J Sched (2009) 12: 281–298
DOI 10.1007/s10951-008-0079-3

Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm

Haitao Li · Keith Womer

Received: 1 May 2007 / Accepted: 4 July 2008 / Published online: 3 September 2008

Abstract We study an assignment type resource-constrained project scheduling problem with resources being multi-skilled personnel to minimize the total staffing costs. We develop a hybrid Benders decomposition (HBD) algorithm that combines the complimentary strengths of both mixed-integer linear programming (MILP) and constraint programming (CP) to solve this *NP-hard* optimization problem. An effective cut-generating scheme based on temporal analysis in project scheduling is devised for resolving resource conflicts. The computational study shows that our hybrid MILP/CP algorithm is both effective and efficient compared to the pure MILP or CP method alone.

Keywords Resource-constrained project scheduling · Multi-skilled personnel · Hybrid MILP/CP algorithms · Benders decomposition

1 Introduction

Scheduling and assignment of multi-skilled personnel to perform a project is of practical importance, as cross-training is becoming an viable practice to save operations costs and to justify technology investment in many organizations, e.g., call centers (Aksin et al. 2006). We consider the following project scheduling problem with multi-skilled personnel as resource constraints. A project consists of a set of tasks which must be completed by a given deadline. Generalized temporal constraints include due dates, minimum and maximum time lags. (A precedence constraint can be viewed as a special case of a minimum time lag constraint.) The project involves a set of skills. Each task may require multiple skills simultaneously in order for the task to progress. Each skill requires one individual selected from a pool of personnel that possess the skill. An individual may be able to perform multiple skills, but only one at a time. The assigned workload for each person cannot exceed the individual's maximum workload capacity during the scheduling horizon. The objective is to minimize the staffing costs subject to the generalized temporal relations, resource constraints, and the deadline on the project completion time (*makespan*). As a generalization of the single-mode resource constrained project scheduling (RCPSP), the project scheduling problem with multi-skilled personnel is strongly *NP-hard*.

We develop a hybrid mixed-integer linear programming (MILP) and constraint programming (CP) solution approach relying on the classical Benders decomposition (Benders 1962) for handling complex mixed-integer programs. The original project scheduling problem with multi-skilled personnel is decomposed into a relaxed master problem (RMP) containing only the assignment variables and constraints, and a feasibility subproblem containing only the scheduling variables and constraints. CP is used for solving the scheduling subproblem to infer "cuts" that are added iteratively into the RMP to exclude infeasible assignments. MILP methods such as branch–and–bound and branch–and–cut are used to solve the master problem. Such a hybrid Benders decomposition (HBD) framework was first proposed by Jain and Grossmann (2001) to solve a class of scheduling problem similar to the open-shop multi-purpose machine (OMPM, Brucker 2001) scheduling problem, where temporal constraints are restricted to release- and due-date constraints

H. Li (✉) · K. Womer
College of Business Administration, University of Missouri—St. Louis, One University Blvd., St. Louis, MO 63121, USA
e-mail: lihait@umsl.edu

K. Womer
e-mail: womerk@umsl.edu

and only the general "no-good" cuts are generated during the solving process. Due to the generality and complexity of our proposed problem, we devise a cut-generating scheme based on temporal analysis in project scheduling to establish a link between the assignment master problem and scheduling subproblem. This insight is the key to the practical effectiveness and efficiency of our algorithm.

In general, the model and solution approach presented in this paper can be used in two ways. First, with the personnel's skill-mix given, the model optimizes the *short-term assignment and scheduling decisions* for accomplishing projects. Second, with the personnel's skill-mix unknown, the model can be used to obtain an optimal skill-mix of personnel for *planning decisions* in the *intermediate-term*. We refer to Li and Womer (2006a) for an application of determining the optimal crew composition and reducing crew size.

The paper is organized as follows. Section 2 introduces the project scheduling problem with multi-skilled personnel and presents its MILP formulation. Section 3 provides an introduction to constraint programming and surveys the hybrid MILP/CP approaches in the current literature. In Sect. 4, we develop our hybrid MILP/CP Benders decomposition algorithm for solving the proposed problem. Section 5 presents the computational results. In Sect. 6, we examine the relationship between our model/algorithm and planning problems. Section 7 gives conclusions and discusses future research opportunities.

2 Project scheduling with multi-skilled personnel

In this section, we first describe the project scheduling problem with multi-skilled personnel and provide an illustrative example. We then present MILP formulation of the problem which our decomposition algorithm depends on. Additional remarks are made on the related models and past work.

2.1 Problem description

We let J be the set of tasks in the project, K be the set of relevant skills and S the personnel set required for executing the project. The set of skills required by task j is denoted by K^j. The set of people who are able to perform skill k is represented by S^k. Each skill must be assigned to one person who is able to perform that skill. Each task j has a constant processing time p_j. The minimum delay between task j and j' is $\delta_{jj'}$, i.e., task j' cannot start until $\delta_{jj'}$ time units after j starts. The due date for task j is d_j. There is a work load capacity of w_s for each individual $s \in S$. The deadline on the *makespan* of the project is $\bar{T}$. The objective is to minimize the total staffing costs to perform the project given a salary of c_s paid to an individual $s \in S$. We additionally make the following assumptions:

(A1) Simultaneous skill requirement: the required skills must be present simultaneously for a task to progress.
(A2) Single skill performance: personnel are treated as *unary resources*, meaning an individual can only perform one skill at each time point.
(A3) A task cannot be interrupted, i.e., no preemption is allowed.
(A4) Setup times are not considered explicitly.

We present a numerical example to illustrate the project scheduling problem with multi-skilled personnel. The software development department in a small firm faced a scheduling problem involving skilled labor. In the past, the department operated as a job shop with projects (jobs) arriving at irregular intervals with varying work requirements. But the department supervisor soon found it necessary to remodel the operational process to improve efficiency. Six categories of skills are needed for the department members: programming, algorithms, mathematical modeling, database, systems analysis, and quality control. Staff members in the department are all cross-trained and possess multiple skills. But an individual can only perform one skill at one time point. Projects arrived with varying requirements for skills and could be broken down into subtasks. Table 1 outlines members in the department and their skill-mix.

An upcoming project is to develop software for the ABC Company. It can be broken down into five tasks, each of which requires a specific set of skills simultaneously for the task to progress. The skill requirements of the project are presented in Table 2.

Additional descriptions of the tasks are provided below:

(J1) Problem identification. It defines the problem faced by the client while considering the technical capacity of the developing group.
(J2) Model development. It sets up the underlying mathematical programming model for identified optimization problem.
(J3) Algorithm design. It develops algorithms for solving the proposed model.
(J4) GUI design. It designs a user-friendly interface allowing easy accessibility.
(J5) Testing. It includes testing and debugging the application software to assure quality.

Table 3 gives the processing time of each task.

Considering the technical requirements among these tasks, the department supervisor believed that project makespan could be significantly shortened if generalized temporal constraints are introduced in addition to the current simple precedence relations as in the job shop scheduling environment. This is achieved by allowing simultaneously executions of some tasks (e.g., the minimal time lags). The temporal constraints he can think of are listed below:

Table 1 Department members and their skill-mix

Members	Skills					
	Programming	Algorithms	Mathematical modeling	Database	System analysis	Quality control
S1	√	√	√	√		√
S2	√			√	√	√
S3	√		√	√		√
S4		√			√	
S5	√	√	√		√	√
S6	√		√			√
S7		√		√		√
S8	√		√	√	√	
S9				√	√	√

Table 2 Skill requirements of the subtasks

Tasks	Skills					
	Programming	Algorithms	Mathematical modeling	Database	System analysis	Quality control
J1			√		√	
J2		√	√		√	
J3	√	√	√			
J4	√			√	√	
J5	√	√			√	√

Table 3 Processing times of the subtasks (in weeks)

Task	J1	J2	J3	J4	J5
Processing time	4	5	4	6	8

Table 4 Workload capacity of each individual (in weeks)

Personnel	S1	S2	S3	S4	S5	S6	S7	S8	S9
Capacity	22	26	26	20	25	26	26	25	26

(i) Model Development can only start at least 2 weeks after Problem Identification starts. (At least two weeks after Problem Identification starts, some initial feedback can be delivered to the following task (model development), allowing it to start right away, without necessarily waiting until the Problem Identification task finishes.)
(ii) Model Development must precede GUI design.
(iii) Algorithm Design can only start at least 3 weeks after Model Development starts.
(iv) GUI design must precede Testing.
(v) Algorithm design must precede Testing.
(vi) Problem Identification must be finished within 7 weeks after the project starts.
(vii) Model Development must be finished within 14 weeks after the project starts.
(viii) The entire project must be completed within 26 weeks after the project starts.

Furthermore, the workload capacity of each individual is given in Table 4. We also assume that they have the same salary.

The supervisor's decision problem consists of finding a schedule for all the tasks and assignments of personnel to tasks/skills so that the total staffing costs for the incoming project is minimized.

An optimal solution to the above problem is illustrated in Fig. 1. It can be observed that the project will be completed within the deadline of 26 weeks and all the temporal constraints have been satisfied. The Gantt chart also shows the assignment (staffing) decisions. Notice that only 5 people are needed in the least-cost solution while satisfying all the task-skill requirements and personnel-skill constraints.

2.2 MILP formulation

We define the following decision variables:

- $z_s = 1$, if and only if s is selected to perform the project, $\forall s \in S$.
- $x_{jks} = 1$, if and only if s is assigned to skill k in task j, $\forall j \in J, k \in K^j, s \in S^k$.
- $t_j \geq 0$, the starting time of task j, $\forall j \in J$.

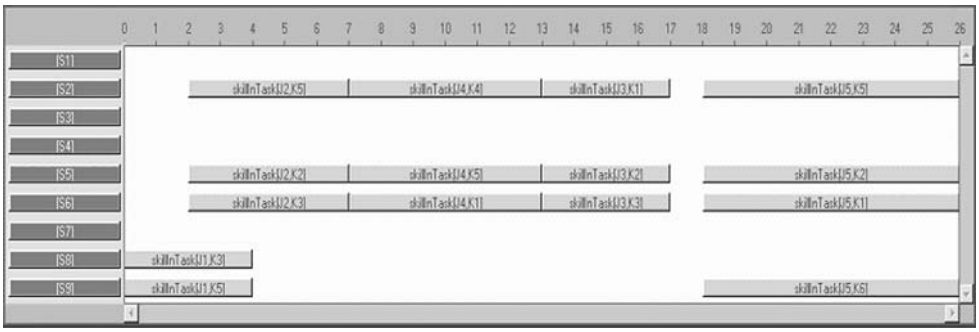

Fig. 1 Gantt chart of an optimal solution to the example problem

- $y_{jj'} = 1$, if and only if task j precedes j', $\forall (j, j') \in J \times J$.

The MILP formulation of the project scheduling problem with multi-skilled personnel can be written as:

$$\min \sum_{s \in S} c_s z_s \tag{1}$$

subject to:

$$\sum_{s \in S^k} x_{jks} = 1, \quad \forall j \in J,\ k \in K^j; \tag{2}$$

$$\sum_{k \in K^j} x_{jks} \le 1, \quad \forall j \in J, s \in S; \tag{3}$$

$$\sum_{j \in J} \sum_{k \in K^j} p_j x_{jks} \le w_s z_s, \quad \forall s \in S; \tag{4}$$

$$t_{j'} - t_j \ge \delta_{jj'}, \quad \forall (j, j') \in J \times J; \tag{5}$$

$$t_j + p_j \le \min\{d_j, \bar{T}\}, \quad \forall j \in J; \tag{6}$$

$$y_{jj'} + y_{j'j} \ge x_{jks} + x_{j'k's} - 1,$$
$$\forall \text{ ordered } (jk, j'k'),\ \forall s \in S; \tag{7}$$

$$t_{j'} \ge t_j + p_j - M(1 - y_{jj'}), \quad \forall j \ne j'; \tag{8}$$

$$z_s, x_{jks}, y_{jj'} \in \{0, 1\};$$

$$t_j \ge 0.$$

The objective function (1) minimizes the total staffing costs for executing the project. Constraints (2) through (4) model the assignment aspect of the problem. Constraint (2) states that each skill in a task requires one person who possesses that skill. Constraint (3) enforces that no individual is assigned to more than one skill in the same task due to assumptions (A1) and (A2). Constraint (4) ensures that the total assigned workload of each person cannot exceed his/her workload capacity. Constraints (5) through (8) take care of the generalized temporal relations, i.e., the scheduling aspect of the problem. When $\delta_{jj'} \ge 0$, constraint (5) represents a minimum time lag between task j and j'; when $\delta_{jj'} < 0$, (5) represents a maximum time lag between task j' and j; when $\delta_{jj'} = p_j$, (5) reduces to a precedence constraint. Constraint (6) satisfies the due date of each task as well as the deadline on the makespan. Constraint (7) states the logic relationship between sequencing and assignment variables, i.e., if two tasks are assigned with the same person then these two tasks cannot overlap (one sequencing relation must be determined) due to assumption (A2). Constraint (8) is the classical big-M formulation in disjunctive programming to define the sequencing variables. For a two-layer resource structure to characterize the problem, we refer to Li and Womer (2006b).

2.3 Additional remarks

The project scheduling problem with multi-skilled personnel studied in this paper is a generalization of the single-mode resource-constrained project scheduling problem (RCPSP) by allowing each task to be performed in multiple ways. We refer to Brucker et al. (1999) for a survey of various RCPSP models. It also belongs to the assignment type RCPSP (Drexl et al. 1998) due to the presence of both the assignment and sequencing variables. It is similar to the multi-mode RCPSP in that each task has multiple ways (modes) to be performed. To be specific, for task j there are $\prod_{k \in K^j} |S^k|$ ways to be considered, which makes the problem size increase explosively if modeled by a standard multi-mode RCPSP. It is also an extension of the multi-purpose machine scheduling problem (MPM, Brucker 2001), due to the presence of the generalized temporal constraints (5) in addition to the usual precedence constraints in the classical machine scheduling problems. This paper extends the work of Li and Womer (2006b) by minimizing the total staffing cost instead of the number of selected personnel. This is not a trivial generalization; we will show in Sect. 3.2 that the Skill-Level Based Decomposition (SLBD) algorithm proposed by Li and Womer (2006b) will not be able to handle the objective function considering the cost of selecting each individual.

Other work on project scheduling problems with multi-skilled personnel include Focacci et al. (2000), Bellenguez and Neron (2004) and Neron et al. (2006). Their models, however, treat a set of personnel merely as *unary resources*,

while the capacity associated with each individual was not considered. In addition, temporal constraints in their studies are restricted to precedence constraints, i.e., minimum or maximum time lag is not considered. Moreover, they all seek to minimize the project makespan instead of cost-related objective functions.

3 CP-based hybrid approaches

3.1 Constraint programming

Constraint programming is the study of computational systems based on constraints. It originated in the Artificial Intelligence areas that investigate the Constraint Satisfaction Problem (CSP, Tsang 1993) and Logic Programming (Van Hentenryck 1999). The main solving technologies of CP include constraint propagation and search. The basic idea of constraint propagation is that when a variable's domain is modified, the effects of this modification are then communicated to any constraint that interacts with that variable. In other words, the domain reduction algorithm modifies the domain of all the variables in that constraint, given the modification of one of the variables in that constraint. We refer to Tsang (1993) for a detailed description of various general-purpose constraint propagation algorithms and Baptiste et al. (2001) for efficient constraint propagation algorithms for scheduling problems. The domain of each variable in an optimization problem can be reduced through constraint propagation. However, reducing a problem to a minimum problem, i.e., no more redundant values can be removed from the domain of the problem is often *NP-hard*. This is why a search procedure is often needed to explore the remaining solution space. Most popular search strategies include depth-first (DF), best-first (BF), and limited discrepancy search (LDS, Harvey and Ginsberg 1995).

The CP techniques can be compared and contrasted with MILP. Hooker (2002) pointed out that the differences and complementary strengths between the two techniques indicate the opportunity for integrating; while the commonalities often make the integration natural and easier. Two areas of differences can be observed between CP and MILP.

First, from the modeling perspective CP's declarative nature can make the model expressive and compact with fewer variables and constraints when compared with the MILP formulation. This is especially true when modeling scheduling problems. In most of the CP software such as ILOG Solver (ILOG 2002a), ECLiPSe (Wallace et al. 1997), and CHIP (Dincbas et al. 1988), there are special constructs available to model scheduling constraints in an expressive and compact way. In contrast, the modeling power of MILP has been greatly hampered by the restrictiveness of linear expressions for handling the temporal and resource constraints encountered in scheduling problems. The disjunctive formulation of these constraints often involves an explosive number of choice variables and constraints as evident in constraint (7) and (8) in Sect. 2.2

Second, from the algorithmic perspective CP solves an optimization problem through a naïve branch–and–bound method by gradually tightening a bound on the objective function. For a minimization problem with an objective function $f(x)$, each time a feasible solution $\bar{x}$ is found, a constraint $f(x) < f(\bar{x})$ is added to the constraint store of each subproblem in the remaining search tree. It is not efficient to do this in practice, since there is no sophisticated relaxation algorithm in CP to obtain tight bounds and the link between the objective function and the decision variables is quite loose (Milano and Trick 2004). In MILP, however, various relaxation methods have been well developed (Nemhauser and Wolsey 1988). Hence, the success of CP depends largely on the effectiveness of constraint propagation; whilst in MILP, the quality of relaxation plays an important role. Thus, the solution procedure may benefit from incorporating bounding information obtained by relaxations techniques in MILP to prune the solution space and accelerate search. Sometimes a problem has some characteristics better handled by CP while others are better handled by MILP. For such problems neither MILP nor CP alone may perform well.

3.2 Hybrid approaches

Various integration schemes have been proposed to take advantage of the complementary strengths of CP and MILP for solving combinatorial optimization problems. These schemes can be generally classified into two categories: preprocessing and hybrid approaches. Constraint propagation has been used as a preprocessing tool to reduce the problem size of an RCPSP by Brucker and Knust (1998). CP-based lower bounds for both the single-mode and multi-mode RCPSP have also been studied by Brucker and Knust (2000, 2003).

As for hybrid approaches, Bockmayr and Kasper (1998) presented a unifying framework called *branch–and–infer*, in which constraints for both MILP and CP are divided into two categories, primitive and non-primitive. They discussed how non-primitive constraints could be used to infer primitive constraints. Hooker and Osorio (1999) proposed a mixed logic/linear modeling framework which is further treated in detail by Hooker (2002). Hybrid approaches often depend on decompositions and allow close communications between the MILP and CP solvers. Different hybrid decomposition schemes include CP-based branch–and–price (Easton et al. 2004), CP based Lagrangian relaxation (Benoist et al. 2001 and Sellmann and Fahle 2003) and CP-based Benders decomposition (Benoist et al. 2002 and Eremin and Wallace 2001). Notably, Jain and Grossmann

(2001) proposed a hybrid Benders decomposition (HBD) algorithm to solve a class of scheduling problem similar to the open-shop multi-purpose machine (OMPM, Brucker 2001) scheduling problem, where temporal constraints are restricted to release- and due-date constraints. The computational results on the same problem were improved by Thorsteinsson (2001) through a framework called *branch–and–check* where the master problem is not solved to optimality but halts once a feasible solution is found, and gradually tightens the objective function value. Jain and Grossmann's cut-generating scheme relies largely on the fact that resource units (machines) are independent in the OMPM, so that cuts could be generated for each individual machine independently. For more general scheduling problems, such as the project scheduling problem with multi-skilled personnel studied in this paper, it is not possible to infer cuts for each person independently as the skilled personnel are interrelated through precedence or time lag constraints. In addition, the cuts generated by Jain and Grossmann (2001) are simple "no-good" cuts, which could be rather weak when they correspond with the entire set of resource units. It is a challenge to infer effective cuts when resource units are interrelated with each other. According to the authors' best knowledge, the HBD approach has not be implemented on solving scheduling problems with precedence or time lag constraints.

Li and Womer (2006b) developed a combined MILP/CP decomposition heuristic called Skill Level Based Decomposition (SLBD) to heuristically solve a similar problem with the objective function minimizing the number selected of personnel. The SLBD works in such a way that a scheduling subproblem is solved first to obtain a feasible schedule, followed by a resource leveling procedure that smoothes the resource utilization, and finally an assignment problem is solved to obtain a feasible solution to the original problem. The skill level, i.e., the skill availability for the scheduling subproblem, is controlled as low as possible to obtain a feasible schedule for the succeeding assignment phase. A more elaborate procedure based on tabu search was implemented by Li and Womer (2006a) in the resource leveling phase to smooth the resource utilization. However, since the objective function (1) may not be a monotonic function of the number of selected personnel, the SLBD is unable to handle the objective function with staffing costs considered in this paper, although it has shown significant advantages over the pure MILP or CP approach alone in both solution quality and speed.

4 The hybrid Benders decomposition algorithm

4.1 Hybrid Benders decomposition

The classical Benders decomposition (Benders 1962) is a method for solving optimization problems with enormous numbers of constraints. As the dual of the column generation method, the strategy here is to generate rows or constraints and successively add them into the constraint system. Cuts (constraints) generated based on duality theory are called "Benders cuts".

Following the hybrid MILP/CP Benders decomposition (HBD) framework by Jain and Grossmann (2001), we decompose the original project scheduling problem with skilled-personnel into a relaxed master problem (RMP), which contains only assignment decision variables, and a feasibility subproblem (SP) which takes care of the scheduling aspect of the problem. We rely on MILP methods such as branch–and–bound and branch–and–cut to solve the RMP. We use CP to model the scheduling SP which would otherwise be difficult for MILP to model and apply CP-based algorithms to solve the feasibility scheduling SP, for which efficient constraint propagation algorithms are available.

We let the assignment binary variable x_{ijk} and selection binary variable z_s be associated with the RMP. The rest of the variables related to the scheduling part of the problem, i.e., the starting time of each task t_j and the sequencing variable $y_{jj'}$ will be replaced by CP variables to construct the scheduling subproblem. All the submodels are constructed in OPL, a modeling language supporting both linear programming and constraint programming (Van Hentenryck 1999).

The MILP formulation of the RMP at iteration n can be written as:

RMP_MILP(n)

$$
\begin{aligned}
&= \{ \\
&\quad \min \sum_{s \in S} c_s z_s \\
&\quad \text{subject to:} \\
&\qquad \text{constraints (2), (3), and (4)} \\
&\qquad \beta_i(X) \le 1, \quad \forall i = 1, \ldots, n. \qquad (9) \\
&\quad \}
\end{aligned}
$$

Constraints (9) include cuts generated at each iteration i. Each cut prevents a pair of overlapping activities from being assigned to the same individual. The overlapping activities are identified through temporal analysis techniques in project scheduling, which will be explained in detail in Sect. 4.3.

For a partial solution (X^n, Z^n) from solving the RMP at iteration n, we define a scheduling feasibility subproblem modeled by CP. By fixing the assignment variables (X^n, Z^n), the subproblem SP decides if this partial solution can be extended to a complete solution. We define the set of available personnel S as an *array* of *unary resources* called

SkilledPersonnel:

UnaryResource SkilledPersonnel [S].

Then the CP formulation of the feasibility SP at iteration n can be written as below, where the italic words are keywords in OPL

SP_CP(n)

$=\{$

Solve

subject to:

$$j \text{ precedes } j', \quad \forall (j, j') \in P; \tag{10}$$

$$j.start + \delta_{jj'} \le j'.start, \quad \forall (j, j') \in J \times J \backslash P; \tag{11}$$

$$j.start + p_j \le \min\{d_j, \bar{T}\}, \quad \forall j \in J; \tag{12}$$

$$\text{if } x^i_{jks} = 1, \text{ then } j \text{ requires SkilledPersonnel}[s], \quad \forall i = 1, \ldots, n; j \in J, k \in K^j, s \in S. \tag{13}$$

$\}$

Constraints (10) through (12) take care of the temporal constraints. CP provides a descriptive way to express the precedence constraint as in (10), where P represents the set of precedence constraints. Constraints (11) and (12) are equivalent with (5) and (6), respectively, where *j.start* refers to the starting time of activity j. Constraint (13) establishes the link between assignment variables and resource constraints. Since the values of x^i_{jks} are fixed through solving the RMP_MILP(i) at iteration i, i.e., the assignment of tasks/skills to the skilled personnel has been made, the original complex assignment-type scheduling problem reduces to a pure single-mode RCPSP with *unary resources* for which efficient constraint propagation algorithms exist (Baptiste et al. 2001). Also notice that comparing with the pure CP formulation in Li and Womer (2006b), here an activity is defined at the task level instead of the skill level, hence the number of activities in the subproblem has been reduced. In addition, SP_CP(n) is a feasibility problem instead of an optimization problem. That is, it only needs to determine whether a feasible solution exists or not, a problem for which CP is relatively efficient to handle.

The way to generate cuts is often the key to success of a Benders decomposition algorithm. A main feature of our HBD is to obtain the possible causes of infeasibility ex ante, i.e., before the main algorithm iterations start. The role of solving the scheduling subproblem is to infer (trigger) the violated constraints. The causes of infeasibility to the subproblem can be interpreted a priori as "two overlapping activities are assigned to the same individual" due to assumption (A2). Theoretically, we could obtain all pairs of overlapping activities and include the complete set of constraints (9) in the RMP. However, it is impractical to do so because of the enormous number of such constraints and also the fact that only a small fraction of such constraints are binding at an optimal solution (Lasdon 1970). Hence, the causes of infeasibility are inferred as cuts, which are added iteratively into the RMP. Next we describe our cut generating schemes in detail.

4.2 Generating cuts

We first introduce the concept of *minimal forbidden set* in the resource-constrained project scheduling literature. A *forbidden set* represents a set of activities whose total requirement for some resource exceeds the available capacity of the resource (Bartusch et al. 1988). A set of activities $F \subseteq A$ is called a *forbidden set* if

$$\sum_{j \in F} r_{jk} > Q_k \quad \text{for some } k \in R,$$

where R refers to the set of resources, Q_k refers to the available capacity of resource k, and r_{jk} denotes the requirement of resource k by activity j. If no proper subset $F' \subset F$ is forbidden, we call F a *minimal forbidden set*. For a unary resource such as each individual $s \in S$ in our model, a *minimal forbidden set* always contains two elements, e.g., $F_s = \{j, j' | s \text{ is assigned to both } j \text{ and } j'\}$. To resolve the resource conflicts caused by F_s, a precedence constraint j precedes j' (or j' precedes j) needs to be introduced to "break up" F_s (to prevent j and j' from being executed simultaneously). We have the following observations concerning the resulting scheduling problem with the newly added precedence constraint:

Observation 1 When j and j' *must overlap* in order to satisfy the temporal constraints (3) and (4) (denoted by $j \| j'$), the resulting scheduling problem will not be feasible.

Observation 2 When j and j' *never overlap* in order to satisfy the temporal constraints (3) and (4) (denoted by $j \sim j'$), the resulting scheduling problem will always be feasible.

Observation 3 When neither (O1) nor (O2) occurs, i.e., j and j' are *possible to overlap* (denoted by $j | j'$), the resulting scheduling problem may or may not be feasible.

Observation 1 indicates that the only way to resolve infeasibility is to prevent j and j' from being assigned with the same person, which generates a valid cut (global cut) for the RMP. Observation 2 implies that no cut will be generated (or generating a redundant cut) for the RMP, as it is always feasible to assign the same person to j and j'. Even though the cut generated through Observation 3 may resolve resource conflicts, it will not be a valid cut when the resulting scheduling problem is feasible.

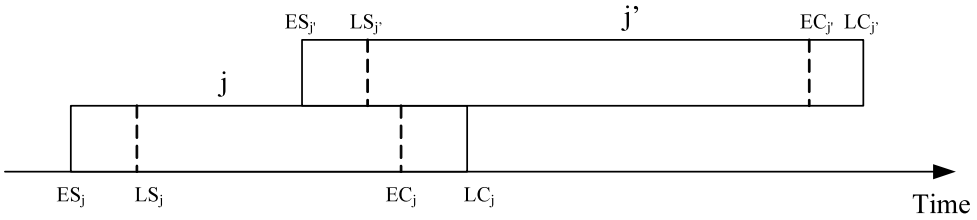

Fig. 2 Identify a pair of (j, j') satisfying $j \| j'$

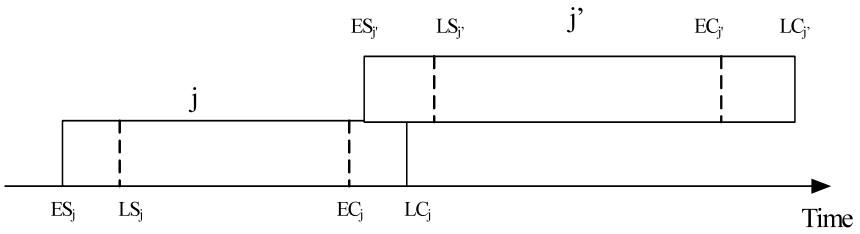

Fig. 3 Identify a pair of (j, j') satisfying $j | j'$

4.2.1 Global cuts

Global cuts are constraints that prevent a pair of overlapping activities $j \| j'$ from being assigned with the same person and must be satisfied by all feasible solutions to the optimization problem. The following proposition is crucial.

Proposition 1 *The set C^g of global cuts consisting of constraints taking the form*: $x_{js} + x_{j's} \leq 1$ *where* $j \| j', s \in S$, *are valid cuts*.

Proof Suppose that there exists a feasible solution that does not satisfy the above inequality, i.e., $x_{js} + x_{j's} = 2$ and $j \| j'$. This is to say that at some time point s has to perform both j and j' simultaneously, which violates assumption (A2), a contradiction. Thus, such a feasible solution does not exist. Therefore, $x_{js} + x_{j's} \leq 1$ where $j \| j'$ is a valid cut. □

We use earliest start (ES), earliest completion (EC), latest start (LS), and latest completion (LC) times to define the time window of an activity. Figure 2 shows how to identify a pair of (j, j') satisfying $j \| j'$ and leads to the following observation.

Observation 4 If $p_j + p_{j'} > \max\{\mathrm{LC}_j, \mathrm{LC}_{j'}\} - \min\{\mathrm{ES}_j, \mathrm{ES}_{j'}\}$ holds, then $j \| j'$.

We identify all pairs of (j, j') satisfying $j \| j'$ using the inequality test in Observation 4.

4.2.2 Trial cuts

Trial cuts are constraints that prevent a pair of possible-overlapping activities $j | j'$ from being assigned with the same person. We state and prove the following proposition.

Proposition 2 *The set C^t of trial cuts consisting of constraints taking the form*: $x_{js} + x_{j's} \leq 1$ *where* $j | j', s \in S$, *are not valid cuts*.

Proof Consider the scheduling feasibility problem with only constraints (3) and (4). Since $j | j'$, there exists a feasible schedule ξ that satisfies all the temporal constraints without overlapping j and j'. Then we solve an assignment subproblem given the feasible schedule ξ while enforcing $x_{js} + x_{j's} = 2$ (assigning s to both j and j') and are always able to obtain a feasible complete solution that does not satisfy $x_{js} + x_{j's} \leq 1$. Hence, $x_{js} + x_{j's} \leq 1$ where $j | j', s \in S$, is not a valid cut. □

Figure 3 illustrates how to identify a pair of (j, j') satisfying $j | j'$ and leads to the following observation.

Observation 5 If $p_j + p_{j'} \leq \max\{\mathrm{LC}_j, \mathrm{LC}_{j'}\} - \min\{\mathrm{ES}_j, \mathrm{ES}_{j'}\}$ and

$$\mathrm{ES}_{j'} - \mathrm{ES}_j < p_j \quad \text{and} \quad \mathrm{ES}_j - \mathrm{ES}_{j'} < p_{j'} \quad \text{hold},$$

then $j | j'$.

We identify all pairs of (j, j') satisfying $j | j'$ using the inequality test in Observation 5.

4.2.3 Infer cuts

For a partial solution (X^i, Z^i) obtained form solving RMP(i) at iteration i, cuts in the global cut set C^g are checked first to see if they are violated. Those violated global cuts are then added into the RMP and the next iteration starts. If none of the global cuts is violated, we check the trial cut set C^t and add any violated trial cuts into the RMP. The procedure for inferring cuts is presented below.

Procedure 1: ***InferCuts*** (X^i)
Step 1. Initialization: **Set** numGlobalCuts := 0
Step 2. Infer global cuts:
For $(j, j') : j \| j'$
  For $k \in K^j$ **and** $k' \in K^{j'}$
    For $s \in S$
      If $x^i_{jks} = 1$ **and** $x^i_{j'k's} = 1$, **then**
        Add a global cut: $x_{jks} + x_{j'k's} \leq 1$ into RMP(i)
        Set numGlobalCuts := numGlobalCuts + 1
Step 3. Infer trial cuts:
If numGlobalCuts = 0, **then**
  For $(j, j') : j | j'$
    For $k \in K^j$ **and** $k' \in K^{j'}$
      For $s \in S$
        If $x^i_{jks} = 1$ **and** $x^i_{j'k's} = 1$, **then**
        Add a trial cut: $x_{jks} + x_{j'k's} \leq 1$ into RMP(i)
End Procedure

The inferred cuts not only cut off the current partial solution, but also eliminate partial solutions with similar assignment decisions. It is important to notice that we try to avoid trial cuts whenever possible, i.e., *Step 3* is executed *only if* no global cut is triggered. The main reason for this is that by adding only global cuts (valid cuts) we maintain the algorithm's ability to prove optimality; otherwise, if any trial cut (non-valid cut) is triggered, the algorithm will no longer be able to prove optimality.

It should be stressed that the cuts generated in this way has incorporated problem-specific information gathered from the scheduling aspect of the original problem and is believed to be tighter than the general "no-good" cuts (Hooker 2000). The computational results indicate that our cut generating procedure works quite effectively requiring only 10 iterations on average to solve the set of 128 test instances.

4.3 Temporal analysis

To obtain the global cut set C^g and trial cut set C^t, we apply temporal project scheduling techniques to identify time windows of the activities. Following Neumann et al. (2002), the earliest starting time ES_j for activity j can be found by solving the following linear program:

ES_LP

$$= \{$$

$$\min \sum_{j \in J} t_j \tag{14}$$

subject to:

constraints (5) and (6);

$$t_0 = 0; \tag{15}$$

$$t_j \geq 0.$$

$$\}$$

Constraint (15) sets the start of the dummy start activity to be zero. To find the latest starting time LS_j, simply replace (14) with a maximizing objective function:

LS_LP

$$= \{$$

$$\max \sum_{j \in J} t_j \tag{16}$$

subject to:

constraints (5), (6), and (15);

$$t_j \geq 0.$$

$$\}$$

Then the earliest and latest completion time EC_j and LC_j can be calculated as follows:

$$\mathrm{EC}_j = \mathrm{ES}_j + p_j, \quad \forall j \in J; \tag{17}$$

$$\mathrm{LC}_j = \mathrm{LS}_j + p_j, \quad \forall j \in J. \tag{18}$$

One may also consider an alternative approach based on solving the *transitive closure* of a *distance matrix* (Brucker and Knust 2000) through the well-known Floyd–Warshall algorithm with a time complexity of $O(|J|^3)$. Brucker (2002) introduced constraint propagation based techniques to refine the *distance matrix*, i.e., to further reduce the value of each entry in the matrix representing a minimum time lag.

We then use the inequality tests in Observations 4 and 5 to obtain C^g and C^t, respectively.

The following two propositions reveal the effect of project deadline on the cardinality of C^g and C^t.

Proposition 3 *The cardinality of C^g is a monotonically decreasing function of the project deadline $\bar{T}$.*

Proof Assume the project deadline becomes less restrictive, i.e., $\bar{T}' > \bar{T}$. It has no effect on the optimal solution of the ES_LP. The LS_LP, however, becomes less restrictive and the latest starting time of at least one activity becomes greater. Thus, the right-hand side of at least one inequality in Observation 4 becomes greater and more restrictive. Therefore, no more number of (j, j') pairs will satisfy the inequality test in Observation 4. Hence, $|C'^g| \leq |C^g|$. □

Proposition 4 *The cardinality of C^t is a monotonically increasing function of the project deadline $\bar{T}$.*

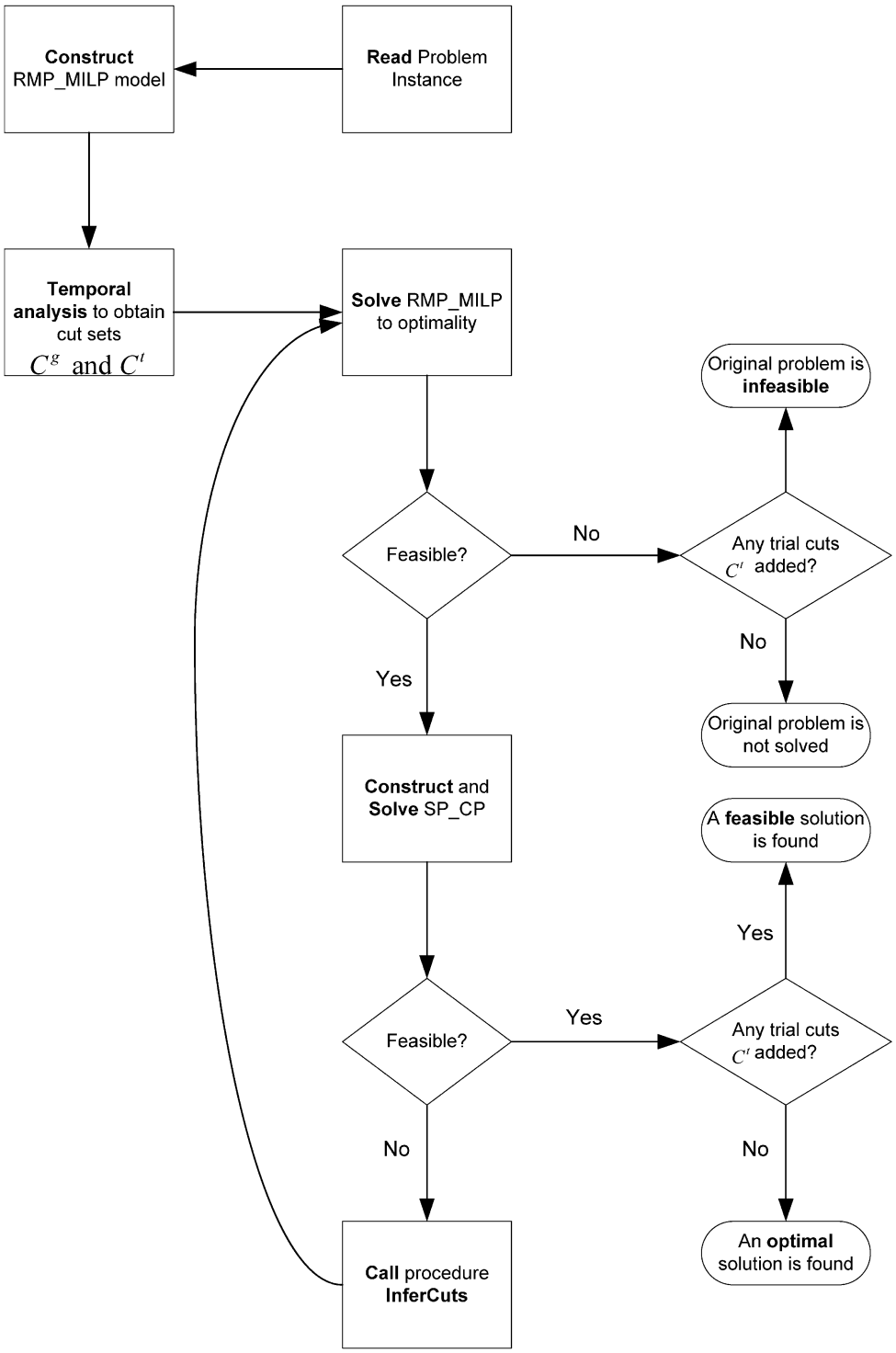

Fig. 4 Hybrid Benders decomposition algorithm

Proof Assume the project deadline becomes looser, i.e., $\bar{T}' > \bar{T}$. Following the similar logic in the proof of Proposition 3, at least one inequality test in Observation 5 becomes less restrictive. Then no less number of (j, j') pairs will satisfy the inequality test in Observation 4. Hence, $|C'^g| \geq |C^g|$. □

Both Propositions 3 and 4 will be verified by our computational experiments.

4.4 The HBD algorithm

The algorithm is summarized in Fig. 4. Each iteration starts by solving the RMP_MILP problem to optimality. If there

is no feasible solution found, then the original problem is infeasible or not solved with the given computational time. Otherwise, the obtained partial solution is used to construct the feasibility SP_CP problem, which attempts to determine if the partial solution can be extended to a complete solution that satisfies the scheduling constraints. If there exists such a complete solution, then this solution is an optimal solution given that no trial cut has been added into the master problem during the solution process. If the SP_CP is infeasible, the causes of infeasibility are inferred as cuts and added to the RMP_MILP model through the *InferCuts* (Procedure 1). The algorithm terminates when the partial solution can be extended to a complete feasible solution, or the RMP_MILP becomes infeasible.

Assuming that only global cuts from the cut set C^g have been added into the RMP_MILP, we state the following proposition concerning the convergence of our HBD algorithm. The proof follows a similar approach as the convergence proof in Jain and Grossmann (2001).

Proposition 5 *If only global cuts from C^g are added into the RMP_MILP, the HBD algorithm converges to the optimal solution or proves infeasibility in a finite number of iterations.*

Proof The assignment subproblem RMP_MILP is either feasible or infeasible as the domain of the decision variables x and z is bounded. We consider the following two cases.

(i) RMP_MILP is feasible.

The domain of RMP_MILP(n) reduces as n increases. Since the domain of x and z is finite, there exists an iteration ξ where the optimal solution (x^ξ, z^ξ) to the master problem leads to a feasible solution t^ξ to the scheduling subproblem SP_CP(ξ). Since global cuts in C^g do not exclude any feasible solution from the original problem according to Proposition 1, RMP_MILP(n) is a relaxation of the original problem. Then the solution (x^ξ, z^ξ, t^ξ) is an optimal solution to the original problem. Hence, the algorithm will converge to an optimal solution in a finite number of iterations.

(ii) RMP_MILP is infeasible.

If the relaxed master problem is infeasible, the subproblem will never obtain a feasible solution. Since the domain of x and z is finite and shrinks with each successive iteration, the master problem will become infeasible after a finite number of iterations. □

If any trial cut from the set C^t is incurred, Proposition 5 will no longer hold due to Proposition 2. Despite this potential loss of ability to prove optimality, trial cuts are still preferable to the general "no-good" cuts in Jain and Grossmann (2001). This is because the general "no-good" generated cuts in the context of interrelated personnel are rather weak, which leads to a slow convergence. Hence, we have to trade-off the ability to proof optimality with the efficiency of the algorithm.

Although the proposed decomposition algorithm may not be able to find optimal solution, it will not remove all feasible solutions from the solution space. Note that both the global and trial cuts restrict the assignment subproblem rather than scheduling subproblem. The format of trial cuts, preventing a person to be simultaneously assigned to a pair of jobs, serves as a heuristic to resolve the resource conflicts caused by the minimum forbidden sets. Practically speaking, it will be costly to examine all possible trial cuts to resolve resource conflicts, although such an exhaustive search is able to prove optimality from a theoretical perspective. The computational results, which will be presented next, suggest that our cut-generating scheme is both *efficient* as of computational time and *effective* as of solution quality.

Our cut-generating scheme also provides insights on the effectiveness of the hybrid Benders decomposition. The likelihood for the scheduling subproblem to be feasible depends largely on the time restrictiveness of the project. The more time-restrictive the project is, the less flexible is the scheduling subproblem to resolve resource conflicts and the more global cuts to infer (Proposition 3). We state the following conjecture.

Conjecture 1 *The original problem with a tighter project deadline requires more iterations for the HBD algorithm to solve.*

We further expect that the problem with a tighter project deadline to be more difficult to prove optimality due to the difficulty of resolving resource conflicts. That is, trial cuts are more likely to be incurred to resolve infeasibility, even though the size of the trial cut set is smaller (Proposition 4). We thus have Conjecture 2.

Conjecture 2 *The original problem with a tighter project deadline is more difficult for HBD to prove optimality.*

Both conjectures will be verified in our computational study presented next.

5 Computational study

A full factorial experimental design is used to conduct the computational study. We compare our HBD algorithm with the pure MILP and CP approach alone, as well as the SLBD algorithm proposed by Li and Womer (2006b). All the four approaches are implemented in ILOG OPL Studio 3.6.1 (ILOG 2002c), which uses CPLEX 8.1 (ILOG 2002b) to solve an MILP, ILOG Solver 5.3 (ILOG 2002a) to solve a CP. The computations are performed on a PC Pentium IV 2.4 GHz with 512 Mb RAM.

Table 5 Factors considered in the experimental design

Factor	Factor explanation	Value
|J|	The number of tasks in a project	{10, 30}
|K|	The number of skills relevant to a project	{4, 8}
|S|	The personnel availability	{Low, High}
RT	Restrictiveness of Thesen	{0.35, 0.65}
R	Multiplier of average number of skills required by a task	{0.3, 0.6}
M	Multiplier of average number of skills possessed by a person	{0.3, 0.6}
DF	Deadline factor	{Low, High}

5.1 Experimental design

Since our problem and the RCPSP share common data concerning the project network, we adopt instances of the RCPSP for our experiments. As our problem also involves minimal and maximal time lags we choose ProGen/max (Schwindt 1996) to generate the RCPSP instances. Seven control factors in our experiment and their explanations are listed in Table 5.

The problem size is directly affected by the number of tasks |J|, the number of skills |K|, and personnel availability |S|. In order to have meaningful comparisons, we have concentrated on the problem space where MILP and CP methods had a chance to remain competitive. In our experiment, |J| is chosen from the two-element set {10, 30} and |K| is chosen from {4, 8}. The low level of personnel availability |S| is chosen such that an instance could be feasible; and the high level of |S| is chosen approximately 50% higher than the low level of |S|. For example, if it is found that at least 60 people are needed for a project, we set the low level of |S| to be 60 and high level to be $60 + 60 * 50\% = 90$. The Restrictiveness of Thesen (RT, Thesen 1977) measures the complexity of a network and has been shown to have an even greater effect on the complexity of RCPSP than the number of tasks |J| (Schwindt 1996). In our experiment, RT is chosen from {0.35, 0.65}. The complexity of skill requirement is controlled by the multiplier R representing the average number of skills required by a task, which affects both the scheduling and assignment aspects of the problem. A low level of R is set to be around 0.3 and a high level around 0.6. Skill mix complexity is reflected by the multiplier M of average number of skills possessed by an individual, whose effect is mainly on the assignment problem. Likewise, a low level of M is set to be around 0.3 and a high level around 0.6.

An important factor we expect to have a significant impact on the algorithm performance is the deadline $\bar{T}$ on the project makespan. In order to verify Conjectures 1 and 2, we have set the lower level of $\bar{T}$ to be lowest possible as long as the problem remains feasible. $\bar{T}$ is calculated following Drexl and Kimms (2001):

$$\bar{T} = \mathrm{DF} \cdot \max \mathrm{EF}_j, \tag{19}$$

where EF_j is the earliest starting time of activity j.

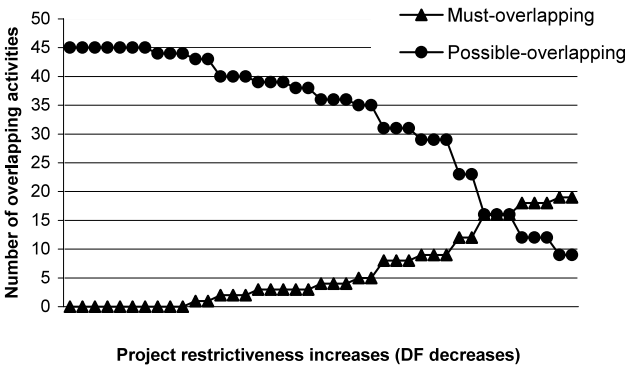

Fig. 5 Number of overlapping activities of a 10-task instance as project deadline decreases

In order to compare with the SLBD algorithm, we assume that each person is paid the same salary, so that the objective function reduces to minimizing the number of selected personnel to perform the project.

5.2 Computational results

A running time limit of 10 hours is imposed for all the four approaches. For the HBD, a limit of 10 seconds is imposed on the CP scheduling subproblem and 600 seconds on the master MILP problem. Among the 128 instances, two instances are infeasible which leaves a total of 126 feasible instances.

5.2.1 The effect of project deadline

We first conducted experiments to analyze the effect of project deadline on the number of must-overlapping (‖) and possible-overlapping (|) activities, i.e., the cardinality of C^g and C^t. Figure 5 illustrates such an experiment performed on a 10-task instance. As the project becomes more restrictive (the project deadline represented by DF decreases), the number of must-overlapping pairs increases, which supports Proposition 3 and the number of possible-overlapping pairs decreases, which supports Proposition 4.

Figure 6 illustrates the effect of project deadline on the performance of HBD on solving the set of 126 instances.

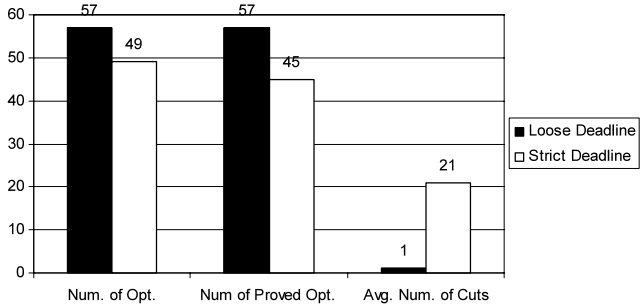

Fig. 6 The effect of project deadline on the performance of HBD

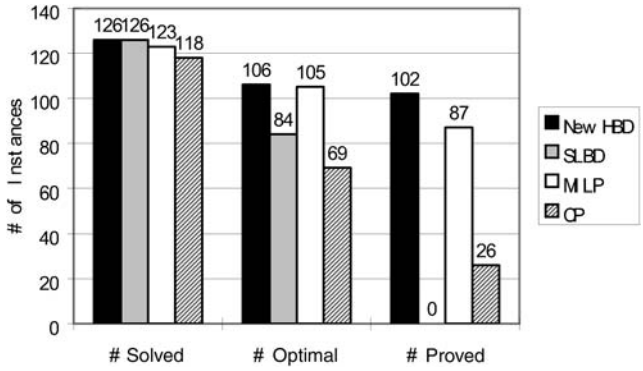

Fig. 7 Comparison of the overall computational results

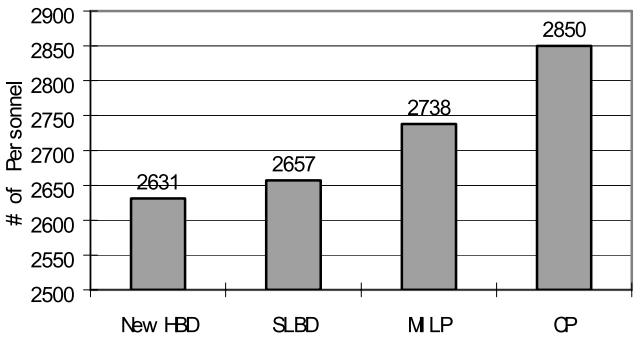

Fig. 8 Comparison of the sum of objective values

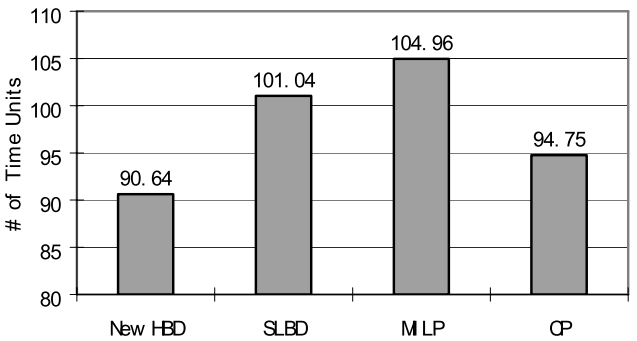

Fig. 9 Comparison of the average project *makespan*

As we have expected in Conjecture 1, on average the problem with a loose project deadline can be solved by the HBD algorithm with less effort (fewer cuts). Concerning the algorithm's ability to find optimal solutions and prove optimality, for the 63 instances with a loose deadline, 57 of them (90%) find optimal solutions and prove optimality (a success rate of 100%); for the other 63 instances with a strict deadline, 49 of them (77%) find optimal solutions and only 45 of the 49 optimal solutions have been proved to be optimal with a success rate of 92%. These results support Conjecture 2.

5.2.2 *Comparison of algorithm performance*

We now compare the overall performance of the four approaches.

As we can see from Fig. 7, only the two decomposition algorithms HBD and SLBD have found feasible solutions to all of the 126 problems. The HBD is able to find more optimal solutions than any of the other approaches. HBD is advantageous over the heuristic approach SLBD since there is no way for SLBD to prove optimality. Notably, HBD has an overall success rate of 96% to prove optimality (when an optimal solution is found) in contrast to 83% of MILP.

We further compare solution quality as shown in Fig. 8.

Clearly, the HBD outperforms the other three in objective function value, saving about one person over MILP and two persons over CP for each problem instance on average. The SLBD is the second best among the four with better solution quality than the pure MILP or CP approach alone.

Although project makespan does not appear in the objective function, desirably one prefers the schedule with a shorter makespan when it results in the same objective value as other solutions. Interestingly, the four approaches produce different project makespan in their best solutions found. The HBD provides solutions with the shortest makespan on average as indicated by Fig. 9. It excels by an average of 4 time units over CP, 11 time units over SLBD, and almost 14 time units over MILP. These could represent tremendous time savings on scheduling a project in the real world: the HBD finds schedules not only with less cost but also with shorter makespan. This merit of HBD can be attributed to its decomposition characteristic. The HBD decomposes the original problem into a pure GAP and a pure single-mode RCPSP, where the project makespan can be well handled by CP techniques. This also explains the fact that the pure CP approach is the second best in the quality of makespan. In contrast with HBD, the SLBD is doing the opposite by finding a feasible schedule first and then solving the assignment part. It always starts from the most restrictive skill level (lowest possible resource availability), which potentially leads to a longer makespan. From a different perspective, the SLBD can be viewed as a time-resource trade-off approach. Since the SLBD is greedy toward finding solutions with lower costs, the quality of makespan has

Table 6 Regression results with objective value as dependent variable

	CP	HBD	SLBD	MILP
Intercept	−40.864	−31.500	−31.448	−43.733
|J|	1.024*	1.024*	1.012*	1.110*
|K|	3.575*	2.900*	2.933*	3.331*
|S|	2.324	1.086	1.268	2.551
RT	−8.087	−9.921*	−10.150*	−7.955
R	14.864*	12.758*	12.857*	14.480*
M	0.512	−0.258	−0.358	1.332
DF	−0.381	−0.365	−0.333	0.0635
Adjusted R-square	0.778	0.800	0.801	0.701

*This estimate is significantly different from zero at 95% confidence level

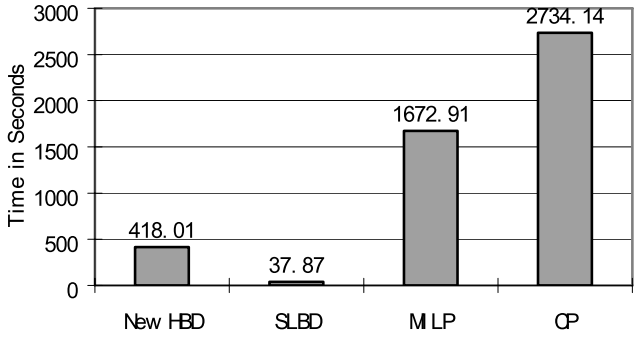

Fig. 10 Comparison of the average computational time

to be sacrificed to some degree. The pure MILP approach performs the worst among the four in project makespan. Different from the decomposition approach, the MILP attempts to solve the original problem with a mixture of assignment and scheduling variables and constraints, which does not exploit the structure of the problem at all. Thus, it often fails to find a shorter makespan for a given assignment when there exists such a makespan.

Next we compare the average computational time for each approach to find their best solutions in Fig. 10.

The two decomposition algorithms are significantly faster than the pure MILP and CP algorithms alone. The SLBD is the fastest among the four due to its heuristic nature, spending considerably less running time than the other three, i.e., around 1/11 of HBD, 1/45 of MILP, and 1/72 of CP. HBD is the second best with about 1/4 of the running time of MILP and 1/7 of CP. The efficiency of CP for solving scheduling problems has been greatly hampered by the assignment part of PSMPR. With decomposition, the advantage of CP has been fully exploited, solving a feasibility scheduling subproblem within seconds and saving the overall computational time significantly.

5.2.3 Multi-factor analysis

Multiple linear regression (MLR) is used to quantify the effects of the factors on the algorithm performance. We find that both the objective value and makespan can be well explained by MLR. The computational time, however, cannot be well explained with an R-square less than 30%. This is probably due to the problem's combinatorial nature and *NP-hardness* such that even a small-size instance may take a long time to solve. Table 6 shows the regression results with objective value as dependent variable.

It is clear that the decomposition algorithms HBD and SLBD have higher adjusted R-square than the direct approaches, which is probably because they are structured to handle different types of subproblems separately instead of solving the original problem directly. Also notice that the network complexity represented by RT is significantly nonzero only for the two decomposition approaches.

The regression results for the makespan as dependent variable are presented in Table 7. Instead of using RT directly, we have used $|J| * RT$ to capture the interactive effect between |J| and RT, which improves the adjusted R-square from around 70% to over 90% for the decomposition approaches. Again, HBD and SLBD have higher adjusted R-square.

The regression equations obtained above can be used to construct the three-dimensional surface graphs for comparing algorithm performance. This analysis provides us with a visualization of which algorithm dominates the other in what regions of the problem space. We set the difference of the performance measures (objective value or makespan) between two methods as a dependent variable and vary the number of tasks |J| and the number of skills |K|, which are two major factors affecting the problem size and are both significant in the associated regression functions.

Figure 11 through Fig. 13 compares HBD with MILP, CP and SLBD in objective function value, respectively. Figure 11 shows that the HBD tends to dominate MILP in objective value when |J| or |K| increases. From Fig. 12 we observe that although |J| does not seem to have much impact on the difference of objective value between HBD and CP, as |K| increases the advantage of HBD over CP increases. Figure 13 indicates that when |J| increases, the SLBD tends

Table 7 Regression results with *makespan* as dependent variable

	CP	HBD	SLBD	MILP
Intercept	−72.871	−47.850	−84.194	−147.090
|J|	1.609*	1.482*	2.450*	3.077*
|K|	6.955*	4.982*	3.400*	6.468*
|S|	−5.820	−0.366	−1.442	11.00290
R	9.257	0.960	8.723*	17.153*
M	−2.664	0.228	−0.00460	−0.966
DF	9.683	1.667	22.492*	29.381*
J ∗ RT	7.869*	7.571*	7.182*	6.765*
Adjusted R-square	0.809	0.919	0.904	0.725

*This estimate is significantly different from zero at 95% confidence level

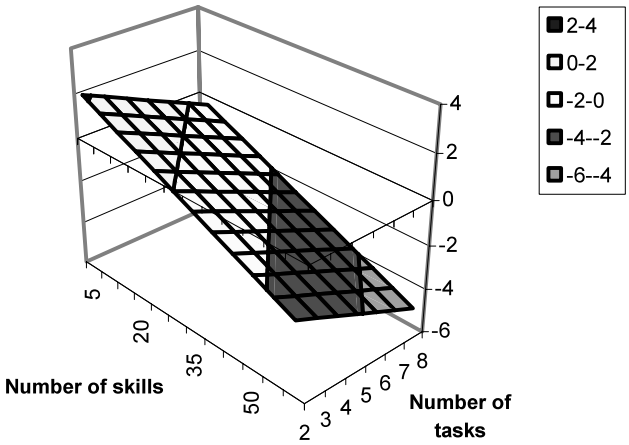

Fig. 11 HBD dominates MILP below zero for objective value

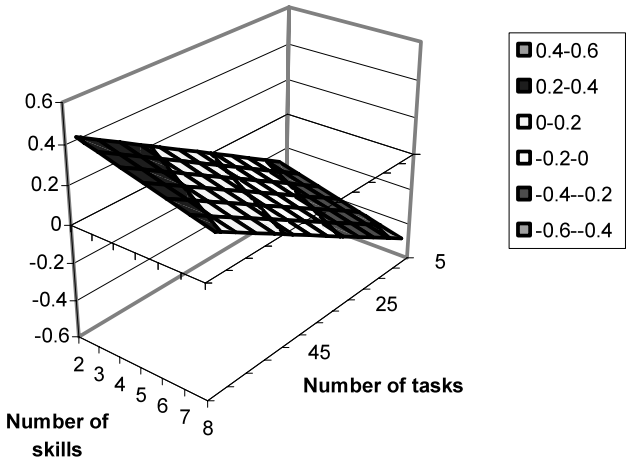

Fig. 13 HBD dominates SLBD below zero for objective value

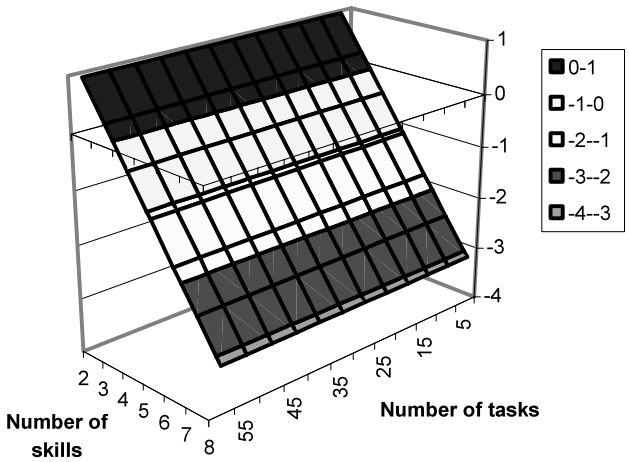

Fig. 12 HBD dominates CP below zero for objective value

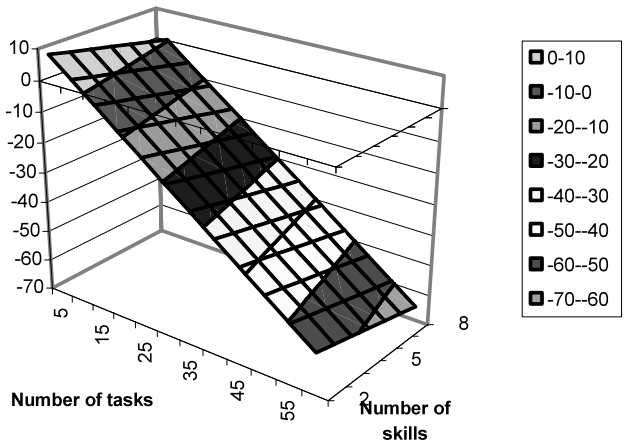

Fig. 14 HBD dominates MILP below zero for *makespan*

to dominate HBD, which is not surprising given the heuristic nature of SLBD and the computational time limits imposed on the HBD. However, as |K| increases the advantage of HBD over SLBD increases.

Figure 14 through Fig. 16 compares HBD with MILP, CP and SLBD in project makespan, respectively. We observe from Figs. 14 and 15 that the HBD tends to dominate both MILP and CP in project makespan when |J| or |K| increases. Figure 16 shows that the only problem space in which the

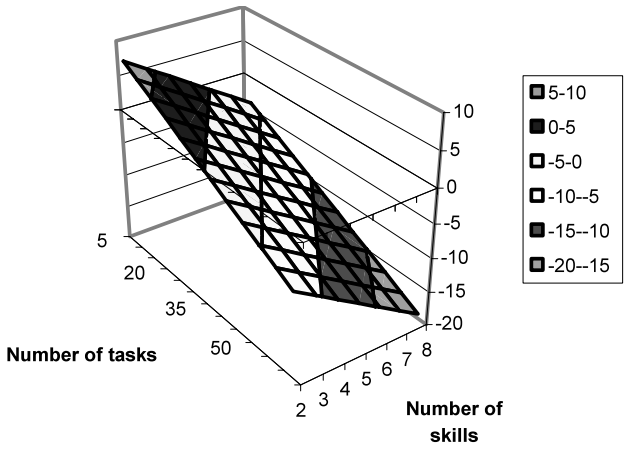

Fig. 15 HBD dominates CP below zero for *makespan*

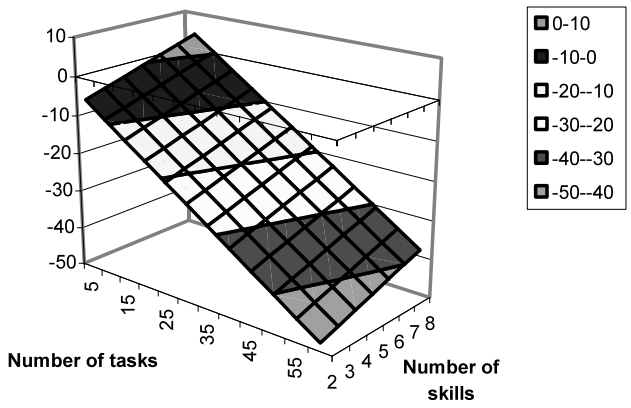

Fig. 16 HBD dominates SLBD below zero for *makespan*

SLBD dominates HBD is where |J| is small and |K| is large (e.g., $|J| = 5$ and $|K| = 8$), which does not seem likely to occur in real world problems.

6 Discussions

Although we present our model and hybrid decomposition algorithm in the context of a pure scheduling problem, our modeling and solution approach can be modified to cope with various planning problems. A variant of our model has been applied to the US Navy's DDX crew optimization problem (Li and Womer 2006a), which finds the minimum number of sailors with optimal skill mix to man a ship, while completing a specific mission consisting of interrelated tasks. There, the resource assignment subproblem is modeled as a bin packing problem with conflicts (BPC, Jasen 1999); while the resulting scheduling subproblem leads to a single-mode RCPSP with unary resources. This application aids the decision maker to plan for necessary training requirements in the intermediate run. A closely related model, the multi-mode RCPSP as discussed in Sect. 2.3, has been applied to tackle an important planning problem in the supply chain optimization field, namely, the supply chain configuration problem with explicit resource and capacity constraints (Li and Womer 2008). There, the assignment aspect involves choosing the optimal configurations/modes for each activity/process in the supply chain (discrete decision domain); while the scheduling aspect involves determining the inbound and outbound service times of each process subject to temporal constraints (continuous decision domain).

We now examine the relationship between our proposed model/algorithm and the temporal planning problems in the planning research community, which involve arranging actions and assigning resources in order to accomplish given tasks and objectives over a period of time (Fox and Long 2003; Wah and Chen 2006). Note that both scheduling and assignment decisions are critical in the temporal planning problem: (i) on one hand, time-related technical requirements are often modeled by a network of temporal constraints (Dechter et al. 1991); (ii) on the other hand, the execution of activities requires renewable resources—machines, vehicles, or skilled personnel (as considered in this paper), which often gives rise to resource allocation/assignment type of subproblems resembling multiprocessor scheduling or bin packing problems (Fox and Long 2001). Both (i) and (ii) fit well into our modeling framework. To be specific, the work-break-down structure (WBS) in project scheduling makes it flexible in defining activities with different levels of details. Moreover, the generalized temporal constraints considered in our model capture a rich set of time dependencies, such as overlaps and delays in addition to precedence relations (Neumann et al. 2002), which are ubiquitous in the planning context (Dechter et al. 1991). As for the resource allocation decisions, the inclusion of workloads of renewable resource units makes it possible to allocate each resource unit through the planning horizon. It is also possible to allow the resource availability vary over time.

Our hybrid decomposition algorithm also shares some similar insights with certain solution approaches developed in the temporal planning field. Notably, Fox and Long (2001) present domain analysis techniques to identify and extract subproblems, for which effective and efficient solution methods are available. They also stress the challenge of integrating different solution procedures to cooperate in solving the original problem. In our HBD algorithm, the original problem is decomposed into an assignment subproblem and a scheduling subproblem, which is solved by integer programming methods and constraint programming, respectively. The two solvers are linked together via "cuts" generated by temporal analysis. Such an OR/AI hybrid framework was recently advocated by Fox (2006), and was believed to be an promising line of research in the planning

community. Another study of utilizing such a hybrid strategy, that integrates max-SAT technique in AI and program evaluation and review technique (PERT) in OR, to cope with temporal planning is provided by Xing et al. (2006). The solution approach by Wah and Chen (2006) relies on constraint partitioning, which decomposes the original problem into significantly smaller subproblems. Instead of generating "cuts" to resolve violations of global constraints, they devise a penalty formulation and adjust the weights of violated constraints to resolve conflicts. The HBD paradigm also shares some similarities with the so called Squeaky Wheel Optimization (SWO) proposed by Joslin and Clements (1999). The core of SWO is a Construct/Analyze/Prioritize cycle. The Construct component operates in the solution space, which is analogous to solving the relaxed master problem in HBD. The Prioritize component operates in the prioritization space to iteratively generate new priorities for Construct. The Analyze component serves as the link between Construct and Prioritize, which finds "troubled" elements in the current solution. The role of Analyze is much similar to that of the cut generating scheme in our HBD, which deduces "violated" assignment constraints and adds them iteratively to the master problem. Frank and Kurklu (2005) successfully applied SWO to an optimization problem that lies on the scheduling boundaries of classical planning on SOFIA at NASA.

7 Conclusions and future research

In this paper, we studied the project scheduling problem with multi-skilled personnel to minimize the total staffing costs for executing a project. Our model is general enough to include minimum and maximum time lags as generalizations to precedence constraints in traditional machine scheduling settings. The model also takes each individual's workload capacity into consideration, a feature often needed in realistic personnel scheduling. We also show the relevance of our model to the planning research community, in particular, the temporal planning problems.

We developed a hybrid MILP/CP Benders decomposition (HBD) algorithm to solve this strongly *NP-hard* problem. The key to the practical effectiveness and efficiency of our algorithm is the use of temporal analysis to generate problem-specific cuts (instead of the general no-good cuts) during the Benders decomposition iterations. The computational results show that the HBD excels other approaches in solution quality with reasonable computational time. Notably, the HBD finds and proves more optimal solutions than the pure MILP method while spending significantly less computation time. When the project does not have a restrictive deadline, it performs particularly well (with a success rate of 100% to prove optimality for the set of tested problem instances). The solution quality of HBD is also robust as evident by our multi-factor analysis.

Two lines of research could be conducted in the future. From a modeling perspective, it might be interesting to consider personnel's skill proficiency. This can be achieved by either generalizing the objective function to minimize the total assignment costs or let task/skill's processing time depend on assigned personnel's proficiency level. From an algorithmic perspective, alternative hybrid strategies could be applied within the HBD framework proposed in this paper. This is motivated by the fact that it is still costly for MILP methods to solve the relaxed master problem which is *NP-hard* itself. Inspired by the *branch–and–check* approach of Thorsteinsson (2001) where the master problem does not need to be solved to optimality, we could apply a local search heuristic to obtain a good quality solution to the master problem, which leads to a hybrid local search and constraint programming algorithm. This development is particularly attractive when dealing with large size problems in the real world. It will also be interesting to further explore the possibility for the HBD paradigm to cope with the temporal planning problems in the planning research community, given the analogies between the optimization problem studied in this paper and temporal planning problem.

Acknowledgements This research was supported by the Office of Naval Research (ONR) under Grant No. N00140310621. We would also like to thank Derek Long and two anonymous referees for their detailed and constructive comments that help improve this paper.

References

Aksin, O. Z., Karaesmen, F., & Ormeci, E. L. (2006). A review of workforce cross-training in call centers from an operations management perspective. In D. Nembhard (Ed.), *Workforce cross training handbook*. Boca Raton: CRC Press.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based scheduling: applying constraint programming to scheduling problems*. New York: Springer.

Bartusch, M., Mohring, R. H., & Randermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, *16*, 201–240.

Bellenguez, O., & Neron, E. (2004). Methods for solving the multi-skill project scheduling problem. In *Proceedings of the ninth international workshop on project management and scheduling*, Nancy, France.

Benders, J. F. (1962). Partition procedures for solving mixed variables programming problems. *Numerische Mathematik*, *4*, 238–252.

Benoist, T., Laburthe, F., & Rotternbourg, B. (2001). Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems. In *Integration of AI and OR techniques in constraint*.

Benoist, T., Gaudin, E., & Rotternbourg, B. (2002). Constraint programming contribution to Benders decomposition: A case study. In *The eighth international conference on principles and practice of constraint programming*, Ithaca, New York.

Bockmayr, A., & Kasper, T. (1998). A unifying framework for integer and finite domain constraint programming. *INFORMS Journal on Computing*, *10*(3), 287–300.

Brucker, P. (2001). *Scheduling algorithms*. Berlin: Springer.

Brucker, P. (2002). Scheduling and constraint propagation. *Discrete Applied Mathematicsu*, *123*(1–3), 227–256.

Brucker, P., & Knust, S. (1998). Solving large-sized resource-constrained project scheduling problems. In J. Weglarz (Ed.), *Project scheduling: recent models, algorithms and applications*. Dordrecht: Kluwer Academic.

Brucker, P., & Knust, S. (2000). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, *127*(2), 335–362.

Brucker, P., & Knust, S. (2003). Lower bounds for resource-constrained project scheduling problems. *European Journal of Operational Research*, *149*, 302–313.

Brucker, P., Drexl, A., Mohring, R. H., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models and methods. *European Journal of Operational Research*, *112*(1), 3–41.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*(1–3), 61–95.

Dincbas, M., Van Hentenryck, P., Simonis, H., & Aggoun, A. (1988). The constraint logic programming language CHIP. In *Proceedings of the 2nd international conference on fifth generation computer systems*.

Drexl, A., & Kimms, A. (2001). Optimization guided lower and upper bounds for the resource investment problem. *Journal of Operational Research Society*, *52*, 340–351.

Drexl, A., Juretzka, J., Salewski, F., & Schirmer, A. (1998). New modeling concepts and their impact on resource-constrained project scheduling. In J. Weglarz (Ed.), *Project scheduling: recent models, algorithms and applications*. Dordrecht: Kluwer Academic.

Easton, K., Nemhauser, G., & Trick, M. (2004). CP based branch–and–price. In M. Milano (Ed.), *Constraint and integer programming*. Berlin: Springer.

Eremin, A., & Wallace, M. (2001). Hybrid Benders decomposition algorithms in constraint logic programming. *Lecture Notes in Computer Science*, *2239*, 1–15.

Focacci, F., Laborie, P., & Nuijten, W. (2000). Solving scheduling problems with setup times and alternative resources. In *Proceedings of the fifth international conference on artificial intelligence planning and scheduling*.

Fox, M. (2006). Planning for mixed discrete continuous domains. In J. C. Beck & B. M. Smith (Eds.), *Lecture notes in computer science*: *Vol. 3990*. *CPAIOR 2006* (p. 2). Berlin: Springer.

Fox, M., & Long, D. (2001). Hybrid STAN: Identifying and managing combinatorial optimization sub-problems in planning. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)*.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expression temporal planning domains. *Journal of Artificial Intelligence Research*, *20*, 61–124.

Frank, J., & Kurklu, E. (2005). Mixed discrete and continuous algorithms for scheduling airborne astronomy observations. In R. Batak & M. Milano (Eds.), *Lecture notes in computer science*: *Vol. 3524*. *CPAIOR 2005* (pp. 183–200). Berlin: Springer.

Harvey, W., & Ginsberg, M. (1995). Limited discrepancy search. In *The fourteenth international joint conference on artificial intelligence (IJCAI-95)*.

Hooker, J. (2000). *Logic-based methods for optimization: combining optimization and constraint satisfaction*. New York: Wiley-Interscience.

Hooker, J. (2002). Logic, optimization and constraint programming. *INFORMS Journal on Computing*, *14*(4), 285–321.

Hooker, J., & Osorio, M. (1999). Mixed logical-linear programming. *Discrete Applied Mathematics*, *96–97*, 395–442.

ILOG (2002a). ILOG Solver 5.3 User's Manual.

ILOG (2002b). ILOG CPLEX 8.1 User's Manual.

ILOG (2002c). OPL Studio 3.6.1 User's Manual: ILOG, Inc.

Jain, V., & Grossmann, I. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, *13*(4), 258–276.

Jasen, K. (1999). An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization*, *3*, 363–377.

Joslin, D. E., & Clements, D. P. (1999). "Squeaky Wheel" optimization. *Journal of Artificial Intelligence Research*, *10*, 353–373.

Lasdon, L. S. (1970). *Optimization theory for large systems*. New York: Macmillan Co.

Li, H., & Womer, K. (2006a). *Determining crew composition for a new technology* (Working Paper).

Li, H., & Womer, K. (2006b). Project scheduling with multi-purpose resources: a combined MILP/CP decomposition approach. *International Journal of Operations and Quantitative Management*, *12*(4), 305–325.

Li, H., & Womer, K. (2008). Modeling the supply chain configuration problem under resource constraints. *International Journal of Project Management*, *26*(6), 646–654.

Milano, M., & Trick, M. (2004). Constraint and integer programming. In M. Milano (Ed.), *Constraint and integer programming: toward a unified methodology*. Berlin: Springer.

Nemhauser, G., & Wolsey, L. (1988). *Integer and combinatorial optimization*. New York: Wiley.

Neron, E., Bellenguez, O., & Heurtebise, M. (2006). Decomposition method for solving multi-skill project scheduling problem. In *Proceedings of the tenth international workshop on project management and scheduling*, Poznan.

Neumann, K., Schwindt, C., & Zimmermann, J. (2002). *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*. New York: Springer.

Schwindt, C. (1996). *Generation of resource-constrained project scheduling problems with minimal and maximal time lags* (Technical Report WIOR-489). Institute für Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Sellmann, M., & Fahle, T. (2003). Constraint programming based Lagrangian relaxation for the automatic recording problem. *Annals of Operations Research*, *118*, 17–33.

Thesen, A. (1977). Measures of the restrictiveness of project networks. *Networks*, *7*, 193–208.

Thorsteinsson, E. S. (2001). Branch–and–check: A hybrid framework integrating mixed integer programming and constraint logic programming. In *Proceedings of the seventh international conference on principles and practices of constraint programming (CP-01)*. Berlin: Springer.

Tsang, E. (1993). *Foundations of constraint satisfaction*. San Diego: Academic Press.

Van Hentenryck, P. (1999). *The OPL optimization programming language*. Cambridge: MIT Press.

Wah, B. W., & Chen, Y. (2006). Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, *170*, 187–231.

Wallace, M., Novello, S., & Schimpf, J. (1997). *ECLiPSe: A platform for constraint logic programming*. Available from: http://www.icparc.ic.ac.uk/eclipse/reports/eclipse/eclipse.html.

Xing, Z., Chen, Y., & Zhang, W. (2006). An efficient hybrid strategy for temporal planning. In J. C. Beck & B. M. Smith (Eds.), *Lecture notes in computer science*: *Vol. 3990*. *CPAIOR 2006* (pp. 273–287). Berlin: Springer.