Constraint Programming Approach to a Bilevel Scheduling Problem

András Kovács and **Tamás Kis**
Computer and Automation Research Institute,
Hungarian Academy of Sciences,
Kende utca 13-17, 1111 Budapest, Hungary
{akovacs,tamas.kis}@sztaki.hu

Abstract

Bilevel optimization problems involve two decision makers who make their choices sequentially, either one according to its own objective function. Many problems arising in economy and management science can be modeled as bilevel optimization problems. Several special cases of bilevel problem have been studied in the literature, e.g., linear bilevel problems. However, up to now, very little is known about solution techniques of discrete bilevel problems. In this paper we show that constraint programming can be used to model and solve such problems. We demonstrate our first results on a simple bilevel scheduling problem.

Introduction

Bilevel programming deals with decision and optimization problems whose outcome is decided by the interplay of two self-interested decision makers. The parties have complete and mutual knowledge about each other's models. First, the decision maker called the *leader* makes its choice. Then, in view of the leader's decision, the *follower* chooses its response. Either decision maker aims at minimizing (maximizing) its own objective function. In the general case, the objective values mutually depend on the choices of the other party. Technically, the follower's role can be seen as solving a parametric optimization problem, whose parameters are determined by the leader. The particularly interesting situation is that of the leader, who is assumed to have a complete knowledge of the follower's constraints, objective, and input data. He endeavors to find his best choice subject to the response that he can expect from the self-interested follower. In the optimistic (pessimistic) case the leader assumes that the follower chooses from the set of its optimal responses the one that is the most (least) favorable for the leader.

Formally, the set of all variables in the problem is partitioned into two sets: the leader's variables X^L, and the follower's variables X^F. The leader can assign values to X^L, while the follower decides about X^F. The leader aims at minimizing f^L subject to the constraint set C^L and the follower's optimality condition, which states that the follower will minimize f^F subject to C^F, see lines (1-3). From this formulation it also follows that the leader must avoid the values of X^L for which the follower's response does not satisfy C^L. Throughout the paper we assume that both the leader and the follower try to *minimize* their objectives, though, the same techniques can be used for maximization or mixed problems as well.

Minimize

$$f^L(X^L, X^F) \tag{1}$$

subject to

$$C^L(X^L, X^F) \tag{2}$$

$$X^F \in \arg\min_{X^{F'}}(f^F(X^L, X^{F'}) \mid C^F(X^L, X^{F'})) \tag{3}$$

Probably the earliest example and a motivation of bilevel optimization problems came from economic game theory. In a two-player *Stackelberg game* two competing firms, the market leader and a follower company, for example a new entrant, produce equivalent goods. The firms decide their production quantities sequentially, which together determine the market price, with the aim of maximizing their own profit (Dempe 2002). Various other bilevel optimization problems arise naturally in economy and management science. Perhaps the most widely discussed example is the *toll setting* problem in a network, e.g., in a system of regional highways (Labbé, Marcotte, & Savard 1998). The owner of the network (the leader) seeks for the optimal pricing of each link in the network so as to maximize its profit. The follower corresponds to the ensemble of the users of the network. A fixed amount of users belong to each origin-destination pair, and each user selects the path that minimizes his costs, composed of the travel time and the tolls to pay. Many variations of this basic problem have been investigated, including problems where tolls or traffic signs are set by the local authorities who wish to control the movement of hazardous materials or consider other environmental effects (Marcotte *et al.* 2009). Another typical application is the optimization of chemical processes. Here, the follower's optimality condition describes that the steady-state result of a chemical reaction is an equilibrium where the reacting substances reach their energy minimum.

Despite all the above, well-founded theoretical results are known for special cases of bilevel problems only. These include various exact and heuristic approaches to linear bilevel

problems (where all constraints and both objective functions are linear expressions over continuous variables), and mostly heuristic methods for other cases, such as bilinear problems (Dempe 2002). For discrete bilevel problems, which are in the focus of this study, only sporadic results are available. For example, (Lukač, Šorić, & Rosenzweig 2008) consider the production planning problem of a pharmaceutical company, while (Karlof & Wang 1996) study a bilevel problem that may arise in flow shop scheduling. The latter papers take a relatively straightforward solution approach: they enumerate (a part of) the leader's possible choices, and for each choice, compute the follower's response. In (Kis & Kovács 2009), we present basic complexity and algorithmic results for bilevel scheduling problems.

To the best of our knowledge, the solution of bilevel optimization problems using constraint programming (CP) techniques has not been investigated yet. Nevertheless, bilevel programming is strongly related to the class of quantified constraint optimization problems (Benedetti, Lallouet, & Vautard 2007). We analyze this relationship later in this paper.

The paper is organized as follows. First, we illustrate bilevel optimization problems on a sample problem from the scheduling domain. After making the necessary basic definitions and presenting some basic theoretical results, we introduce a generic CP approach to discrete bilevel optimization problems. Then, we illustrate the use of those techniques on the sample problem and present computational results.

A sample problem

The classical approach in management science assumes that the different departments of the same company, although have individual decision roles and responsibilities, subsume their interest to the same global objective. This objective is related to maximizing the long-term profit of the company. The reality is often different: the performance of each department is evaluated using, and rewarded based on, a different performance measure. These performance measures are only distantly related to the global objective of the company, and are often conflicting. Hence, a relevant alternative model of the joint operation of several departments is using multilevel programming techniques. A simple case study is presented below.

Consider the bilevel scheduling problem where the management of the company (the leader) is responsible for *order acceptance* and the workshop foreman (the follower) decides on the *execution sequence* of the tasks corresponding to accepted orders. The leader has no direct influence on the sequencing decisions. Formally, there is a set of tasks T, some of which will have to be scheduled on a single unary resource. Task j is characterized by its processing time p_j, release time r_j, and deadline d_j. The difference between the profit if j is executed on time and the loss of reputation if it is rejected is captured by the cost (or task weight) w_j^L to be paid if the task is rejected. A solution is acceptable for the leader only if all the accepted tasks are completed on time. The leader must select the tasks that will be actually executed: the binary variable x_j is 1 if task j is accepted and 0 if rejected. The objective of the leader is to minimize $\sum_j w_j^L(1-x_j)$ subject to the temporal constraints.

The sequencing decisions are made by the follower, who aims at minimizing the total weighted completion time of the tasks selected by the leader, i.e., $\{j \mid x_j = 1\}$. The completion time of tasks j is denoted by C_j, and the task weights w_j^F that express the importance of tasks for the follower are independent from the leader's task weights w_j^L. We assume that the follower observes the release times, but the organizational relations within the company are such that the leader cannot force the follower to obey the deadlines. Hence, it might happen that a set of tasks could be scheduled on time, but the follower prefers to execute them in a sequence that violates some deadlines. Such task sets do not lead to feasible solutions of the bilevel problem. Using the classical scheduling notation, the follower's problem corresponds to a parametric version of $1|r_j|\sum_j w_j^F C_j$, where the parameters x_j decide the set of tasks to be considered by the follower. This sample problem is a special case of bilevel problems where the leader's objective depends only on the leader's variables. However, the feasibility of a solution depends on the follower's response as well.

Below we present an example to demonstrate the difference between the single level and the bilevel problem. Consider the instance presented in Fig. 1. In the single level case, the leader could accept all the four tasks and process them, e.g., in the order $(1, 2, 3, 4)$. In the bilevel case, the leader only chooses the tasks to process, but the follower sequences them. If the leader selects all tasks, then the follower's response is the solution of the corresponding $1|r_j|\sum_j w_j^F C_j$ problem, i.e., the sequence $(4, 3, 2, 1)$. This solution is infeasible, because the deadline of task 1 is violated. In fact, the optimal bilevel solution is selecting the tasks $\{1, 2, 3\}$, and processing them in the order $(1, 3, 2)$, which respects all deadlines. An interesting, seemingly paradoxical situation is that the strictly smaller set of tasks $\{1, 2\}$ cannot be scheduled, because the follower's response, $(2, 1)$, violates the deadline of task 1. This also warns us that inference methods that work for the single level case might not generalize to the bilevel problem.

Basic properties of discrete bilevel problems

In this section we present some basic properties of discrete bilevel problems. First, we show that bilevel optimization differs substantially from single level bicriteria optimization, but they are strongly related to quantified constraint satisfaction problems. Then, we investigate the potential definitions of the follower's optimality condition, and analyze how the bilevel problem can be relaxed to a single level problem. Finally, we address the computational complexity of bilevel problems.

Difference of the bilevel and bicriteria approaches

Although both bilevel programming and single level bicriteria approaches seek for solutions that are attractive w.r.t. two different objective functions, the two approaches differ essentially. They model two different situations: bicriteria optimization looks for the best compromise in a centralized

Task j	p_j	r_j	d_j	w_j^L	w_j^F
1	1	0	1	2	1
2	2	0	100	2	4
3	1	1	100	2	20
4	1	0	100	1	5

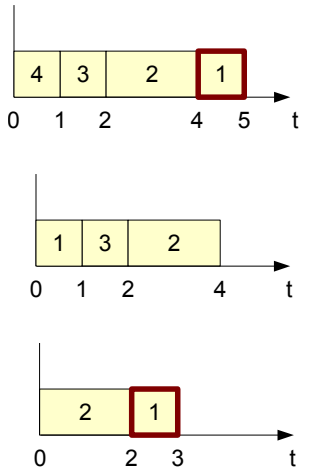


Figure 1: A bilevel problem instance and the follower's response for various choices of the leader. The tasks marked with a thick frame in the schedules violate their deadlines.

way, while bilevel optimization follows a simple, hierarchical protocol with two autonomous partners, each interested in optimizing its own objective value. Indeed, the optimal solution of the bilevel problem might not be Pareto optimal for the corresponding single level bicriteria problem, and vice versa. Below we illustrate this phenomenon on our sample problem.

Consider the problem instance presented in Fig. 2. The candidate task sets to be scheduled are $\{1\}$, $\{2\}$, $\{3\}$, and $\{1,2\}$, and it is easy to see that no other task set can be scheduled to meet the deadlines. A Pareto optimal schedule is $(1,2)$, denoted by S_1, which leads to objective values $f^L = 3$ and $f^F = 7$. Observe that schedule S_1 is not a feasible solution of the bilevel problem: if the leader decided to accept tasks $\{1,2\}$, then the follower would sequence these according to $(2,1)$, resulting in schedule S_2. However, S_2 violates the leader's deadline constraint on task 1, and hence, it is not a feasible solution of the bilevel problem. In fact, the optimal bilevel solution is the schedule (3), called S_3, which has $f^L = 4$ and $f^F = 20$. The leader prefers S_3 to $\{1\}$ and $\{2\}$ as well. Note that S_1 Pareto dominates S_3, which means that the bilevel optimal solution is Pareto dominated.

On the optimistic and pessimistic cases

Constraint (3) in the definition of the bilevel problem states that X^F must be chosen in such a way that, given X^L, it minimizes the follower's cost. However, it does not provide further details on how the follower chooses its response if he has several optimal solutions, although these can differ essentially from the viewpoint of the leader. This holds especially if only a subset of the follower's optimal solutions satisfy C^L or they lead to different values of f^L. The definition presented in lines (1-3) assumes that the leader is allowed to choose one from the follower's optimal solutions, or, equivalently, the follower is friendly enough to choose an optimal response that satisfies C^L and minimizes f^L, if there exists one. This is called the *optimistic* bilevel case.

In contrast, in the *pessimistic* case, the leader wishes to safeguard against the risks of an unfavorable follower response by assuming that the follower selects its optimal response that is the least favorable for the leader. Hence, the leader must select values for X^L in such a way that *all* optimal solutions of the follower satisfies C^L. Also, the optimal follower response that maximizes f^L is considered. In this paper, we limit our scope to the optimistic case. Note however that similar techniques can be used for the pessimistic case, except that the pessimistic model requires reified constraints to check whether there exists an optimal follower response that violates *at least one* constraint in C^L. Hence, the solvability of the pessimistic case depends on whether the applied CP solver allows reification for the constraints in C^L, and it is typically more challenging computationally than the optimistic case.

Related problems in CP

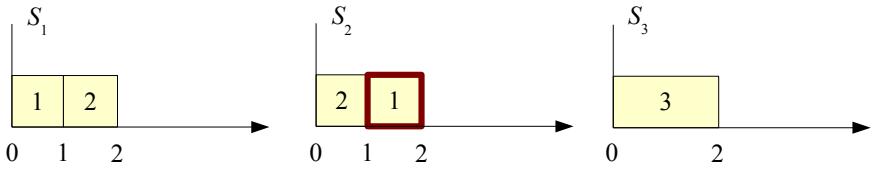
In constraint programming, a problem class strongly related to bilevel programming is the class of *quantified constraint satisfaction problems* (QCSPs), and their optimization versions, *quantified constraint optimization problems* (QCOPs) (Benedetti, Lallouet, & Vautard 2007). While a classical constraint program corresponds to evaluating a formula that contains existentially quantified variables only (e.g., $\exists x \exists y\ C(x,y)$), in QCSP it is allowed to have universally quantified variables as well (e.g., $\exists x \forall y\ C(x,y)$). In (Benedetti, Lallouet, & Vautard 2007), the basic QCSP and QCOP language has been extended with restricted quantification, resulting in the QCSP+ and QCOP+ languages. A sample QCSP+ formula is $\exists x \forall y [L(x,y)]\ C(x,y)$, which contains the restricted quantification $\forall y[L(x,y)]$. This reads *"for all y such that $L(x,y)$ it holds that..."*. It is easy to show that a QCSP+ formula can be translated in a QCSP formula with negation and disjunction. A generic solver for QCOP+ called QeCode has been made available by the the same authors (Benedetti, Lallouet, & Vautard 2006).

Now, we show that the class of discrete bilevel problems is equivalent to QCOP+ with a single pair of quantifiers $\exists \diamond \forall \diamond$, assuming that the function symbols f^L and f^F and the relation $\leq$ is available in the constraint language. First, note that the optimistic bilevel problem corresponds to the QCOP+

$$\min_{x^L, x^F} \{ f^L(x^L, x^F) \mid C^L(x^L, x^F)\ \wedge\ C^F(x^L, x^F) \wedge \forall x^{F'} [C^F(x^L, x^{F'})]\ f^F(x^L, x^F) \leq f^F(x^L, x^{F'}) \}. \quad (4)$$

Furthermore, the pessimistic bilevel problem can be rewritten as

Task j	p_j	r_j	d_j	w_j^L	w_j^F
1	1	0	1	2	1
2	1	0	2	2	3
3	2	0	2	3	10



Schedule	f^L	f^F	Feasibility	Optimality
S_1	3	7	Feasible	Pareto optimal
S_2	3	5	Task 1 violates deadline	-
S_3	4	20	Feasible	Bilevel optimal

Figure 2: Difference of the bilevel and the bicriteria Pareto optimal solutions.

$$\min_{x^L,x^F}\{f^L(x^L,x^F) \mid C^L(x^L,x^F) \wedge C^F(x^L,x^F) \wedge \forall x^{F\prime}[C^F(x^L,x^{F\prime})]\ f^F(x^L,x^F) \le f^F(x^L,x^{F\prime}) \wedge \forall x^{F\prime\prime}[C^F(x^L,x^{F\prime\prime}) \wedge f^F(x^L,x^F) = f^F(x^L,x^{F\prime\prime})]\ C^L(x^L,x^{F\prime\prime}) \wedge f^L(x^L,x^F) \ge f^L(x^L,x^{F\prime\prime})\}. \quad (5)$$

On the other hand, the QCOP+ formula $\min\{f(x) \mid \forall y[L(x,y)] : C(x,y)\}$ can be rewritten to a bilevel program with $f^L \equiv f$, $f^F \equiv 0$, $C^L \equiv C$, and $C^F \equiv L$.

Another related problem in constraint programming is the class of *adversarial constraint satisfaction problems* (ACSP) (Brown *et al.* 2004). ACSP can be used to model games played by n agents with potentially conflicting interests that consist of a fixed number of rounds. The number of rounds equals the number of variables in the ACSP, which is typically much larger that the number of agents. The main differences between ACSP and bilevel programming is that in ACSP in each round the forthcoming agent is free to choose an arbitrary variable to instantiate, i.e., variables are not assigned to agents a priori. Also, in ACSP, all agents must satisfy the same set of constraints, although in theory it is possible to incorporate a measure of constraint violations into the optimization criteria of the agents.

The single level relaxation

Various components of the solution algorithms for bilevel problems rely on well understood techniques for single level problems. Therefore, it seems natural to look for relations between bilevel and single level problems. The simplest way of reduction is to let the leader decide on every variable, and completely disregard the existence of the follower. The resulting problem will be called the *single level relaxation* of the bilevel problem, and its solution value is obviously a lower bound on the bilevel solution cost:

Definition 1 *The single level relaxation of a bilevel program is, using the set of variables $X = X^L \cup X^F$, the problem* $\min\{f^L(X) \mid C^L(X) \wedge C^F(X)\}$.

Computational complexity

Bilevel problems are complex optimization problems, they often belong to a higher complexity class than their corresponding single level relaxations. For example, linear bilevel problems (where both the single level relaxation and the follower's subproblem is a linear program) are known to be NP-complete (Dempe 2002). Here, we focus on the complexity of decision versions of discrete bilevel problems, especially in the case where the (decision version of the) single level relaxation is NP-hard. It is easy to observe that a bilevel problem is – except for degenerate cases – at least as complex as its single level relaxation, hence, NP-hard. On the other hand, since discrete bilevel problems can be reduced to QCSP+ formulae, they are in PSPACE.

Now, whether a discrete bilevel program belongs to NP or not depends, at the first place, on the complexity of the follower's subproblem. While a follower's subproblem in P guarantees that the bilevel problem is in NP, it is easy to define problems that are outside NP. Consider an unconventional, but valid bilevel scheduling problem where all variables and constraints belong to the follower, while the leader is represented only via its objective function. All tasks must be scheduled on a single machine without overlap or preemption, subject to release times r_j and strict deadlines d_j. Assume that all parameters are integer. The objective of the leader is to minimize $\sum C_j$, whereas the follower minimizes the weighted earliness penalty $\sum w_j(d_j - C_j)$. This problem is outside NP (unless P=NP), because it is both NP-hard and co-NP-hard. The latter holds because verifying the feasibility of a bilevel solution is equivalent to proving that no better solution exists for the follower's NP-hard subproblem.

The above complexity results indicate that no direct encoding of bilevel problems into CP or MIP can be expected. For the case of our main sample problem, we were not able

to prove that it is outside NP, but we conjecture that it is, since no trivial certificate seems to exist for a positive answer.

Modeling and solving bilevel problems by CP

In this section we first present a generic approach to solving discrete bilevel optimization problems by CP. Then, we introduce several algorithmic techniques to improve the efficiency of the solver. Each of the subsections has a counterpart in the next section, where the use of the given technique is illustrated on the sample problem.

The constraint model

Given the discrete bilevel problem (1-3), let us define an equivalent constraint program that encodes the bilevel problem from the leader's point of view. We also call it the *master problem*. The variables are both X^L and X^F. They are subject to constraints C^L, C^F, and C^*. C^L is the set of the leader's constraints, and C^* states that the follower selects its optimal response for any given decision of the leader. Finally, the follower's set of constraints C^F is also part of the model; it is redundant, but it strengthens propagation by filtering out choices that do not lead to a feasible follower response at all, since the propagation of C^* can be incomplete.

Minimize

$$f^L(X^L, X^F) \tag{6}$$

subject to

$$C^L(X^L, X^F) \tag{7}$$

$$C^F(X^L, X^F) \tag{8}$$

$$C^*(X^L, X^F) \tag{9}$$

In the sequel we assume that model constraints C^L and C^F are classical constraints over finite-domain variables, which have appropriate propagation algorithms defined in the literature. In contrast, constraint C^* embeds the *follower's subproblem*, and states that

$$X^F \in \arg\min_{X^{F'}} \{f^F(X^L, X^{F'}) \mid C^F(X^L, X^{F'})\}. \tag{10}$$

The propagation of constraint C^* is challenging because it requires the characterization of the set of all optimal solution of a parametric optimization problem over different choices of the parameters. Unless problem-specific methods are available, we propose to overcome this difficulty by using a generate-and-test approach, i.e., solving the follower's subproblem when all of the leader's variables X^L become bound. Then, the problem reduces to a classical single level optimization problem with known parameters X^L, solvable by any exact approach, e.g., CP search. Note that the subproblem solver must ignore the domains of X^F in the master problem, since those domains are corrupted by the propagators of C^L, i.e., constraints that the follower disregards. In the optimistic case, the follower selects from the set of his optimal solutions a solution that satisfies C^L and minimizes f^L. Therefore, the solver embedded into C^* must be called twice:

(1) Determine the follower's minimum cost, $f^{F*} = \min_{X^{F'}}\{f^F(X^L, X^{F'}) \mid C^F(X^L, X^{F'})\}$. Return failure if no solution exists.

(2) Find the follower's response according to the optimistic assumptions, i.e., $X^F := \arg\min_{X^{F'}}\{f^L(X^L, X^{F'}) \mid f^F(X^L, X^{F'}) = f^{F*} \wedge C^L(X^L, X^{F'}) \wedge C^F(X^L, X^{F'})\}$. Return failure if no solution exists.

Regarding *search techniques* for the master problem, we propose to perform any kind of search in the space of the instantiations of the leader's variables. Any exact or non-exact search technique can be used. It is not necessary to consider the follower's variables, since values will be assigned to them by constraint C^*. At the same time, exact search in the space of follower responses must be performed within C^*, since the exact optimum must be known to check this constraint.

The proposed solution method resembles the (logic-based) Benders decomposition approach to single-level problems (Hooker & Ottosson 2003). However, in the single level Benders case one is free to choose the separation of the master and the subproblem as it is the most efficient computationally, whereas in the bilevel case the separation comes from the problem definition. This leads to the second main difference of the two techniques, namely that for the problems investigated we were not able to feedback strong cuts (or constraints) from the subproblem to the master problem.

Lifting the follower's dominance rules into the master problem

While in general it is impossible to compute the follower's response before the complete instantiation of X^L, various dominance properties may be known for the follower's problem. Weak dominance rules describe properties that *all* optimal solutions of the follower's sub-problem must hold. These rules can be encoded as constraints in the master problem. Note that in general, the dominance rule compiles into a *reified* constraint, i.e., one containing disjunction or implication. If the reified constraint takes a sufficiently simple form so that it propagates even when variables X^F are unbound, then it is worth adding them to the CP master problem to strengthen propagation.

Comparing the follower's lower and upper bounds

Below we present a novel technique that prunes the search tree based on the difference of the problem models perceived by the leader and the follower. We will characterize the values that f^F can take in solutions that are feasible for the leader, as well as the values that f^F can take in optimal responses of the follower. Clearly, if the two ranges do not overlap, then there is no feasible bilevel solution in the current branch of the search tree.

Let *UB* denote the value of the best known solution, and let us characterize the current branch of the search tree by

the domain of X^L, denoted by $\mathrm{Dom}(X^L)$. Any feasible improving solution of the master problem must obey C^L, C^F, and $f^L < UB$. Hence, a valid lower bound g^{Lmin} on the values f^F in the solutions that are acceptable for the leader in the current branch is:

$$g^{Lmin} = \min_{X^{L\prime} \in \mathrm{Dom}(X^L)} \min_{X^F} \{ f^F(X^{L\prime}, X^F) \mid C^L(X^{L\prime}, X^F) \wedge C^F(X^{L\prime}, X^F) \wedge f^L(X^{L\prime}, X^F) < UB \}.$$

On the other hand, for any fixed leader's choice in this branch, the follower will return a response that minimizes f^F subject to C^F. By taking the maximum of these minimum values, we get an upper bound g^{Fmax} on f^F in the solutions that are acceptable for the follower in the current branch:

$$g^{Fmax} = \max_{X^{L\prime} \in \mathrm{Dom}(X^L)} \min_{X^F} \{ f^F(X^{L\prime}, X^F) \mid C^F(X^{L\prime}, X^F) \}.$$

Note that the constraints in the definition of g^{Fmax} are a subset of the constraints for g^{Lmin}, and therefore $g^{Fmax} < g^{Lmin}$ can occur. This means that no solution in the current branch of the search tree is both feasible for the leader and optimal for the follower. At the same time $g^{Fmax} \geq g^{Lmin}$ is also possible, since g^{Fmax} is a maximin, whereas g^{Lmin} is a minimum.

Lemma 1 *If $g^{Fmax} < g^{Lmin}$, then the current search branch contains no feasible improving bilevel solution, and therefore it can be fathomed.*

In general, it is difficult to compute the exact values of g^{Fmax} and g^{Lmin}. Instead, an upper estimate of g^{Fmax}, denote by $\hat{g}^{Fmax}$ can be used, and similarly, a lower estimate of g^{Lmin}, denoted by $\check{g}^{Lmin}$ can be applied.

Lower bounds on the leader's cost

In theory, it is straightforward to apply the classical lower bounding technique of operations research to bilevel problems: let $\check{f}^L$ be a lower bound and $\hat{f}^L$ an upper bound, typically the value of the best known solution. Now, if $\check{f}^L \geq \hat{f}^L$ then the current branch of the search tree does not contain a feasible improving solution. In practice, the effective use of this technique is challenging, because good lower bounds for bilevel problems are rarely available from the literature. A possible approach is using the single level relaxation, whose solution imposes a lower bound on the bilevel problem. If the single level relaxation is still (NP-)hard, then it can be relaxed further.

Modeling and solving the sample problem

The basic constraint model

The basic constraint model of our sample problem contains n binary variables x_j to denote if task j is scheduled, and n optional activities with start and end time variables. The activities are subject to a unary resource and time window constraints, and the follower's optimality constraint. The objective function is expressed as $f^L = \sum_j w_j^L(1 - x_j)$ using a weighted sum constraint. Our search strategy selects in each node the task j whose x_j variable is unbound and has the greatest w_j^L. Then, it creates two children of the node according to $x_j = 1$ (left branch) or $x_j = 0$ (right branch).

The follower's optimality constraint is a custom developed constraint, which embeds a constraint-based solver for the $1|r_j|\sum w_j^F C_j$ problem. The naive constraint model with a unary resource constraint, release time constraint, and the cost expressed using a weighted sum constraint is used. A classical chronological schedule-or-postpone search strategy (called *setTimes* in Ilog) is used, and the subproblem solver also includes dominance rules from (Jouglet, Baptiste, & Carlier 2004).

Lifting the follower's dominance rules into the master problem

Despite the various dominance rules known for the $1|r_j|\sum w_j^F C_j$ problem, the condition side of most of these rules is too complex to fire when only the x_j variables are bound, and very little is known about the task start times or the order. We lifted one simple dominance rule to the leader's model. It states that tasks that start after the last release time are scheduled in a *weighted shortest processing time* (WSPT, non-increasing w_j^F/p_j) order, with ties broken by *earliest due date* (EDD, non-decreasing d_j). EDD is necessary as a tie breaker due to the optimistic assumption. Now, let $r_{\max} = \max_j r_j$, and let $i \rightarrow j$ denote a precedence constraint between i and j. Then, for each pair of tasks i and j such that i precedes j in the WSPT/EDD/task index order, we add the following reified constraint to the model:

$$\mathrm{start}_i < r_{\max} \quad \vee \quad \mathrm{start}_j < r_{\max} \quad \vee \quad i \rightarrow j$$

Comparing the follower's lower and upper bounds

The upper bound Our follower's subproblem is $1|r_j|\sum w_j^F C_j$. Let $f^F(T_1)$ denote the follower's minimal cost when the leader accepts the task set T_1. It is obvious that if $T_1 \subseteq T_2$ then $f^F(T_1) \leq f^F(T_2)$. Therefore, the cost of any heuristic solution to the follower's problem with task set $T_{max} = \{j \mid 1 \in \mathrm{Dom}(x_j)\}$ can be used as $\hat{g}^{Fmax}$. In our solver, we have implemented the constructive heuristic called *CPRTWT* heuristic with the *makeBetter* improvement step after the insertion of each task to the schedule, originally introduced in (Jouglet *et al.* 2008).

The lower bound Computing $\check{g}^{Lmin}$ requires obtaining a lower bound on a $\sum w_j^F C_j$ problem subject to release times, deadlines, and optional activities. Our LB is based on the model of (Pan & Shi 2005) for a similar problem, though, without optional activities:

$$\min \sum_j w_j^F C_j \tag{11}$$

subject to

$$\forall j \qquad (r'_j + p_j)z_j \leq C_j \tag{12}$$

$$\forall j \qquad C_j \leq d'_j z_j \tag{13}$$

$$\forall i,j\ (i \neq j) \quad (C_i \leq C_j - p_j z_j) \ \vee \ (C_j \leq C_i - p_i z_i) \tag{14}$$

$$\sum_j w_j^L z_j \geq W \tag{15}$$

$$\forall j \qquad z_j \in \mathrm{Dom}(x_j) \tag{16}$$

In this formulation, variables z_j indicate if task j is processed ($z_j = 1$) or not ($z_j = 0$). Variables C_j denote the completion times. Constraints (12) and (13) specify the release times and deadlines of the task, and also ensure that C_j is 0 if j is not scheduled. Note that the time windows taken from the CP model can be used, since these are strengthened by propagation compared to the original values. Line (14) defines the unary resource constraint. Constraint (15) states that the total weight of the tasks selected for processing must be at least $W = \sum_j w_j^L - UB + 1$ in order to achieve an improving solution for the leader. In a given search node, it can already be known for some tasks if they are already selected for processing by the leader or not, while it is still an open question for the rest (16). Note that $\mathrm{Dom}(x_j) \subseteq \{0, 1\}$. Now, by moving inequalities (12) and (13) into the objective function with multipliers a and b, we receive the following Lagrangian relaxation (LR):

$$\min \sum_j w_j^F C_j + \sum_j [a_j((r'_j + p_j)z_j - C_j)) + b_j(C_j - d'_j z_j)] \tag{17}$$

$$= \sum_j (w_j^F - a_j + b_j)C_j \ + \ \sum_j [a_j(r'_j + p_j) - b_j d'_j]z_j \tag{18}$$

subject to

$$\forall i,j\ (i \neq j) \quad (C_i \leq C_j - p_j z_j) \ \vee \ (C_j \leq C_i - p_i z_i) \tag{19}$$

$$\sum_j w_j^L z_j \geq W \tag{20}$$

$$\forall j \qquad z_j \in \mathrm{Dom}(x_j) \tag{21}$$

Next, we present how the LR problem can be solved to optimality for fixed non-negative Lagrangian multipliers a and b. We exploit that the first component of the objective function corresponds to $\sum_{\{j \mid z_j = 1\}} w'_j C_j$ with $w'_j = w_j^F - a_j + b_j$, while the second component does not contain completion time variables C_j. Therefore, for any fixed z, the optimal solution is a no-delay schedule containing the selected tasks in WSPT order.

Computing the LB in leaves of the search tree We have developed two different methods for computing the optimal solution of LR: one to be used in the leaves of the search tree, and another for internal search nodes. In leaves we exploit that there are no optional tasks, i.e., the variables z_j are fixed. In this case, $\breve{g}^{Lmin}$ equals the objective value of the WSPT schedule, which can be computed in $O(n \log n)$ time.

Computing the LB in internal search nodes In internal search nodes, the optimal choice of z and the implied cost can be determined using the following dynamic program (DP). As the initialization step, we sort the tasks j that may be scheduled ($1 \in \mathrm{Dom}(x_j)$) by non-increasing w'_j/p_j, which corresponds to the WSPT order. Tasks that cannot be scheduled ($1 \notin \mathrm{Dom}(x_j)$) are completely ignored by the algorithm. The DP fills in a 3 dimensional table, whose element $u(k, v, t)$ equals the optimal relaxed cost for tasks $\{1, ..., k\}$, total leader's weight v, and schedule end time t. The first layer of the table for $k = 1$ contains the two trivial solutions, $u(1, w_1^L, p_1) = w_1^F p_1$ for the schedule that contains task 1 only, and $u(1, 0, 0) = 0$ for the empty schedule. Further elements can be obtained by recursion from $k - 1$ to k. When computing $u(k, v, t)$, one has to decide if task k is appended to the end of an earlier schedule, or a previous schedule is kept:

$$u(k, v, t) = \min(u(k-1, v - w_k^L, t - p_k) + t w_k^F,\ u(k-1, v, t))$$

The optimal solution of the DP is contained in the last layer of the table, among the elements that satisfy constraint (20), i.e., those with $v \geq W$. The DP runs in pseudo-polynomial time and space: its complexity is $O(nVP)$, where $V = \sum_j w_j^L$ and $P = \sum_j p_j$.

Setting the Lagrangian multipliers The above methods result in an optimal solution of LR for any fixed non-negative multipliers a and b, assuming $w'_j = w_j^F - a_j + b_j \geq 0$. To find the multipliers that provide the strongest LB, the above methods were embedded into a loop, and the multipliers were adjusted after each cycle. Namely, if task j violated its release time in the current optimal solution of LR, then its weight was increased to $w'_j = \frac{w'_k p_j}{p_k} + \varepsilon$, where k is the predecessor task of j in the schedule. Similarly, if j violated its deadline, then its weight was decreased to $w'_j = \frac{w'_k p_j}{p_k} - \varepsilon$, where k is the successor of j. The method was initialized with $a_j = b_j = 0$.

The best run times were achieved with the number of cycles set to 15 in leaves, and not using this method in internal search nodes (c.f. the experimental results for further details).

Lower bounds

The single level relaxation (SLR) of our problem is a $1|r_j|\sum w_j U_j$ scheduling problem, which is NP-complete. Nevertheless, various solution techniques and polynomial lower bounds are available from the literature. The current best algorithm for the SLR is the branch-and-bound of (M'Hallah & Bulfin 2007). We have implemented the mixed-integer programming (MIP) formulation of the single

level problem proposed in this paper, and solved its linear relaxation in each node of the search tree. The parameters r_j and d_j were updated in each node by the tighter time windows taken from the CP model. Note that the gap between the value of the bilevel solution originates from two sources: solving the SLR instead of the bilevel problem and the further linear relaxation of SLR. For the specific problem, we have found that over 75% of the gap is due to taking the SLR, and only 25% originates from solving the linear relaxation.

A modified propagator for C^*

It is straightforward to build a propagator for the follower's optimality constraint C^* based on the generic scheme presented in the previous section. However, below we present a modified algorithm that fully exploits the follower's lower and upper bounds during the exact solution of the follower's subproblem. Moreover, it first solves the follower's subproblem *with* the leader's deadline constraints, which is a tighter and easier-to-solve problem. Then, it also exploits the results of this step for solving the more complicated version of the subproblem, *without* the leader's constraints. This modified algorithm exploits that the leader's objective, f^L, does not depend on the follower's response.

(1) Compute $\hat{g}^{Fmax}$ and $\check{g}^{Lmin}$;

(1a) Return failure if $\hat{g}^{Fmax} < \check{g}^{Lmin}$;

(2) Compute a follower's response with minimum f^F that satisfies the leader's constraints, $X^{F+} := \arg\min_{X^{F'}}\{f^F(X^L, X^{F'}) \mid C^L(X^L, X^{F'}) \wedge C^F(X^L, X^{F'}) \wedge f^F(X^L, X^{F'}) <= \hat{g}^{Fmax}\}$ and $f^{F+} = f^F(X^L, X^{F+})$;

(2a) Return failure if no solution exists;

(3) Determine the follower's minimum cost, $f^{F*} = \min_{X^{F'}}\{f^F(X^L, X^{F'}) \mid C^F(X^L, X^{F'}) \wedge f^F(X^L, X^{F'}) <= \hat{g}^{Fmax} - 1\}$.

(3a) Abort the branch and bound if a solution with $f^{F*} < f^{F+}$ is reached. Then, return failure;

(3b) Otherwise X^{F+} is a feasible bilevel solution.

Computational experiments

In this section we report computational results achieved on the sample problem. The solver was implemented in such a way that each technique presented in a separate section above could be switched on or off individually. Moreover, the comparison of the follower's bounds could be performed independently in internal search nodes using the DP or in leaves using the WSPT schedule. Preliminary experiments showed that all the presented techniques contribute to pruning the search tree, but it does not pay off in terms of search time to use the leader's lower bound or to compare the follower's bounds in internal nodes. Therefore we decided to switch off these two components in the main experiments. Below we present a comparison of three versions of the solver: the fastest version that uses all techniques except for the previously mentioned two inefficient components, *[Main]*; a version with all features of *[Main]* except for the computation of follower's bounds, *[No fbds]*; and finally a naive solver that lacked all the improvements proposed in the previous sections, *[Naive]*.

The solver was implemented in C++ using ILOG Solver and Scheduler, both for the bilevel master problem and the follower's subproblem solver. ILOG Cplex was used for the computation of the LP lower bound. The experiments were run on a 1.86 GHz Intel Xeon computer with 2 GB of RAM under Windows Server 2003. The time limit was set to 600 seconds per problem instance.

Problem instances have been generated similarly to the instances for the single level problem of minimizing the weighted number of late jobs in (Dauzère-Pérès & Sevaux 2003) and (M'Hallah & Bulfin 2007), with the only difference that we have also added the follower's weights w_j^F. The parameters of the generator are number of tasks n, the range of release times, k_R (a larger value means a greater variance of the release times), and the tightness of the deadlines, k_D (the larger the value, the wider the time windows). Parameter n varied between 20 and 50 with increments of 5, while k_R and k_D were chosen from the set $\{1, 5, 10, 20\}$. Generating 10 instances with all possible combinations of the 3 parameters resulted in 1120 instances altogether. Processing times were generated using $U[1, 100]$, release times from $U[0, k_R n]$, deadlines from $U[r_j + p_j, r_j + p_j + k_D n]$, while weights w_j^L and w_j^F from $U[1, 10]$, where $U[a, b]$ denotes the discrete uniform distribution over integers from the interval $[a, b]$.

The comparison of the results achieved with the three different versions of the solver is displayed in Table 1, while Table 2 provides further statistics about runs of the propagator of C^* in the *[Main]* version. In both tables, each row contains combined result for the instances with a given value of N and k_D. Column *Opt* displays the number of instances that could be solved to proven optimality out of 40. *Time* contains the average computation time in seconds or 600 for instances where the time limit was hit. Column *Nodes* shows the average number of search nodes. The additional columns in Table 2 contain the number of times the propagator of C^* reached the different steps of computation: calculating the follower's bounds (*Step (1)*), the follower's minimum cost when the leader's constraints are respected (*Step (2)*), and the minimum cost when the leader's constraints are ignored (*Step (3)*). The numbering of the steps refers to the phases of the modified propagator presented in the previous section.

The results show that the two stronger versions of the solver, *[Main]* and *[No fbds]* were able to solve instances with up to 20-25 tasks to optimality, whereas the *[Naive]* version started to have difficulties even with some 20-task instances. On the whole, *[Main]* solved 5% more instances than *[No fbds]*, and 30% more than *[Naive]*. The difference becomes slightly more significant as the problem size increases, and the comparison of average computation times brings roughly the same result. Smaller values of k_D made the problems easier to solve for all versions, because then the leader had a smaller choice of task sets to accept, and those sets are identified relatively efficiently without the follower's optimality condition, too. This is made apparent especially by the low number of calls to the propagator of C^*

		[Main]			*[No fbds]*			*[Naive]*		
N	k_D	Opt	Time	Nodes	Opt	Time	Nodes	Opt	Time	Nodes
20	1	40	0.02	43	40	0.03	43	40	0.03	43
	5	40	0.08	270	40	0.18	270	40	0.30	309
	10	40	0.21	529	40	0.48	529	40	4.28	688
	20	40	7.59	4705	40	21.25	4705	26	273.73	7583
25	1	40	0.03	97	40	0.03	97	40	0.03	97
	5	40	0.57	1759	40	2.60	1759	40	5.38	1942
	10	40	5.64	7355	40	21.35	7355	35	145.75	8583
	20	29	239.78	34782	21	310.04	19417	6	548.46	12863
30	1	40	0.06	284	40	0.11	284	40	0.10	285
	5	38	34.10	27941	38	42.42	11916	37	71.89	7061
	10	36	104.97	53246	31	182.96	40530	9	505.32	23421
	20	12	449.43	124099	12	458.68	69999	0	600.00	11410
35	1	40	0.16	706	40	0.27	706	40	0.26	706
	5	37	62.16	62103	36	90.36	52112	28	257.19	21321
	10	24	333.00	210391	16	403.61	123248	3	576.94	20327
	20	4	569.98	163991	1	586.91	90687	0	600.00	1571
40	1	39	15.26	10252	39	15.34	3764	39	15.27	3376
	5	34	184.19	229326	30	272.01	169537	15	469.50	57638
	10	8	549.42	432902	7	557.19	285322	0	600.00	7933
	20	0	600.00	126248	0	600.00	44883	0	600.00	300
45	1	39	16.53	12557	39	18.81	6696	39	19.23	6361
	5	18	385.20	375424	16	407.03	187852	4	555.99	22541
	10	2	593.11	352346	0	600.00	194131	0	600.00	4485
	20	0	600.00	115571	0	600.00	35093	0	600.00	474
50	1	39	20.13	25036	39	29.42	21928	39	31.87	21907
	5	10	519.11	618822	8	540.12	304749	1	586.47	18676
	10	0	600.00	203165	0	600.00	86580	0	600.00	1334
	20	0	600.00	59771	0	600.00	19525	0	600.00	106
$\sum$		729			693			561		

Table 1: Comparison of three different versions of the solver.

with $k_D = 1$ (column *Step (1)* in Table 2). The results also depend on k_R (small k_R makes them easier to solve), but much less than on k_D or N.

The analysis of the runs of the propagator in Table 2 shows that follower's bounds computation inferred the infeasibility of the leaf in 64% of the cases. The exact CP solver had to be called without the leader's constraints in the remaining 36% of the runs (*Step (2)*), and with the leader's constraint in only 0.5% of the runs (*Step (3)*). On the one hand, this low percentage is an excellent result, since the last step of the algorithm is the most time consuming. On the other hand, it also shows that at least 99.5% of the leaves did not contain a solution that is both feasible for the leader and optimal for the follower. Hence, future research should address the efficient propagation of the follower's optimality constraint C^* also in the internal search nodes. On average, about 5% of the total computation time was spent in the propagator of C^* in the leaves.

Conclusions

This paper introduced novel CP-based modeling and solution techniques for discrete bilevel optimization problems, and showed how these can be applied to a sample bilevel scheduling problem. Since bilevel problems are computationally difficult – they are often outside NP –, techniques that improve the efficiency of the solver are of key importance. Beyond presenting how some classical techniques can be generalized to bilevel problems, we introduced a novel inference method that reduces the search space by comparing the values that the follower's objective can take in solutions acceptable for the leader and for the follower. Computational results were also presented.

We think that an interesting direction for future research is the development of new inference techniques for bilevel problems. Depending on the specific problem, these can include the filtering of the leader's variable domains based on inference from the follower's optimality condition, or the re-use of the follower's response computed in earlier visited leaves.

Acknowledgements

The authors thank József Váncza for his valuable comments and suggestions. The work reported here has been supported by OTKA grants K76810 and T73376. A. Kovács acknowledges the support of the János Bolyai scholarship No. BO/00138/07.

References

Benedetti, M.; Lallouet, A.; and Vautard, J. 2006. Qecode's web page. `www.univ-orleans.fr/lifo/members/vautard/qecode`.

N	k_D	Opt	Time	Nodes	Step (1)	Step (2)	Step (3)
20	1	40	0.02	43	6	3	3
	5	40	0.08	270	73	14	4
	10	40	0.21	529	100	39	5
	20	40	7.59	4705	2624	1270	10
25	1	40	0.03	97	6	4	3
	5	40	0.57	1759	921	43	4
	10	40	5.64	7355	2964	1065	10
	20	29	239.78	34782	17135	10990	18
30	1	40	0.06	284	21	6	4
	5	38	34.10	27941	22283	8391	137
	10	36	104.97	53246	17622	9986	102
	20	12	449.43	124099	59287	20118	12
35	1	40	0.16	706	37	20	5
	5	37	62.16	62103	31694	13873	81
	10	24	333.00	210391	59021	29022	398
	20	4	569.98	163991	79428	15446	6
40	1	39	15.26	10252	8267	4259	5
	5	34	184.19	229326	39983	18003	780
	10	8	549.42	432902	78845	37022	506
	20	0	600.00	126248	70704	10952	4
45	1	39	16.53	12557	7977	2980	17
	5	18	385.20	375424	69220	36070	1237
	10	2	593.11	352346	93630	35124	100
	20	0	600.00	115571	58313	4516	0
50	1	39	20.13	25036	6103	1799	146
	5	10	519.11	618822	73507	31618	684
	10	0	600.00	203165	61564	24269	54
	20	0	600.00	59771	30829	3552	0

Table 2: Detailed results achieved with the *[Main]* version of the solver.

Benedetti, M.; Lallouet, A.; and Vautard, J. 2007. Qcsp made practical by virtue of restricted quantification. In *International Joint Conference on Artificial Intelligence*, 38–43.

Brown, K. N.; Little, J.; Creed, P. J.; and Freuder, E. C. 2004. Adversarial constraint satisfaction by game-tree search. In *Proc. of ECAI 2004*, 151–155.

Dauzère-Pérès, S., and Sevaux, M. 2003. Using Lagrangean relaxation to minimize the weighted number of late jobs on a single machine. *Naval Research Logistics* 50(3):273–288.

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

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders decomposition. *Mathematical Programming* 96:33–60.

Jouglet, A.; Baptiste, P.; and Carlier, J. 2004. Branch-and-bound algorithms for total weighted tardiness. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall / CRC. chapter 13.

Jouglet, A.; Savourey, D.; Carlier, J.; and Baptiste, P. 2008. Dominance-based heuristics for one-machine total cost scheduling problems. *European Journal of Operational Research* 184:879–899.

Karlof, J. K., and Wang, W. 1996. Bilevel programming applied to the flow shop scheduling problem. *Computers and Operations Research* 23(5):443–451.

Kis, T., and Kovács, A. 2009. On bilevel machine scheduling problems. *Submitted to Algorithmica*.

Labbé, M.; Marcotte, P.; and Savard, G. 1998. A bilevel model of taxation and its application to optimal highway pricing. *Management Science* 44(12):1608–1622.

Lukač, Z.; Šorić, K.; and Rosenzweig, V. V. 2008. Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research* 187:1504–1512.

Marcotte, P.; Mercier, A.; Savard, G.; and Verter, V. 2009. Toll policies for mitigating hazardous materials transport risk. *Transportation Science* 43(2):228–243.

M'Hallah, R., and Bulfin, R. 2007. Minimizing the weighted number of tardy jobs on a single machine with release dates. *European Journal of Operational Research* 176:727–744.

Pan, Y., and Shi, L. 2005. Dual constrained single machine sequencing to minimize total weighted completion time. *IEEE Transactions on Automation Science and Engineering* 2(4):344–357.