A Decomposition Approach for Solving A Resource Constrained Project Scheduling Problem[1]

Haitao Li

School of Business Administration, University of Mississippi,
Holman 320, University, MS 38677
Telephone: (662) 513-9965
Email: hli@bus.olemiss.edu

Keith Womer

College of Business Administration, University of Missouri – St Louis
One University Blvd, St Louis, MO 63121-4400
Telephone: (314) 516-6109
Email: keithwomer@umsl.edu

ABSTRACT

We study a project scheduling problem with multi-purpose resources (PSMPR). A multi-purpose resource is a resource unit that is "versatile" enough to perform different types of operations. A mixed-integer linear programming (MILP) and a constraint programming (CP) formulation for the standard PSMPR model are each presented in this paper. We also propose a decomposition approach combining the MILP and CP techniques to take advantage of the complementary strengths of the two.

INTRDUCTION

This paper was motivated by a scheduling problem in the software development department in a small firm. The department operated as a job shop with projects arriving at irregular intervals with varying work requirements. The department consisted of three categories of staff: programmers, systems analysts, and quality control specialists. The programmers were generally only able to perform that function. Systems analysts could perform all three functions but only some of the quality control specialists could perform either of the other two functions. Projects arrived with varying requirements for work on several tasks that required all three specialties. At any point in time an array of projects was in process. When a new project arrives the department supervisor needs to schedule work on the new project to provide quick response while not disrupting the required completion date for existing projects. We model skilled labor and flexible machines as multi-purpose resources in a project scheduling context and we refer to this problem as the project scheduling problem with multi-purpose resources (PSMPR).

This paper is organized as follows. First we introduce the standard PSMPR model and present both the MILP and CP formulations for the PSMPR. Then a decomposition approach integrating the mixed integer linear programming (MILP) and constraint programming (CP) is developed for solving the standard PSMPR. Finally conclusions and future research directions are discussed.

PROJECT SCHEDULING WITH MULTI-PURPOSE RESOURCES

The PSMPR belongs to the field of assignment-type resource constrained project scheduling problem (Drexl, Salewski and Schirmer, 1998). It is a generalization of the single-mode RCPSP, closely related but much richer than the multi-mode resource constrained project scheduling problem (MRCPSP, Talbot, 1982). And it can also be viewed as an extension of the classical multi-purpose machine scheduling problem (MPM, Brucker, 2001), due to the presence of generalized temporal constraints.

Problem Description

Consider a project consisting of a task set J, a skill set K and a personnel set S. Each task $j \in J$ has a constant processing time p_j and no preemption is allowed. Generalized temporal constraints, including release dates, due dates, precedence and minimum delay constraints, etc., have to be satisfied due to technical requirements. There is a deadline $\bar{T}$ on the makespan of the project. Each task $j \in J$ requires a set of skills, denoted by $K^j \subseteq K$, to be present simultaneously for the task to progress. For each skill $k \in K^j$, task j requires r_{jk} units of k. Each skill $k \in K$ requires an individual $s \in S^k$ to conduct, where S^k is the set of persons who possess skill k. Personnel are assumed to be unary resource, i.e., an individual can only perform one skill at one time point. Furthermore, personnel are multi-purpose in that for each $s \in S$, there is a skill set K^s representing the set of skills that can be performed by s. Setup times are not considered by assuming they are included into the processing times. We find a schedule of starting times of all the tasks and assignments of personnel to tasks/skills while minimizing the number of persons needed to perform the project.

The PSMPR is a generalization of the single-mode RCPSP since each task has more than one way (mode) to be performed. It also generalizes the MPM into a more complex project scheduling environment where temporal constraints are not restricted to release dates or due dates. Comparing with the MRCPSP, the PSMPR exploits the versatility of the skilled labor, which leads to a combinatorial number of possible ways for a task to be performed.

MILP Formulation

In this section we present an MILP formulation of the standard PSMPR model. Let J be the set of tasks in the project, K be the set of relevant skills required for project and S the personnel set. The set of skills required by task j is denoted by K^j. The set of persons who possess skill k is S^k. 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 limit w_s for each individual s. The deadline on the makespan of the project is $\bar{T}$. Our objective is to minimize the number of persons selected to perform the project. Define decisions variables as follows:

$z_s = 1, \text{iff s is selected to perform the project}, \forall s \in S$

$x_{jks} = 1$, iff 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, $\forall j \in J$

$y_{jj'} = 1$, iff task j precedes j', $\forall j, j' \in J$

The MILP formulation of the standard PSMPR model can be written in Figure 1. Constraint (1) ensures an individual cannot be assigned to any task/skill unless he/she is selected. Constraint (2) enforces that no individual is assigned to more than one skill in the same task. Constraint (3) says each skill in a task requires a person who possesses that skill. Constraints (4) through (6) take care of the generalized temporal relations. When $\delta_{jj'}$ equals p_j, constraint (4) becomes the precedence constraint. Constraint (7) stipulates that the sequencing variables $y_{jj'}$ and $y_{j'j}$ cannot equal 1 simultaneously. Constraint (8) states the logic relations between sequencing variables and assignment variables, i.e., if two tasks are assigned with the same person then these two tasks cannot overlap (sequencing relation must be determined). Constraint (9) is the big-M formulation, a traditional way to handle disjunctive scheduling constraints. Constraint (10) says sequencing variables equal zero if two tasks/skills are assigned with different persons. Constraint (11) ensures the total work load of each person s cannot exceed the work load limit of w_s.

$$\min \quad \sum_{s \in S} z_s$$

$$
\begin{array}{lll}
s.t. \quad x_{jks} - z_s \leq 0 & \forall j \in J, k \in K^j, s \in S^k & (1) \\
\sum_{k \in K^j} x_{jks} \leq 1 & \forall j \in J, s \in S & (2) \\
\sum_{s \in S^k} x_{jks} = 1 & \forall j \in J, k \in K^j & (3) \\
t_{j'} - t_j \geq \delta_{jj'} & \forall < j, j' > \in J \times J & (4) \\
t_j + p_j \leq d_j & \forall j \in J & (5) \\
t_j + p_j \leq \bar{T} & \forall j \in J & (6) \\
y_{jj'} + y_{j'j} \leq 1 & \forall ordered < j, j' > & (7) \\
y_{jj'} + y_{j'j} \geq x_{jks} + x_{j'k's} - 1 & \forall ordered \ < jk, j'k' >, \forall s \in S & (8) \\
t_{j'} \geq t_j + p_j - M(1 - y_{jj'}) & \forall j \neq j & (9) \\
y_{jj'} + y_{j'j} + x_{jks} + x_{j'k's'} \leq 2 & \forall ordered \ < jk, j'k' >, \forall s, s' \in S, s \neq s' & (10) \\
\sum_{j \in J} \sum_{k \in K^j} p_j x_{jks} \leq w_s & \forall s \in S & (11)
\end{array}
$$

Figure 1: MILP formulation of the PSMPR

CP Formulation

Constraint programming (CP) is the study of computational systems based on constraints. Although it originates in the Artificial Intelligence (AI) area back in 1960's, there have been increasing interests in combining CP with the traditional OR techniques to solve difficult combinatorial problems in recent years (Lustig and Puget 2001). Problem reduction and search are the two main solving techniques in CP (Tsang 1993). Problem reduction is often referred as consistency maintenance in literature. It is achieved by various constraint propagation algorithms. An important observation on problem reduction or consistency maintenance is that, reducing a problem to a minimal problem where no more redundant values and compound labels can be removed from the domain of the problems, is NP-hard (Tsang 1993). In other words, only easy redundant values and compound labels are removed. This is why a search procedure is often needed and the efficiency of CP will be greatly affected by the procedure of a tree search. Thus the success of CP usually depends on efficient constraint propagations algorithms and appropriate search procedures.

A conceptual discussion on the potential advantage of CP for solving a project scheduling problem has been treated in Dula et al. (2004). From modeling perspective, the expressive nature of CP often makes the model compact and easy to read; from algorithm perspective, many efficient constraint propagation algorithms are available for solving scheduling problems (Baptiste, Le Pape and Nuijten, 2001). Here we present a CP formulation for the standard PSMPR model using OPL (Van Hentenryck, 1999), a programming language that supports both mathematical programming and constraint programming. We explain the main CP constructs before presenting the CP formulation.

Variable Definition. Binary choice variables and assignment variables are defined in the same way as in the MILP model. For the scheduling part, however, we do not need those sequencing variables any more. CP provides us with a compact and elegant way to define variables and express scheduling constraints. We define activity at skill level, i.e., treat each skill required in a task, instead the task itself, as an activity:

Activity activity [j in Tasks, k in Skills] (duration[j]);

It defines a two-dimensional array of *Activity* for each skill k required in each task j. Notice that the set of skills within task j have the same duration of task j. An activity in OPL has attributes such as *start*, *duration*, etc., that can be retrieved for expressing temporal constraints.

Temporal Constraints. They can be stated in a general way:

a1.start – a2.start >= δ;

It says activity a1 cannot start until δ time units after activity a2 starts. It is clear that when δ equals the processing time of a2, the above constraint becomes precedence constraint. An alternative way to represent precedence constraint is:

a2 *precedes* a1;

The due date constraint can be stated as:

a.start + a. duration <= d;

where d is a constant due date for activity a.

Resource Constraints. Since each sailor can only perform one skill at a time, he/she can be modeled as unary resource as follows:

UnaryResource skilledLabor[Personnel];

It declares an array of unary resource called "skilledLabor" over the whole set of available people called "Personnel".

Since a skill can be assigned to any individual who can perform that skill, the idea of alternative resource in OPL (Van Hentrenryck 1999) can be applied:

AlternativeResources alterRes (skilledLabor);

It declares a pool of alternative resources called "alterRes" over the set of "skilledLabor" declared earlier.

The resource requirement constraint is stated as:

a *requires* alterRes;

It says activity a requires one person from the resource pool called "alterRes".

Another important consideration is to prevent assigning sailors to skills they cannot perform. Suppose sailor s cannot perform activity a, such a constraint can be enforced by:

not activityHasSelectedResource (a, alterRes, s);

It prevents sailor s to be chosen from the alternative resource pool to perform activity a. activityHasSelectedResource (a, alterRes, s) is a Boolean function that returns true if sailor s is assigned to activity a. By adding a *not* in front of it, we prevent this event from happening any time during the solving process.

Linking Constraints. We also need to link the assignment variables to the CP constructs:

activityHasSelectedResource (a, alterRes, s) <=> assign [a, s] = 1;

Now the CP formulation for our standard PSMPR model can be stated in Figure 2.

$$\min \quad \sum_{s \in S} z_s$$

$$s.t. \quad x_{jks} - z_s \le 0 \qquad \forall j \in J, k \in K^j, s \in S^k \qquad (12)$$

$$\sum_{j \in J} \sum_{k \in K^j} p_j x_{jks} \le w_s \qquad \forall s \in S \qquad (13)$$

$$\text{a } \textit{precedes} \text{ makespan} \qquad \forall \text{a} \in \text{A} \qquad (14)$$

$$\text{a } \textit{precedes} \text{ b} \qquad \forall (\text{a}, \text{b}) \in \text{P} \qquad (15)$$

$$\text{a.start} + \delta_{\text{ab}} \le \text{b.start} \qquad \forall (\text{a}, \text{b}) \in \text{MinDelay} \qquad (16)$$

$$\text{a.start} + p_a \le d_a \qquad \forall \text{a} \in \text{A} \qquad (17)$$

$$\text{makespan.end} \le \bar{T} \qquad (18)$$

$$\text{a } \textit{requires} \text{ alterRes;} \qquad \forall \text{a} \in \text{A} \qquad (19)$$

$$\textit{not activityHasSelected}\,\text{Re}\,\textit{source}(\text{a, alterRes, s}) \qquad (20)$$

$$\textit{activityHasSelected}\,\text{Re}\,\textit{source}(\text{a, alterRes, s}) \Leftrightarrow \text{assign[a, s]} = 1 \qquad (21)$$

Figure 2: CP formulation of the PSMPR

The role of the binary variables in the CP model is only to count the number of selected personnel in order to express the objective function.

A COMBINED MILP/CP DECOMPOSITION

The single-mode RCPSP has been proved to be NP-hard (Blazewicz, Lenstra and Rinnooy Kan, 1983), which means currently no polynomial algorithms exist to solve it to optimality (Garey and Johnson, 1979). Since the PSMPR includes the single-mode RCPSP as a special case, exact methods such as the MILP and CP often fail to get quality solutions for it with reasonable computational time. In this section we develop a combined MILP/CP decomposition approach to solve this challenging combinatorial problem.

Decomposition Framework

We propose a service level based decomposition (SLBD) framework to facilitate the above cooperation scheme. Two layers of resources are present in the standard PSMPR model. The first layer is the *service level*, which is the set of skills (service) provided by the skilled labor. The second layer is the actual personnel or skilled labor. Each task interacts only with the service level directly, i.e., we know which and how many skills a task requires, but we do not know which person among the second layer resource performs each skill. Figure 3 illustrates the SLBD framework.

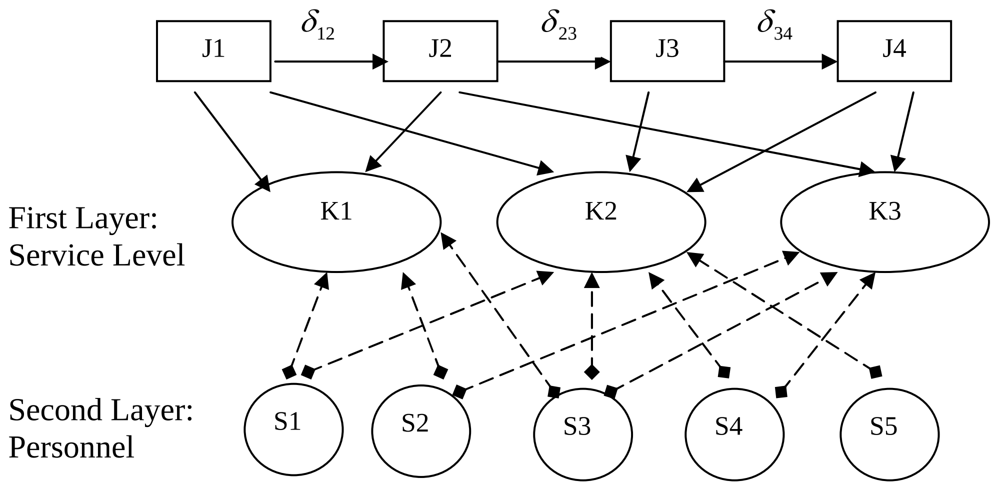

Figure 3: The SLBD framework

The rectangles represent tasks with δ_{ij} being the minimum delay between task i and j. The requirement of first layer resources (service level) is given by the task-skill relations. For instance, task J2 requires 1 unit of skill K1 and K3. The dotted arrow shows the skill-mix of the personnel, with the actual assignments unknown. For example, S3 is the most versatile individual as he/she is able to perform all the three skills, but which task(s)/skill(s) is S3 actually assigned to is what we need to find out. The PSMPR can be decomposed into two sub-models: a single-mode RCPSP with "cumulative" first-layer-resource (service level) and a generalized assignment problem (GAP).

The RCPSP CP Model. The single-mode RCPSP CP sub-model is shown in Figure 4.

$$\begin{array}{lll}
\min & \text{makespan.end} & \\
s.t. & \text{a } \textit{precedes} \text{ makespan} \quad \forall a \in A & (22) \\
& \text{a } \textit{precedes} \text{ b} \quad \forall (a,b) \in P & (23) \\
& \text{a.start} + \delta_{ab} \le \text{b.start} \quad \forall (a,b) \in \mathit{MinDelay} & (24) \\
& \text{a.start} + p_a \le d_a \quad \forall a \in A & (25) \\
& \text{a } \textit{requires} \text{ (1) resource k} \quad \forall a \in A, k \in K^a & (26)
\end{array}$$

Figure 4: The scheduling sub-model

There are two crucial points that have to be stressed. First, unlike the case in the pure CP model presented earlier, activities are defined at task level instead of skill level. Second, we treat the first-layer "service level" as the "cumulative" resource in the scheduling sub model, whereas in the pure CP formulation the second-layer personnel are directly treated as "unary" resources. In this way, there is only one mode to perform each activity, which gives us a single-mode RCPSP, much easier to solve than the original PSMPR.

The GAP ILP Model. Figure 5 shows the GAP integer linear programming (ILP) sub-model.

$$\begin{array}{lll}
\min & \sum_{s \in S} z_s & \\
s.t. & x_{jks} - z_s \le 0 \quad \forall j \in J, k \in K^j, s \in S^k & (27) \\
& \sum_{s \in S^k} x_{jks} = 1 \quad \forall j \in J, k \in K^j & (28) \\
& \sum_{k \in K^j} x_{jks} \le 1 \quad \forall j \in J, s \in S & (29) \\
& \sum_{j \in J} \sum_{k \in K^j} p_j x_{jks} \le w_s \quad \forall s \in S & (30) \\
& x_{jks} + x_{j'k's} \le 1 \quad \forall (j, j') \in OL, k \in K^j, k' \in K^{j'}, s \in S & (31)
\end{array}$$

Figure 5: The assignment sub-model

OL is the set of overlapping tasks, obtained by solving the RCPSP sub-model. Constraint (31) prevents all pairs of overlapping tasks/skills from being assigned with the same person. It establishes the link between the RCPSP and GAP sub-models.

CONCLUSIONS AND FUTURE RESEARCH

In this paper, we have studied an assignment-type resource constrained project scheduling problem called PSMPR to model the "versatility" of skilled labor and multi-purpose machines in a project scheduling context. In order to solve the standard PSMPR model efficiently, we developed a service level based decomposition (SLBD) approach combining the complementary strengths of both MILP and CP techniques.

Two directions of research could be possible in the future. From the modeling perspective, extensions of the standard PSMPR model, e.g., objective being minimizing the assignment cost, proficiency related processing time, etc. could be studied. From the algorithmic perspective, more sophisticated hybrid decomposition approaches (Jain and Grossmann, 2001) could be developed for solving the PSMPR.

NOTES

1. Supported by Office of Naval Research (ONR) grant # N00140310621.

REFERENCES

Baptiste, P., C. Le Pape, and W. Nuijten. (2001). *Constraint-Based Scheduling: applying constraint programming to scheduling problems*, Kluwer Academic Publishers.

Blazewicz, J., J. K. Lenstra, and A.H.G. Rinnooy Kan. (1983). "Scheduling Subject to Resource Constraints: Classification and Complexity." *Discrete Applied Mathematics* 5, 11-24.

Brucker, P. (2001). *Scheduling Algorithms*, Springer-Verlag, Berlin.

Drexl, A., J. Juretzka, F. Salewski, and A. Schirmer. (1998). "New Modeling Concepts and Their Impact on Resource-Constrained Project Scheduling." In Weglarz, J., editor, *Project Scheduling: Recent models, algorithms and applications*, Kluwer Academic Publishers.

Dula, J., K. Lewis, K. Womer and H. Li. (2004). "A Constraint Programming Approach to Solve Resource Constrained Project Scheduling Problem." In *Proceedings of International Academy of Business and Public Administration Disciplines Conference (IABPAD-04)*, New Orleans, Louisiana.

Garey, M.R., and D.S. Johnson. (1979). *Computers and Intractability – A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco.

ILOG, Inc. (2002). *OPL Studio 3.6.2 User's Manual*, ILOG, Inc, France.

Jain, V. and I.E. Grossmann. (2001). "Algorithms for Hybrid MILP/CP Models for A Class of Optimization Problems." *INFOMRS Journal on Computing* 13, 258-276.

Kolish, R., A. Sprecher, and A. Drexel. (1995). "Characterization and Generation of A General Class of Resource-Constrained Project Scheduling Problems." *Management Science* 41, 1693-1703.

Schwindt, C. (1996). “Generation of Resource-Constrained Project Scheduling Problems with Minimal and Maximal Time Lags.” *Technical Report WIOR – 489*, Karlsruhe, Germany.

Talbot, F.B. (1982). “Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case.” *Management Science* 28, 1197-1210.

Van Hentenryck, P. (1999). *The OPL Optimization Programming Language*, MIT Press, Cambridge, MA.