A Hybrid Algorithm for a Class of Resource Constrained Scheduling Problems

Yingyi Chu and Quanshi Xia

IC-Parc, Imperial College London,
London SW7 2AZ, UK
{yyc, qx1}@imperial.ac.uk

Abstract. This paper presents a hybrid algorithm for a class of resource constrained scheduling problems based on decomposition. The general minimum completion time problem is considered, which has not been solved in a decomposed way by existing methods. The problem is first decomposed into an assignment master problem and a number of scheduling subproblems. The subproblem is formulated as both a constraint programming model and an integer programming model. The hybrid algorithm then combines constraint programming, integer programming and linear programming solvers in its three steps: the master problem solving, the subproblems solving and the cut generation. In particular, the cut generation method is based on the integer programming model, and in practice it is done by solving a linear program. Computational experiments have been carried out for the considered minimum completion time problems. The results show that the proposed algorithm could substantially reduce the solving time, compared with directly solving by mixed integer solvers.

1 Introduction

This paper studies an important class of resource constrained scheduling problems, where a set of jobs are assigned to and processed by a set of facilities, subject to resource constraints and release/due date constraints. Depending on the objective function, there are different versions of the problem, e.g. minimum cost problem, minimum completion time problem, etc.

These problems have attracted substantial research interests due to its importance in many application domains. Solution methods based on Benders decomposition have been proposed recently [12, 8, 13, 9] for some of the problems. A decomposition method partitions the problem into an assignment *master* problem and a number of independent scheduling *subproblems*, each for one facility. The master problem and the subproblems are solved iteratively and in particular the scheduling subproblems are often solved by constraint solvers as strong reasoning techniques are available. The key step is to generate valid cuts from the subproblems, guiding the search of the assignment solution in the master problem.

R. Barták and M. Milano (Eds.): CPAIOR 2005, LNCS 3524, pp. 110–124, 2005.

For minimum cost problems, Jain and Grossmann (2001), Harjunkoski and Grossmann (2002) and Hooker (2004) employ the no good cut that excludes sets of incompatible jobs assigned to a facility. For minimum makespan problems, a valid cut is proposed by Hooker (2004), but under the assumption of same release dates for all jobs (and same due dates in the computational study). As is pointed out in [9, 10], all these cuts are based on the *logical explanation* of the individual solution processes for the specific problems, instead of the dual information from the linear integer formulation of the subproblems.

This paper presents a general approach for tackling the resource constrained scheduling problems. A hybrid method is proposed where the necessary cuts are generated based on the *dual information* from the subproblem's formulation. The subproblem is formulated as equivalent integer programming (IP) and constraint programming (CP) models. The IP formulation is used to generate the integer Benders cuts by exploiting the dual information, while the CP formulation is used to efficiently solve the scheduling subproblems using strong constraint propagations for cumulative scheduling. To be concrete, this hybrid approach is instantiated to a solution algorithm for the *general minimum completion time problem*, which is not solvable by previous decomposition methods, as different release/due dates are allowed.

The paper is organized as follows. Section 2 introduces the considered problems. Section 3 presents the hybrid method. Section 4 details the key step in the proposed method, i.e., the cut generation. Section 5 presents the computational experiments and results. Section 6 concludes the paper.

2 The Resource Constrained Scheduling Problems

This section introduces the considered scheduling problems. A formulation of the general minimum completion time problem is given, which is used for the subsequent algorithm development and the computational experiments.

Consider a set of jobs, denoted by $\mathcal{J}$, and a set of facilities (machines), denoted by $\mathcal{M}$. Each job $j \in \mathcal{J}$ has a release date r_j and a due date d_j. Each facility $m \in \mathcal{M}$ provides a fixed amount of resources specified by C_m. The processing time of job j on facility m is given by p_{jm}, and the job j consumes the amount C_{jm} of resources during its processing time on m.

Following [9], we employ a discrete time formulation (where times are discretized to integers), instead of the continuous time model used in [12, 8, 15], because the discrete time formulation is often easier to solve, especially when the cumulative constraint (instead of the simpler disjunctive constraint) is considered. Let $\mathcal{T}$ denote the whole set of discrete time points in the considered problem, $\{\min_j\{r_j\}, \cdots, \max_j\{d_j\}\}$.

Define binary variables x_{jmt} for any job j, any facility m and any time point t in $\mathcal{T}_{jm}$, where $\mathcal{T}_{jm} \equiv \{r_j, \cdots, d_j - p_{jm}\}$ represents the possible starting times of job j on facility m. The variables are used to indicate when and on which facility a job starts, i.e. $x_{jmt} = 1$ if and only if job j starts from the discrete time point t at facility m. Variable H denotes the overall completion time of all

jobs. Using these variables, the following constraint states that each job must be processed by exactly one facility:

$$\forall j \in \mathcal{J}: \quad \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}_{jm}} x_{jmt} = 1 \tag{1}$$

and the resource constraints have to be observed on every facility at any time:

$$\forall t \in \mathcal{T}, \forall m \in \mathcal{M}: \quad \sum_{j\in\mathcal{J}} \sum_{t'=t-p_{jm}+1}^{t} C_{jm} x_{jmt'} \leq C_m \tag{2}$$

By definition, the completion time variable H satisfies:

$$\forall j \in \mathcal{J}, \forall m \in \mathcal{M}: \quad \sum_{t\in\mathcal{T}_{jm}} (t + p_{jm}) x_{jmt} \leq H \tag{3}$$

The minimum completion time problem is formulated as:

$$\mathbf{P}: \min_{x_{jmt}, H} H$$
$$s.t. \begin{cases} \sum_{m\in\mathcal{M}} \sum_{t\in\mathcal{T}_{jm}} x_{jmt} = 1 & \forall j \in \mathcal{J} \\ \sum_{j\in\mathcal{J}} \sum_{t'=t-p_{jm}+1}^{t} C_{jm} x_{jmt'} \leq C_m & \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \\ \sum_{t\in\mathcal{T}_{jm}} (t + p_{jm}) x_{jmt} \leq H & \forall j \in \mathcal{J}, \forall m \in \mathcal{M} \\ x_{jmt} \in \{0, 1\} & \forall j \in \mathcal{J}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}_{jm} \\ H \in \mathbb{Z}[\max_j\{r_j + \min_m\{p_{jm}\}\}, \max_j\{d_j\}] \end{cases}$$

3 A Hybrid Algorithm Framework

3.1 Decomposition

To decompose the problem, we first reformulate the problem $\mathbf{P}$ by disaggregating the variables. Introducing the assignment variables y_{jm}, to indicate whether job j is assigned to facility m or not, we can rewrite the problem as:

$$\mathbf{P}': \min_{y_{jm}, x_{jmt}, H} H$$
$$s.t. \begin{cases} \sum_{m\in\mathcal{M}} y_{jm} = 1 & \forall j \in \mathcal{J} \\ \sum_{t\in\mathcal{T}_{jm}} x_{jmt} = y_{jm} & \forall j \in \mathcal{J}, \forall m \in \mathcal{M} \\ \sum_{j\in\mathcal{J}} \sum_{t'=t-p_{jm}+1}^{t} C_{jm} x_{jmt'} \leq C_m & \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \\ \sum_{t\in\mathcal{T}_{jm}} (t + p_{jm}) x_{jmt} \leq H & \forall j \in \mathcal{J}, \forall m \in \mathcal{M} \\ y_{jm} \in \{0, 1\} & \forall j \in \mathcal{J}, \forall m \in \mathcal{M} \\ x_{jmt} \in \{0, 1\} & \forall j \in \mathcal{J}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}_{jm} \\ H \in \mathbb{Z}[\max_j\{r_j + \min_m\{p_{jm}\}\}, \max_j\{d_j\}] \end{cases}$$

A decomposition is based on a partition of variables. For the problem $\mathbf{P}'$, the variables y_{jm} and H are solved in an assignment master problem. If these variables are tentatively fixed, the rest of the problem, pertaining the variables x_{jmt}, further decomposes into $|\mathcal{M}|$ smaller subproblems, one for each facility m.

The master problem is written as:

$$\mathbf{MP}: \min_{y_{jm},H} H$$
$$s.t. \begin{cases} \sum_{m\in\mathcal{M}} y_{jm} = 1 & \forall j \in \mathcal{J} \\ \sum_{j\in\mathcal{J}} C_{jm}p_{jm}y_{jm} \le C_m(H - \min_j\{r_j\}) & \forall m \in \mathcal{M} \\ \text{cuts generated from subproblems} \\ y_{jm} \in \{0,1\} \ \ \forall j \in \mathcal{J} \\ H \in \mathbb{Z}[\max_j\{r_j + \min_m\{p_{jm}\}\}, \max_j\{d_j\}] \end{cases}$$

The second constraint is a strengthening valid constraint, asserting that, for each facility, the total 'volume' of resources consumed by the assigned jobs cannot exceed the available 'volume' of resources up to the completion time H.

Based on the idea of Benders decomposition, the master problem is solved to obtain a tentative assignment $\bar{y}_{jm}$, and a tentative completion time $\bar{H}$. Fixing the tentative assignment, the subproblem is obtained, and it is immediately decomposed according to the facilities. The subproblems try to schedule the tentatively assigned jobs to minimize the completion time, subject to resource constraint and release/due date constraint. If at all facilities the jobs are indeed finished within $\bar{H}$, then the optimal solution is found. Otherwise, a cut is generated from each subproblem where a feasible schedule within $\bar{H}$ is impossible, and the master problem is resolved with the new cuts added. The algorithm iterates until the optimal solution is attained.

Given the tentative solution, the subproblem for each facility is a cumulative scheduling problem for the assigned jobs. In our method, the subproblems have to be formulated in two different ways, serving different functionalities in the hybrid scheme.

3.2 Subproblem Formulations

The most parsimonious formulation of the subproblems is the constraint programming formulation, where variables t_j are defined to denote the starting time of the job j and $\mathcal{J}_m$ denotes the set of assigned jobs $\{j | j \in \mathcal{J}, \bar{y}_{jm} = 1\}$.

$$\forall m : \mathbf{SP}^m_{\mathbf{CP}}(\bar{y}_{jm}, \bar{H}) : \bar{H} \ge \min_{t_j : j\in\mathcal{J}_m} \max_{j\in\mathcal{J}_m}\{t_j + p_{jm}\}$$
$$s.t. \begin{cases} \text{cumulative}([t_j : j \in \mathcal{J}_m], [p_{jm} : j \in \mathcal{J}_m], [C_{jm} : j \in \mathcal{J}_m], C_m) \\ t_j \in \mathbb{Z}[r_j, d_j - p_{jm}] \ \ \forall j \in \mathcal{J}_m \end{cases}$$

The subproblem minimizes the completion time on facility m subject to the cumulative constraint on the tentatively assigned jobs, and then the optimal value of it is compared with the tentative value $\bar{H}$.

In order to generate cuts based on Benders decomposition, the subproblems are also formulated as an integer programming model. In problem $\mathbf{P}'$, by fixing the master problem variables, we obtain the following subproblems:

$$\forall m : \mathbf{SP}^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H}) : \min_{x_{jmt}} 0$$

$$s.t. \begin{cases} \sum_{t\in\mathcal{T}_{jm}} x_{jmt} = \bar{y}_{jm} & \forall j \in \mathcal{J} \\ \sum_{j\in\mathcal{J}} \sum_{t'=t-p_{jm}+1}^{t} C_{jm} x_{jmt'} \le C_m & \forall t \in \mathcal{T} \\ \sum_{t\in\mathcal{T}_{jm}} (t + p_{jm}) x_{jmt} \le \bar{H} & \forall j \in \mathcal{J} \\ x_{jmt} \in \{0, 1\} & \forall j \in \mathcal{J}, \forall t \in \mathcal{T}_{jm} \end{cases}$$

In this case the subproblems are feasibility problems as the variables x_{jmt} do not contribute to the objective function in $\mathbf{P}'$.

Note that the subproblems are obtained based on formulation $\mathbf{P}'$, following the classic Benders decomposition procedure (ref. [2, 7]). The tentative master problem solution values only appear in the right hand sides. Yet the only difficulty is that the subproblems are now integer programs.

3.3 Hybrid Algorithm

The hybrid scheme is partitioned into three functional modules: the solution of the master problem, the solution of the subproblems, and the generation of cuts. This partition of functionality makes the *hybridization of solvers and formulations* possible. Different models and solvers are used for different modules: the master problem, given as an *IP model*, is solved by an *integer programming solver*; the subproblem is solved by a *constraint solver* using its *CP formulation*; the cut generation is based on the *IP formulation* of the subproblem and it is done by *solving a linear program*. The cut generation, which is the key step, will be detailed in Section 4, while this section presents the hybrid framework, and discusses the solution method of the master problem and the subproblems.

Algorithm 1 Hybrid Algorithm for Problem $\mathbf{P}'$

1. INITIALIZATION. Setup the initial master problem $\mathbf{MP^{(0)}}$ with no cut; set $k = 0$.

2. ITERATION.

(1) Master Problem Phase. Solve the integer linear program $\mathbf{MP^{(k)}}$ to obtain the tentative solution $\bar{y}^{(k)}_{jm}$ and $\bar{H}^{(k)}$; if $\mathbf{MP^{(k)}}$ is infeasible, then exit with the original problem infeasible.

(2) Subproblems Phase. For each facility m, solve the corresponding subproblem using the CP formulation $\mathbf{SP}^m_{\mathbf{CP}}(\bar{y}^{(k)}_{jm}, \bar{H}^{(k)})$; if all subproblems are feasible, then exit with the optimal solution found; otherwise continue to phase (3).

(3) Cut Generation Phase. For each subproblem that is infeasible, generate a Benders cut based on the IP formulation $\mathbf{SP}^m_{\mathbf{IP}}(\bar{y}^{(k)}_{jm}, \bar{H}^{(k)})$; add the new cuts to the master problem to construct $\mathbf{MP^{(k+1)}}$; set $k = k + 1$ and go back to phase (1).

The hybrid algorithm for the problem $\mathbf{P}'$ is summarized in the Algorithm 1. In step 2.(1), the master problem is solved by a standard mixed integer programming (MIP) solver, using the formulation **MP**. In step 2.(2), the subproblem

is solved by constraint programming. In particular, the edge-finding constraint propagation algorithm is applied to the cumulative constraint in $\mathbf{SP}^m_{\mathbf{CP}}(\bar{y}_{jm}, \bar{H})$. The edge-finding algorithm reduces the domain of the t_j variables by finding the jobs that have to precede or succeed a set of other jobs, based on the volume of resources they consume (ref. [1]). Furthermore, the branching search is enhanced by a probe backtracking technique, which uses a forward probing method (in addition to the conventional forward local consistency checking) to prune and guide the search (ref. [6]). The constraint solving algorithms used here are provided as libraries of the ECL^{i}PS^e [11] platform. The step 2.(3) is unspecified in the Algorithm 1, but it will be completed at the end of Section 4.

4 Cut Generation

4.1 Benders Cuts from Integer Subproblems

The Benders cut generation from integer subproblems is developed in a general setting, and it is then applied to the considered scheduling problem. This approach is based on the earlier idea reported in [4], but here a more general method is developed and a more efficient way of cut generation is presented.

Consider the following generic program $\mathbf{P}_g$ in the general decomposed form.

$$\mathbf{P}_g : \min_{\boldsymbol{y},\boldsymbol{x_m}} \boldsymbol{c}^T\boldsymbol{y} + \boldsymbol{d_1}^T\boldsymbol{x_1} + \boldsymbol{d_2}^T\boldsymbol{x_2} + \cdots + \boldsymbol{d_M}^T\boldsymbol{x_M}$$

$$s.t. \begin{cases} \boldsymbol{A_0 y} & & & & & \geq \boldsymbol{b_0} \\ \boldsymbol{A_1 y} + \boldsymbol{B_1 x_1} & & & & & \geq \boldsymbol{b_1} \\ \boldsymbol{A_2 y} + & \boldsymbol{B_2 x_2} & & & & \geq \boldsymbol{b_2} \\ \vdots \quad \vdots & & \ddots & & \vdots \quad \vdots & \\ \boldsymbol{A_M y} + & & & \boldsymbol{B_M x_M} & \geq \boldsymbol{b_M} \\ \boldsymbol{y} \in \mathcal{D}_y, \quad \boldsymbol{x_m} \in \{0,1\}^{n_m} & & & & \end{cases}$$

where the vector $\boldsymbol{y}$ represents the master problem variables, and the subproblem variables are divided to the vectors $\boldsymbol{x_m}$ $(m = 1, \cdots, M)$. In Benders decomposition, when $\boldsymbol{y}$ is fixed, the rest of the problem is decomposed into M subproblems. The master problem variables belong to a *finite* domain $\mathcal{D}_y$, while the subproblem variables are considered as binary.

The formulation $\mathbf{P}'$ for the scheduling problem fits into the above generic model. However we develop the cut generation method in a general setting using the problem $\mathbf{P}_g$.

According to Benders decomposition, the master problem and the subproblems are written as:

$$\mathbf{MP}_g : \min_{\boldsymbol{y},z_m} \boldsymbol{c}^T\boldsymbol{y} + z_1 + \cdots + z_M$$

$$s.t. \begin{cases} \boldsymbol{A_0 y} \geq \boldsymbol{b_0} \\ \text{Benders cuts} \end{cases}$$

$$\forall m : \mathbf{SP}_g^m(\bar{\boldsymbol{y}}) : \min_{\boldsymbol{x_m}} \boldsymbol{d_m^T x_m}$$
$$s.t. \begin{cases} \boldsymbol{B_m x_m} \geq \boldsymbol{b_m} - \boldsymbol{A_m \bar{y}} \\ \boldsymbol{x_m} \in \{0,1\}^{n_m} \end{cases}$$

where variables z_m represent the objective value of the subproblems. The master problem is solved to give a tentative solution $(\bar{\boldsymbol{y}}, \bar{z}_1, \cdots, \bar{z}_M)$. For each m, if the resulting $\mathbf{SP}_g^m$ is infeasible or the value $\bar{z}_m$ cannot be reached by $\mathbf{SP}_g^m$, then a Benders cut (over the master problem variables $\boldsymbol{y}$ and z_m) is generated. While there is a standard way of generating Benders cut when the subproblem is continuous, difficulties arise when the subproblem is an integer program. Next we focus on the Benders cut generation from the mth subproblem $\mathbf{SP}_g^m$.

It is essential that the Benders cut to be generated is *valid*, which means that it does not cut off any *feasible combination* of the values of $\boldsymbol{y}$ and z_m (i.e. the value of z_m can be reached by the subproblem parameterized by the value of $\boldsymbol{y}$). A valid Benders cut is often derived using the dual information from the subproblem. In order to extract dual information from the integer subproblem $\mathbf{SP}_g^m$, we define the fixed subproblems by fixing the integer variables $\boldsymbol{x_m}$ to a given value $\tilde{\boldsymbol{x}}_{\boldsymbol{m}}$:

$$\forall \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \in \{0,1\}^{n_m} : \mathbf{SP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}}) : \min_{\boldsymbol{x_m}} \boldsymbol{d_m^T x_m}$$
$$s.t. \begin{cases} \boldsymbol{B_m x_m} \geq \boldsymbol{b_m} - \boldsymbol{A_m \bar{y}} \\ \boldsymbol{x_m} = \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \\ \boldsymbol{x_m} \geq \boldsymbol{0} \end{cases}$$

For each subproblem there are totally 2^{n_m} number of fixed subproblems. As $\boldsymbol{x_m}$ is fixed to an integer value, the integrality constraint is dropped. The fixed subproblems are dualized to $\mathbf{DSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$ in order to elicit dual values.

$$\forall \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \in \{0,1\}^{n_m} : \mathbf{DSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}}) : \max_{\boldsymbol{u},\boldsymbol{v}} (\boldsymbol{b_m} - \boldsymbol{A_m \bar{y}})^T \boldsymbol{u} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \boldsymbol{v}$$
$$s.t. \begin{cases} \boldsymbol{B_m^T u} + \boldsymbol{v} \leq \boldsymbol{d_m} \\ \boldsymbol{u} \geq \boldsymbol{0}, \quad \boldsymbol{v} : \text{free} \end{cases}$$

If $\mathbf{SP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$ is infeasible (and thus $\mathbf{DSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$ is unbounded), then we use the homogeneous dual $\mathbf{HDSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$.

$$\forall \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \in \{0,1\}^{n_m} : \mathbf{HDSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}}) : \max_{\boldsymbol{u},\boldsymbol{v}} (\boldsymbol{b_m} - \boldsymbol{A_m \bar{y}})^T \boldsymbol{u} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \boldsymbol{v}$$
$$s.t. \begin{cases} \boldsymbol{B_m^T u} + \boldsymbol{v} \leq \boldsymbol{0} \\ \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1}, \quad \boldsymbol{-1} \leq \boldsymbol{v} \leq \boldsymbol{1} \end{cases}$$

In the programs, $\boldsymbol{u}, \boldsymbol{v}$ are dual variables. As $\mathbf{SP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$ is a *linear program*, strong duality property holds.

Much dual information can be extracted from the above dual programs. From an arbitrary feasible solution of any $\mathbf{DSP}_g^m{}_{\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$, we can derive an *optimality inequality* over the master problem variables:

$$(\boldsymbol{b_m} - \boldsymbol{A_m y})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}} \leq z_m \tag{4}$$

From an arbitrary feasible solution of any $\mathbf{HDSP}^{m}_{g\,\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$, we can derive a *feasibility inequality* over the master problem variables:

$$(\boldsymbol{b_m} - \boldsymbol{A_m y})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}} \leq 0 \tag{5}$$

However, not all these inequalities are valid. The following lemmas identify the valid ones among them[1].

Lemma 1. *An optimality inequality (4) is valid if the following sign condition is satisfied:*

$$\begin{cases} \tilde{\boldsymbol{v}}_i \leq 0 & \text{if } (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i = 1 \\ \tilde{\boldsymbol{v}}_i \geq 0 & \text{if } (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i = 0 \end{cases} \quad \forall i = 1, \cdots, n_m \tag{6}$$

Lemma 2. *A feasibility inequality (5) is valid if the sign condition (6) is satisfied.*

Using the above condition, one can find valid Benders cuts from the large family of inequalities specified by (4) and (5).

While any valid cut can be added to the master problem, it is desirable to find one that is as tight as possible with respect to the tentative solution $(\bar{\boldsymbol{y}}, \bar{z}_m)$. Formally, a *tightest* valid optimality cut with respect to $(\bar{\boldsymbol{y}}, \bar{z}_m)$ is defined as a valid cut

$$(\boldsymbol{b_m} - \boldsymbol{A_m y})^T \tilde{\boldsymbol{u}}^* + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^{*T} \tilde{\boldsymbol{v}}^* \leq z_m$$

such that

$$(\boldsymbol{b_m} - \boldsymbol{A_m}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}}^* + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^{*T} \tilde{\boldsymbol{v}}^* = \max_{\tilde{\boldsymbol{x}}_{\boldsymbol{m}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}} \{(\boldsymbol{b_m} - \boldsymbol{A_m}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}} : \quad s.t.(6)\}$$

The *tightest* valid feasibility cut is defined similarly. In other words, it is a matter of choice of $\tilde{\boldsymbol{x}}_{\boldsymbol{m}}$ and $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$, to maximize the left hand side value of the cut (with $\boldsymbol{y}$ instantiated to $\bar{\boldsymbol{y}}$), giving a tightest cut with respect to $(\bar{\boldsymbol{y}}, \bar{z}_m)$.

4.2 Cut Generation Programs

To elicit a valid optimality or feasibility cut, one needs to find out an assignment $\tilde{\boldsymbol{x}}_{\boldsymbol{m}}$ and a dual feasible value $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ such that the sign condition is satisfied. The sign condition (6) can be expressed as the following constraints:

$$\begin{cases} (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \leq 0 \\ (1 - \tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \geq 0 \end{cases} \forall i \in 1, \cdots, n_m \tag{7}$$

To find a tightest cut, one could maximize the left hand side value with $\boldsymbol{y}$ instantiated to $\bar{\boldsymbol{y}}$:

$$\max_{\tilde{\boldsymbol{x}}_{\boldsymbol{m}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}} (\boldsymbol{b_m} - \boldsymbol{A_m}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}} \tag{8}$$

[1] The proofs of all lemmas are given in the appendix.

Therefore, a tightest valid optimality cut can be generated using the dual constraint from $\mathbf{DSP}_{g\ \mathbf{F}}^{m}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$, the sign condition constraints (7) and the objective function (8):

$$\mathbf{CGP}_{g}^{m}(\bar{\boldsymbol{y}}): \max_{\tilde{\boldsymbol{x}}_{\boldsymbol{m}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}} (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}}$$

$$s.t. \begin{cases} (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \leq 0 \qquad \forall i \in 1, \cdots, n_m \\ (1 - \tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \geq 0 \ \forall i \in 1, \cdots, n_m \\ \boldsymbol{B}_{\boldsymbol{m}}^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{v}} \leq \boldsymbol{d}_{\boldsymbol{m}} \\ \tilde{\boldsymbol{u}} \geq 0, \tilde{\boldsymbol{v}} : \text{free} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \in \{0,1\}^{n_m} \end{cases}$$

A tightest valid feasibility cut can be generated using the dual constraint from $\mathbf{DSP}_{g\ \mathbf{F}}^{m}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$, the sign condition constraints (7) and the objective function (8):

$$\mathbf{HCGP}_{g}^{m}(\bar{\boldsymbol{y}}): \max_{\tilde{\boldsymbol{x}}_{\boldsymbol{m}}, \tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}} (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}_{\boldsymbol{m}}^T \tilde{\boldsymbol{v}}$$

$$s.t. \begin{cases} (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \leq 0 & \forall i \in 1, \cdots, n_m \\ (1 - \tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i \geq 0 & \forall i \in 1, \cdots, n_m \\ \boldsymbol{B}_{\boldsymbol{m}}^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{v}} \leq \boldsymbol{0} \\ \boldsymbol{0} \leq \tilde{\boldsymbol{u}} \leq \boldsymbol{1}, -\boldsymbol{1} \leq \tilde{\boldsymbol{v}} \leq \boldsymbol{1} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{m}} \in \{0,1\}^{n_m} \end{cases}$$

However, the above nonlinear mixed integer programs can be simplified to linear cut generation programs. Define $\tilde{\boldsymbol{v}}_i^+ \equiv (1 - \tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i$ and $\tilde{\boldsymbol{v}}_i^- \equiv (\tilde{\boldsymbol{x}}_{\boldsymbol{m}})_i \tilde{\boldsymbol{v}}_i$. Obviously, $\tilde{\boldsymbol{v}}_i = \tilde{\boldsymbol{v}}_i^+ + \tilde{\boldsymbol{v}}_i^-$. By this way the variables $\tilde{\boldsymbol{x}}_{\boldsymbol{m}}$ and $\tilde{\boldsymbol{v}}$ can be eliminated. A tightest valid optimality cut is generated by the following cut generation program:

$$\mathbf{CGP}'^{m}_{g}(\bar{\boldsymbol{y}}): \max_{\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}^+, \tilde{\boldsymbol{v}}^-} (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \boldsymbol{1}^T \tilde{\boldsymbol{v}}^-$$

$$s.t. \begin{cases} \boldsymbol{B}_{\boldsymbol{m}}^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{v}}^+ + \tilde{\boldsymbol{v}}^- \leq \boldsymbol{d}_{\boldsymbol{m}} \\ \tilde{\boldsymbol{v}}^+ \geq \boldsymbol{0}, \tilde{\boldsymbol{v}}^- \leq \boldsymbol{0} \\ \tilde{\boldsymbol{u}} \geq \boldsymbol{0} \end{cases}$$

A tightest valid feasibility cut is generated by the following cut generation program:

$$\mathbf{HCGP}'^{m}_{g}(\bar{\boldsymbol{y}}): \max_{\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}^+, \tilde{\boldsymbol{v}}^-} (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\bar{\boldsymbol{y}})^T \tilde{\boldsymbol{u}} + \boldsymbol{1}^T \tilde{\boldsymbol{v}}^-$$

$$s.t. \begin{cases} \boldsymbol{B}_{\boldsymbol{m}}^T \tilde{\boldsymbol{u}} + \tilde{\boldsymbol{v}}^+ + \tilde{\boldsymbol{v}}^- \leq \boldsymbol{0} \\ \boldsymbol{0} \leq \tilde{\boldsymbol{v}}^+ \leq \boldsymbol{1}, -\boldsymbol{1} \leq \tilde{\boldsymbol{v}}^- \leq \boldsymbol{0} \\ \boldsymbol{0} \leq \tilde{\boldsymbol{u}} \leq \boldsymbol{1} \end{cases}$$

It is worth noticing that, to generate cuts, it is only necessary to construct the above cut generation programs, but not the fixed subproblems or their duals.

Although the generated Benders cut is valid, it may not be tight enough to cut off the current tentative master problem solution. This causes a problem for an iterative Benders algorithm such as the Algorithm 1. If, in some iteration, the tentative master problem solution is not cut off by any generated cut, then the subsequent iterations will stuck at the same tentative solution. Note that there

is no such problem if a branch-and-cut based Benders algorithm, as is suggested in [14, 3], is used, where the master problem is only solved once by a branching procedure and the Benders cuts are accumulated to guide the search.

However, the problem for iterative algorithms can be remedied by excluding the tentative solution from subsequent master problems with a no-good cut. As long as the master problem variables have a finite domain $\mathcal{D}_y$, one can formulate a no-good cut that only excludes $(\bar{\boldsymbol{y}}, \bar{z}_m)$. For example, consider $\mathcal{D}_y = \{0,1\}^{n_y}$. When the subproblem $\mathbf{SP}^m_{g\,\mathbf{F}}(\bar{\boldsymbol{y}})$ is infeasible, the following no-good cut excludes only $\bar{\boldsymbol{y}}$:

$$\textstyle\sum_{i=1}^{n_y} \bar{\boldsymbol{y}}_i(1-\boldsymbol{y}_i) + \sum_{i=1}^{n_y}(1-\bar{\boldsymbol{y}}_i)\boldsymbol{y}_i \geq 1 \tag{9}$$

When the subproblem $\mathbf{SP}^m_{g\,\mathbf{F}}(\bar{\boldsymbol{y}})$ is feasible but cannot reach the tentative $\bar{z}_m$, the following no-good cut excludes $(\bar{\boldsymbol{y}}, \bar{z}_m)$:

$$z_m \geq \phi_{\mathbf{SP}} - (\phi_{\mathbf{SP}} - z^L_m)[\textstyle\sum_{i=1}^{n_y} \bar{\boldsymbol{y}}_i(1-\boldsymbol{y}_i) + \sum_{i=1}^{n_y}(1-\bar{\boldsymbol{y}}_i)\boldsymbol{y}_i] \tag{10}$$

where $z^L_m \equiv \sum_{(\boldsymbol{d_m})_i<0}(\boldsymbol{d_m})_i$ is a lower bound of the variable z_m in $\mathbf{MP}$, and $\phi_{\mathbf{SP}}$ is the subproblem's objective value in the current iteration.

4.3 Generating Cuts from $\mathbf{SP}^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H})$

Applying the general method to problem $\mathbf{P}'$, we are now able to generate Benders cuts based on the subproblem formulations $\mathbf{SP}^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H})$. In this case the subproblems $\mathbf{SP}^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H})$ are feasibility problems, and therefore only feasibility cuts will be generated based on the homogeneous duals. Let $\mathcal{M}'$ denote the set of facilities where a cut needs to be generated. The corresponding cut generation programs can be formulated as is following. Note that in practice the IP formulations of the subproblems never need to be explicitly setup or solved.

$$\forall m \in \mathcal{M}' : \mathbf{HCGP}'^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H}) :$$

$$\max_{\tilde{u}^A_j, \tilde{u}^B_t, \tilde{u}^C_j, \tilde{v}^+_{jt}, \tilde{v}^-_{jt}} \sum_{j\in\mathcal{J}} \bar{y}_{jm}\tilde{u}^A_j + \sum_{t\in\mathcal{T}} C_m\tilde{u}^B_t + \sum_{j\in\mathcal{J}} \bar{H}\tilde{u}^C_j + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}_{jm}} \tilde{v}^-_{jt}$$

$$s.t. \begin{cases} \tilde{u}^A_j + \displaystyle\sum_{t'=t}^{t+p_{jm}-1} C_{jm}\tilde{u}^B_{t'} + (t+p_{jm})\tilde{u}^C_j + \tilde{v}^+_{jt} + \tilde{v}^-_{jt} \leq 0 \quad \forall j \in \mathcal{J}, \forall t \in \mathcal{T}_{jm} \\ -1 \leq \tilde{u}^A_j \leq 1, \;\; -1 \leq \tilde{u}^B_t \leq 0, \;\; -1 \leq \tilde{u}^C_j \leq 0 \\ 0 \leq \tilde{v}^+_{jt} \leq 1, \;\; -1 \leq \tilde{v}^-_{jt} \leq 0 \end{cases}$$

The subscript $\mathbf{IP}$ is used to emphasize that it is derived based on the IP formulation of the subproblem. Variables $\tilde{u}^A_j$, $\tilde{u}^B_t$, $\tilde{u}^C_j$ represent the dual variables associated with the first, second and third constraint of $\mathbf{SP}^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H})$ respectively.

Using the optimal values solved from the $\mathbf{HCGP}'^m_{\mathbf{IP}}(\bar{y}_{jm}, \bar{H})$, a Benders cut over the master problem variables can be generated:

$$\forall m \in \mathcal{M}' : \sum_{j\in\mathcal{J}} y_{jm}\tilde{u}^{A*}_j + \sum_{t\in\mathcal{T}} C_m\tilde{u}^{B*}_t + \sum_{j\in\mathcal{J}} H\tilde{u}^{C*}_j + \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}_{jm}} \tilde{v}^{-*}_{jt} \leq 0 \tag{11}$$

The above cut is valid according to Section 4.1 and Section 4.2, and it is eligible to be added to the master problem **MP**.

In case the cut (11) is not tight enough, a no-good cut needs to be formulated if the Algorithm 1 is used. If, on the mth facility, the tentatively assigned jobs cannot be scheduled in the facility at all (i.e. the optimization problem in $\mathbf{SP^m_{CP}}$ is already infeasible), then the no-good cut is given as:

$$\sum_{j\in\mathcal{J}_m} \bar{y}_{jm}(1-y_{jm}) \geq 1 \tag{12}$$

If the assigned jobs can be scheduled but not within the tentative completion time $\bar{H}$, then the no-good cut is given as:

$$H \geq H^*_{\mathbf{SP}} - H^*_{\mathbf{SP}} \sum_{j\in\mathcal{J}_m} \bar{y}_{jm}(1-y_{jm}) \tag{13}$$

where $H^*_{\mathbf{SP}}$ is the minimum completion time can be attained by the optimization problem in $\mathbf{SP^m_{CP}}$.

The cut generation phase (step 2.(3)) in the Algorithm 1 can now be specified as the Procedure 2, and the finite convergence of the algorithm is guaranteed.

Procedure 2 Cut Generation Phase for Problem $\mathbf{P'}$

2.(3) Cut Generation Phase.
For each m belonging to $\mathcal{M}'$:
(a) construct the cut generation program $\mathbf{HCGP'^m_{IP}}(\bar{y}^{(k)}_{jm}, \bar{H}^{(k)})$.
(b) generate a valid Benders cut (11) using the solution.
(c) if (11) does not cut off $(\bar{y}^{(k)}_{jm}, \bar{H}^{(k)})$, then generate a no-good cut (12) or (13).
Add the generated cuts to the master problem to construct $\mathbf{MP^{(k+1)}}$; set $k = k+1$ and go back to phase (1).

Lemma 3. *The Algorithm 1, with its Cut Generation Phase instantiated by the Procedure 2, converges to the optimal solution of* $\mathbf{P'}$ *in a finite number of iterations.*

5 Computational Experiments

The proposed algorithm is implemented to solve the general minimum completion time problem. Problem instances are randomly generated by a similar way as in [9], but different release and due dates are allowed. The size of a problem instance is specified by the number of facilities M and the number of jobs J. For each problem size configuration, 10 problem instances are randomly generated. The capacity C_m of each facility is set to 10. The consumption of resources C_{jm} is drawn from a uniform distribution on $[1, 10]$. The processing time p_{jm} for each job j on a certain facility m is drawn from a uniform distribution on $[m, 20m]$, rounded to the nearest integer. Thus the average processing speeds of the facilities are different. As the average of $20m$ over all facilities is $10(M+1)$, the total processing time for the jobs is roughly proportional to $10J(M+1)/M$ per facility (ref. [9]). The release date r_j for each job is drawn from a uniform

distribution on $[1, 10]$, rounded to the nearest integer. The due date d_j is calculated by r_j plus a time window w_j, which is calculated at one third of the value $10J(M + 1)/M$.

The algorithm is implemented in the ECL^{i}PS^e 5.8 [11] platform. The external solver used for solving the master problems and the cut generation programs is the XPRESS-MP 14.27 [5]. The cumulative scheduling subproblems are solved by constraint programming using the `ic_probe_for_scheduling` and `ic_edge_finding` libraries in the ECL^{i}PS^e. For comparison purpose, the test problems are also solved directly by the same external MIP solver. The formulation used for directly solving is $\mathbf{P}$, instead of $\mathbf{P}'$. To investigate the benefits of the Benders cuts of the form (11), we also implemented the hybrid algorithm with only the no-good cuts being generated. To show the effects of the CP component of the hybrid scheme, we implemented the decomposition algorithm with the subproblems solved by the MIP solver using the IP formulation. For all the algorithms, we set the timeout to 1800 seconds.

The computational results are summarized in Table 1. All numbers except for those in columns 'M', 'J' and '#TO' are average values. The first two columns record the problem size. The 'Optimal' shows the average optimal objective values. The computational results of the proposed algorithm are summarized under the heading 'Hybrid'. The solving times and numbers of iterations of the hybrid decomposition algorithm are shown in 'CPU' and '#Iter' respectively. In column 'CGT%', we give the percentage of solving time spent in the cut generation step. The results for the algorithm with no-good cuts only are recorded in 'Hybrid (NGC)'. The solving times and numbers of iterations are shown. The results for the non-hybrid algorithm without using CP for the subproblems (but still using the proposed Benders cuts) are shown in 'Non-hybrid'. As this variation of the algorithm dose not solve all instances within the time limit, we show the number of timeout cases (out of 10) in the column '#TO'. The solving times are given in 'CPU'. For comparison, the performance of directly MIP solving is summarized under 'Direct MIP'. Again as the MIP solver does not solve every problem instance within 1800 seconds, we show in column '#TO' the number of instances (out of 10) for which the MIP solving times out, and 'CPU' gives the solving time. Note that the values with a plus sign are computed using only the instances for which the corresponding solver does not time out. All other values are the average of 10 instances. The unit of times in the table is second.

The results show that the solving times of all methods increase as the problem scales, reflecting the growing complexity of the problem. Using the proposed hybrid method, all the tested problem instances are solved to optimality within the time limit, while directly MIP solving fails finding the optimality for some instances, and there are more timeout cases as the problem size increases. For smaller problems where both algorithms can prove optimality, the decomposition algorithm also spends much less solving time than the MIP solver. The results suggest that the proposed algorithm could be very useful in solving the minimum completion time problems efficiently. For other objectives, similar performance might be expected, but it is subject to further empirical study.

Table 1. Computational Results

M	J	Optimal	Hybrid			Hybrid (NGC)		Non-hybrid		Direct MIP	
			CPU	#Iter	CGT%	CPU	#Iter	#TO	CPU	#TO	CPU
2	8	31.2	0.87	10.7	13.3%	1.02	15.0	0	4.62	0	9.93
2	10	35.8	4.45	15.4	10.7%	6.03	25.3	0	53.06	1	213.29^{+}
2	12	38.9	131.11	48.0	6.3%	159.43	61.2	4	472.02^{+}	5	600.67^{+}
3	10	32.9	3.89	28.8	14.9%	4.03	33.6	0	19.24	0	115.25
3	12	36.2	19.89	53.0	11.5%	24.85	63.7	0	145.48	4	96.51^{+}
3	14	39.5	129.66	85.0	8.2%	153.59	102.7	3	448.52^{+}	6	735.87^{+}
4	12	31.6	4.28	19.6	12.6%	7.66	32.6	0	51.14	5	109.46^{+}
4	14	34.6	47.26	54.1	7.8%	135.43	83.3	2	427.69^{+}	8	132.25^{+}
4	16	34.0	148.97	78.8	7.4%	396.48	105.8	4	209.34^{+}	7	159.26^{+}

Next, the effects of incorporating the proposed Benders cuts are studied. Firstly, the results show that the overheads of cut generation account for a small portion of the total solving time, and the percentage decreases as the problem scales. Secondly, compared with the alternative algorithm with no-good cuts only, the algorithm that uses the proposed Benders cuts experiences less iterations. This difference becomes substantial in some larger problems. In terms of solving time, the algorithm also consistently outperforms the one with no-good cuts, in spite of the overheads incurred by solving the cut generation programs. The comparison indicates that the Benders cuts of the form (11) are indeed useful in improving the performance, yet without incurring too much overheads.

Finally, to show the effects of the CP component in the hybrid scheme, we compare the hybrid algorithm with an algorithm that uses MIP to solve both the master problem and the subproblems. The results show that the non-hybrid algorithm is substantially slower than the hybrid one, although its performance is still much better than that of directly MIP solving. Without CP, the algorithm even fails finding the optimal solution (within 1800 seconds) for a few instances. This difference could be attributed to the fact that the scheduling subproblems are often hard to solve, and that the employed CP methods, which are specially designed for single machine scheduling problems, are much more efficient than the MIP solver. The results show that the incorporation of CP solution methods indeed plays an important role in the proposed algorithm, in reducing the solving times for the considered class of problems.

6 Conclusions

This paper presents a hybrid method for the resource constrained scheduling problems. Different models and solvers are used in the three components of the hybrid scheme. In particular, the cut generation uses the dual information based on the integer programming model under a Benders decomposition framework. The approach has been instantiated to an algorithm for the minimum completion time problem. Computational results have shown that the proposed algorithm achieves substantial reduction of solving times, especially for larger problem instances.

References

1. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, 2001.
2. J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.
3. A. Bockmayr and N. Pisaruk. Detecting infeasibility and generating cuts for mip using cp. In *Proc.of CPAIOR 03, Montreal, Canada*, 2003.
4. Y. Chu and Q. Xia. Generating benders cuts for a general class of integer programming problems. In M. Rueher J.-C. Regin, editor, *Lecture Notes in Computer Science 3011 – CPAIOR 04*, pages 127–141. Springer-Verlag, 2004.
5. Dash Inc. *Dash XPRESS-MP 14.27 User's Manual*, 2003.
6. H. El Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5(4):359–388, 2000.
7. A.M. Geoffrion. Generalised benders decomposition. *Journal of Optimization Theory and Application*, 10:237–260, 1972.
8. Iiro Harjunkoski and I.E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Comp. and Chem. Engineering*, 26:1533–1552, 2002.
9. J.N. Hooker. A hybrid method for planning and scheduling. In M. Wallace, editor, *Lecture Notes in Computer Science 3258 – Principles and Practice of Constraint Programming CP 04*, pages 305–316. Springer-Verlag, 2004.
10. J.N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96:33–60, 2003.
11. Imperial College London. *ECLiPSe 5.8 User's Manual*, 2004.
12. V. Jain and I.E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimisation problems. *INFORMS Journal on Computing*, 13:258–276, 2001.
13. R. Sadykov. A hybrid branch-and-cut algorithm for the one-machine scheduling problem. In M. Rueher J.-C. Regin, editor, *Lecture Notes in Computer Science 3011 – CPAIOR 04*, pages 409–415. Springer-Verlag, 2004.
14. E.S. Thorsteinsson. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh, editor, *Lecture Notes in Computer Science 2239 – Principles and Practice of Constraint Programming CP 01*, pages 16–30. Springer-Verlag, 2001.
15. M. Turkay and I.E. Grossmann. Logic-based minlp algorithms for the optimal synthesis of process networks. *Comp. and Chem. Engineering*, 20:959–978, 1996.

Appendix: Proofs of Lemmas

Lemma 1:

Proof. To prove the validity, let $(\hat{\boldsymbol{y}}, \hat{z}_1, \cdots, \hat{z}_M)$ be any solution of $\mathbf{MP}_g$ such that the value of $\hat{z}_m$ can be reached by the subproblem parameterized by $\hat{\boldsymbol{y}}$ for each m. We prove that this solution is not cut off by the optimality inequality (4) Consider the mth subproblem. We have:

$$\hat{z}_m \geq \phi(\mathbf{SP}_g^m(\hat{\boldsymbol{y}})) = \min_{\boldsymbol{x_m}} \{\boldsymbol{d_m^T x_m} : \boldsymbol{B_m x_m} \geq \boldsymbol{b_m} - \boldsymbol{A_m \hat{y}}, \boldsymbol{x_m} \in \{0,1\}^{n_m}\} \quad (14)$$

where $\phi(\cdot)$ is the value function of a program. Then there must exist a value $\hat{\boldsymbol{x}}_{\boldsymbol{m}}$ such that $\hat{z}_m \geq \phi(\mathbf{SP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})) = \phi(\mathbf{DSP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}}))$, i.e., $\hat{z}_m \geq (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\hat{\boldsymbol{v}}$, where $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ is an *optimal* solution of $\mathbf{DSP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})$.
Note that the feasible region of $\mathbf{DSP}^m_{g\,\mathbf{F}}(\boldsymbol{y}, \boldsymbol{x}_{\boldsymbol{m}})$ is independent of the value of $\boldsymbol{y}$ and $\boldsymbol{x}_{\boldsymbol{m}}$. Therefore, the dual value $(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ (used by the optimality inequality (4)), which is feasible for $\mathbf{DSP}^m_{g\,\mathbf{F}}(\bar{\boldsymbol{y}}, \tilde{\boldsymbol{x}}_{\boldsymbol{m}})$, is also a *feasible* solution of $\mathbf{DSP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})$. This implies that

$$(\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\tilde{\boldsymbol{v}} \leq (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\hat{\boldsymbol{v}} \leq \hat{z}_m$$

Due to the assumption (6), we have $\tilde{\boldsymbol{x}}^T_{\boldsymbol{m}}\tilde{\boldsymbol{v}} \leq \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\tilde{\boldsymbol{v}}$ no matter which binary values the variables $\hat{\boldsymbol{x}}_{\boldsymbol{m}}$ take. Thus,

$$(\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\tilde{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T_{\boldsymbol{m}}\tilde{\boldsymbol{v}} \leq (\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\tilde{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\tilde{\boldsymbol{v}} \leq \hat{z}_m$$

i.e. the optimality inequality (4) is satisfied by $(\hat{\boldsymbol{y}}, \hat{z}_m)$. □

Lemma 2:

Proof. Similar to the proof of Lemma 1, let $(\hat{\boldsymbol{y}}, \hat{z}_1, \cdots, \hat{z}_M)$ be any feasible solution of $\mathbf{MP}_g$. We prove that it is not cut off by the feasibility inequality (5). Consider the mth subproblem. Since it is feasible, there must exist a value $\hat{\boldsymbol{x}}_{\boldsymbol{m}}$ such that $\mathbf{SP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})$ is feasible, and therefore the corresponding homogeneous dual $\mathbf{HDSP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})$ has a non-positive optimal value, i.e., $(\boldsymbol{b}_{\boldsymbol{m}} - \boldsymbol{A}_{\boldsymbol{m}}\hat{\boldsymbol{y}})^T\hat{\boldsymbol{u}} + \hat{\boldsymbol{x}}^T_{\boldsymbol{m}}\hat{\boldsymbol{v}} \leq 0$, where $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ is an *optimal* solution of $\mathbf{HDSP}^m_{g\,\mathbf{F}}(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}_{\boldsymbol{m}})$. Then applying the same reasoning as in the proof of Lemma 1, the conclusion follows. □

Lemma 3:

Proof. First we show that the algorithm terminates in a finite number of iterations. Due to the cut generation procedure, in each iteration except for the last, the current tentative solution from the master problem is cut off, and the value of $(\bar{y}_{jm}, \bar{H})$ is different in different iterations. Since the master problem variables have a finite domain, the algorithm has to terminate in finite iterations.
Next we show that the algorithm returns the optimal solution of the original program $\mathbf{P}'$. Note that the master problem $\mathbf{MP}^{(\mathbf{k})}$ is always a relaxation of $\mathbf{P}'$ as all the cuts added are valid. If the Algorithm 1 terminates in step 2.(1), then the original problem has to be infeasible as well. If it terminates in step 2.(2) of some iteration k, then the tentative assignment in this iteration renders feasible subproblems on every machine, and the value of $\bar{H}^{(k)}$ can be attained by the subproblems. As $\mathbf{MP}^{(\mathbf{k})}$ is a relaxation of $\mathbf{P}'$, $\bar{H}^{(k)}$ is always a lower bound of the optimal solution of $\mathbf{P}'$. Thus, $\bar{H}^{(k)}$ is a minimum completion time that can be achieved. The current assignment is the optimal solution for variables y_{jm}. The optimal starting times of the assigned jobs can be obtained from the subproblems. □