Springer-Verlag, Berlin, 1996.
[4] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Oxford University Press and Institute of Physics, London, 1997.
[5] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, 2001.
[6] C.A. Coello Coello, D.A. Van Veldhuizen, G.B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002.
[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE. Trans. Evolut. Comput. 6 (2002) 182–197.
[8] I. Nishizaki, M. Sakawa, K. Niwa, Y. Kitaguchi, A computational method using genetic algorithms for obtaining Stackelberg solutions to two-level linear programming problems, Electr. Eng. Japan 85 (2002) 55–62.
[9] Y. Wang, Y. Jiao, H. Li, An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme, IEEE. Trans. Syst. Man. Cybern. C 35 (2005) 221–232.
[10] R. Mathieu, L. Pittard, G. Anandalingam, Genetic algorithms based approach to bi-level linear programming, RAIRO-Oper. Res. 28 (1994) 1–22.
[11] B. Liu, Stackelberg–Nash equilibrium for multilevel programming with multiple followers using genetic algorithms, Comput. Math. Appl. 36 (1998) 79–89.
[12] J. Gao, B. Liu, Fuzzy multilevel programming with a hybrid intelligent algorithm, Comput. Math. Appl. 49 (2005) 1539–1548.
[13] Y. Yin, Genetic-algorithms-based approach for bilevel programming models, J. Transp. Eng. 126 (2000) 115–120.
[14] V. Oduguwa, R. Roy, Bi-level optimisation using genetic algorithm, in: Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems, ICAIS02, IEEE Computer Society, Washington, 2002, pp. 123–128.
[15] S.R. Hejazi, A. Memariani, G. Jahanshahloo, M.M. Sepehri, Linear bilevel programming solution by genetic algorithm, Comput. Oper. Res. 29 (2002) 1913–1925.
[16] H.I. Calvete, C. Gale, P.M. Mateo, A new approach for solving linear bilevel problems using genetic algorithms, Eur. J. Oper. Res. 188 (2008) 14–28.
[17] Y. Yin, Multiobjective bilevel optimization for transportation planning and management problems, J. Adv. Transp. 36 (2002) 93–105.
[18] A. Chen, K. Subprasom, Z. Ji, A simulation-based multi-objective genetic algorithm (SMOGA) procedure for BOT network design problem, Optim. Eng. 7 (2006) 225–247.
[19] A. Osyczka, S. Kundu, A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm, Struct. Optim. 10 (1995) 94–99.
[20] P. Marcotte, A note on a bilevel programming algorithm by LeBlanc and Boyce, Transp. Res. 22 B (1988) 233–237.
[21] J. Fliege, L.N. Vicente, A multicriteria approach to bilevel optimization, J. Optim. Theory Appl. 131 (2006) 209–225.

[22] U.P. Wen, S.T. Hsu, Efficient solutions for the linear bilevel programming problem, Eur. J. Oper. Res. 62 (1991) 354–362.
[23] M. Soismaa, A note on efficient solutions for the linear bilevel programming problem, Eur. J. Oper. Res. 112 (1999) 427–431.
[24] I. Das, J. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, Struct. Optim. 14 (1997) 63–69.
[25] J.D. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastive multiobjective optimizers, TIK-Report No.214, Computer Engineering and Networks Laboratory, ETH Zurich, 2006.
[26] S. Bleuler, M. Laumanns, L. Thiele, E. Zitzler, PISA—A platform and programming language independent interface for search algorithms, in: C.M. Fonseca, et al. (Eds.), Evolutionary Multi-Criterion Optimization (EMO 2003), in: Lecture Notes in Computer Science, vol. 2632, Springer-Verlag, Berlin, 2003, pp. 494–508.
[27] M. Frank, P. Wolfe, An algorithm for quadratic programming, Nav. Res. Logist. Q 3 (1956) 95–110.