Single-Facility Scheduling over Long Time Horizons by Logic-Based Benders Decomposition

Elvin Coban, John N. Hooker
Tepper School of Business, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15212-3890, USA, ecoban@andrew.cmu.edu

Abstract
We use logic-based Benders decomposition to minimize tardiness in single-facility scheduling problems with many jobs and long time horizons. Release dates and due dates are given. A mixed integer linear programming (MILP) based master problem allocates jobs to segments of the time horizon, and a constraint programming (CP) based subproblem schedules the jobs in each segment.

1 Introduction
Scheduling is a decision-making process occurring frequently not only in service industries but also in manufacturing. The allocation of limited resources to jobs over time is the main concern. The resources and the jobs have many forms. For example, the resources may be machines, crews at a construction site, or processing units in a computing environment. The respective jobs may be operations in a production process, stages in construction project, executions of computer programs, and so on. Each job may have different priority level, release and due dates. Minimization of completion time of the last job (makespan), minimization of total tardiness, minimization of total cost are some possible objective functions of scheduling problems [20].

We study a single-facility scheduling over long time horizons. The goal is to assign jobs to facilities subject to the release and due dates of the jobs and minimize total tardiness.

Single-facility scheduling problems can be thought as simple and rare. In practice a complicated facility environment is often decomposed into subproblems that deal with single facility. The results obtained by single-facility models provide a basis for heuristics for more complicated facility environments [20]. Furthermore, there are obvious cases involving single facility scheduling, e.g. the processing of jobs through a small non-time-sharing computer. There are less obvious cases such as a large complex plant acts as if it was one facility, e.g. in paint manufacture whole plant may have to be committed to produce one color of paint at a time [6]. A complicated facility environment with a single bottleneck may result also in a single facility model [20]. Additionally, the problem of minimizing the total tardiness on a facility is shown to be NP-hard [5]. When the time horizon is long, the problem becomes harder as the possible number of slots where a job can be scheduled may increase.

We consider the single-facility scheduling problem for many jobs and a long time horizon as a multiple facility scheduling problem. We divide the long time horizon into segments and consider each segment as a fictitious facility. The objective function is to minimize total tardiness, in addition all jobs might have different release and due dates.

We can decompose the problem into two portions: assignment, and scheduling. Central managers assign jobs to the facilities (in our work, fictitious facilities), whereas operations managers prepare detailed schedules for each facility. If the assignment done by central managers causes infeasibility, then operations managers call central managers and ask for a different allocation of jobs. Until a mutually satisfactory solution is reached, the calls will be necessary to obtain a feasible schedule. The solution found at the end might not be necessarily a good solution [13].

Benders decomposition can be seen as a form of “learning from mistakes”. In this problem-solving strategy we assign some of the variables trial values and find the best solution consistent with these solutions. Then, information about the quality of other trial solutions is obtained and used for reducing the number of solutions
we have to generate to find the optimal solution [10]. Applying Benders decomposition is one of the possible approaches mimicking the feedback loop between central and operations managers. The Benders master problem is the planning problem by which jobs are allocated for each facility, and the subproblem separates into several independent scheduling problems. Benders cuts are the telephone calls [13].

The classical Benders approach [1, 7] requires the subproblem to be a continuous linear or nonlinear programming problem. However, scheduling is a highly combinatorial problem without any practical linear or nonlinear programming model [13]. Furthermore, even if integer programming (IP) has been successfully applied to a wide range of problems, such as capital budgeting, bin packing, etc., in the last two decades CP has been shown to be a flexible, efficient and commercially successful technique for scheduling problems. IP models may become very large and intractable as scheduling problems involve permutations [15].

We first decompose single-facility scheduling problem with long time horizons into assignment (master problem) and scheduling (subproblem) parts. Then, scheduling part decouples for each facility. By solving separate scheduling problems for each fictitious facility in the subproblem, we take advantage of working with smaller problems. Therefore, in this paper we extend Benders decomposition to a logic based context to solve single-facility scheduling problem with long time horizons by integrating CP and MILP.

We can also model the scheduling problem entirely as MILP or CP but these models are hard to solve because MILP is not effective for scheduling problems and CP is not effective for optimization problems with complex structure and many feasible solutions. By integrating them, we take advantage of relative strengths of MILP and CP [13, 19].

In contrast to classical Benders, unfortunately, there is no standard form to generate Benders cuts in logic-based Benders decomposition. Thus, we develop Benders cuts specific to minimum total tardiness problem in which jobs may have different release and due dates. We also develop relaxations of the subproblem in the master problem to obtain better performance.

2 Previous Work

In the last two decades, extending Benders decomposition to logic-based context becomes increasingly attractive as substantial speedups can be achieved. By modeling some problems entirely as MILP or CP may not be effective as MILP performs good for optimization problems and CP performs good when highly constrained discrete optimization problems are considered and finding a feasible solution is enough [17]. Thus, there is a growing trend for integrating CP and MILP.

Certainly, while integrating CP and MILP, one has to highlight inference duality, which is a general approach to sensitivity analysis that applies to both continuous and discrete problems. Key point is inferring from the constraints a best possible bound on the optimal value. A generalization of Benders decomposition to any optimization problem is achievable by inference duality. In logic-based Benders, we obtain the Benders cuts by solving the inference dual of the subproblem. The solution of the inference dual is a proof of optimality for the fixed values of master problem variables (primary variables) and by using the same proof, Benders cut is developed. Hence, a valid bound is obtained on the optimal value when the primary variables take other values. [11, 13]. (Details of inference duality can be found in Section 7.1.)

Hooker and Yan introduced the logic-based Benders decomposition for circuit verification in [16] (whether a logic circuit correctly implements a given boolean function). The authors concluded that Benders approach performed better than BDD for circuits that do not contain many irreducible gates. In [10], Hooker extended the Benders decomposition to a logic-based context for propositional satisfiability and 0-1 programming problem.

In [15], the authors exploited the CP and LP structure. Unifying modeling and solution framework are proposed in the paper to show the strength of integrating CP and LP. Motivated by the complementary strength
of CP and MILP, Jain and Grossman applied logic-based Benders to multiple-machine scheduling [17]. When relaxation is tight and optimization is the dominating aspect, MILP is very efficient. However, for highly constrained discrete optimization problems and when feasibility is the main concern, CP works better than MILP. Therefore, while considering minimum-cost planning and scheduling problems, the authors exploited integration of MILP and CP. According to computational results, the hybrid models are shown to perform two to three orders of magnitude reduction in CPU time compared to standalone MILP and CP models [17]. The subproblems are disjunctive scheduling problems (jobs must be run one at a time) which is a similar case to our work. However, the objective function is to minimize cost which is an easy problem. Harjunkoski and Grossmann considered multistage scheduling problems in which the subproblems are feasibility problems rather than optimization problems [9]. Thus, the Benders cuts are generally simple.

Hooker and Ottosson applied logic-based Benders decomposition to the propositional satisfiability problem, the 0-1 programming problem, and a machine scheduling problem. They concluded that logic-based Benders approach can be applied to any optimization or feasibility problem as inference duality can be defined for these problems. The performance depends on not only defining the variables (primary and secondary variables) and the subproblem solvability but also on developed Benders cuts [14]. Furthermore, in [3] logic-based Benders methods were adapted while solving integer programming variables. Minimal dispatching of automated guided vehicles [4], steel production scheduling [8], real-time scheduling of computer processors [2], traffic diversion [3], batch scheduling in a chemical plant [19], and polypropylene batch scheduling [21] are examples in which similar ideas were applied. However, in all these examples, the subproblem is a feasibility problem, not an optimization problem.

Additionally, in [18], the author provided a broad review and extensions of single-facility scheduling problems in which the objective functions are to minimize total tardiness. However, the release dates are equal to zero which makes the problem simple.

3 The Problem

We consider single-facility scheduling problems for many jobs and long time horizons. The objective function is to minimize total tardiness, in addition all jobs might have different release and due dates. We divide the long time horizon into segments and consider each segment as a facility, represented in Figure 1. \(z_1\) and \(z_{m+1}\) are the start and finish time of the long time horizon, respectively. Each artificial facility is assumed to be one time segment. As represented in Figure 1, time segments can differ in length. With these time segments, we create \(m\) artificial facilities (i.e. \(i \in \{1,2,\ldots,m\}\)). Currently, in the master problem and subproblem we assume that time segments are equal for the sake of simplicity.

![Figure 1: Long time horizon divided into time segments.](image)

3.1 Mixed Integer Linear Programming Model

Each job \(j \in \{1,\ldots,n\}\) will be assigned to fictitious facility \(i \in \{1,\ldots,m\}\). \(p_j, r_j,\) and \(d_j\) represent the processing time, the release date and the due date for job \(j\), respectively. If job \(j\) will start at time \(t\), then the decision variable, \(x_{jt}\), is equal to 1, and 0 otherwise. As \(t\) carries the information about what time the job starts, finding
to which fictitious facility job is assigned becomes trivial. Thus, there is no extra index, $i$, at decision variable $x_{jt}$ representing facility.

If we want to model the scheduling problem entirely as MILP, then the model is

$$\min \sum_j T_j$$
\begin{align*}
s.t. & \sum_t x_{jt} = 1 \text{ for all } j \quad (1) \\
& x_{jt} + x_{j\overline{t}} \leq 1 \text{ for all } j, \overline{t}, t, \text{ and } \overline{t} = t, ..., t + p_j - 1 \quad (2) \\
& x_{jt} = 0 \text{ for all } t < r_j \quad (3) \\
& x_{jt} = 0 \text{ for all } i, j, t, \text{ where } t = z_{i+1} - p_j + 1, ..., z_{i+1} - 1 \quad (4) \\
& T_j \geq \sum_t (t + p_j) * x_{jt} - d_j \text{ for all } j \quad (5) \\
& x_{jt} \in \{0, 1\} \text{ for all } j, t \quad (6) \\
& T_j \geq 0 \text{ for all } j \quad (7)
\end{align*}

Every job must be assigned to one of the facilities is represented in (2). (3) prevents overlap of jobs. A job can not start when there is a job which has already started and not finished yet. A job cannot start before its release date is represented in (4). Additionally, a job cannot overlap from one time segment (namely, fictitious facility) to another time segment. Therefore, pre-emption is prevented by (5). Relaxing this constraint is a future research.

Currently, we are motivated by the case that there may be recurrent times when a facility will not work, such as weekends, holidays, and so on. We consider our fictitious facilities are divided by such breaks. For instance, consider a facility, which is working during weekdays and is closed during weekends. Additionally, any job has to be finished during weekdays. Thus, preventing pre-emption is rational. Tardiness is defined by (6), and (8).

### 3.2 Constraint Programming Model

$y_j$ and $z_{ij}$ are decision variables representing the time intervals during which job $j$ and assignment of job $j$ to facility $i$ are done, respectively. $w_i$ is another decision variable of the subproblem defining the sequence of $y_j$ time intervals at facility $i$. Then, if we want to model scheduling problem entirely as CP, the model is

$$\min \sum_j \max\{endOf (y_j) - d_j, 0\};$$
\begin{align*}
s.t. & \text{startOf} (y_j) \geq r_j \text{ for all } j \quad (9) \\
& \text{alternative} (y_j, \text{ all } i \text{ in Machines} \ z_{ij}) \text{ for all } j \quad (10) \\
& \text{noOverlap} (w_i) \text{ for all } i \quad (11) \\
& \text{noOverlap} (w_i) \text{ for all } i \quad (12)
\end{align*}

Minimization of total tardiness is formulated by (9). A job can start only after its release date. This constraint is formulated by (10). By (11), a time interval during which a job is done will correspond to a time interval on the respective facility. Lastly, the sequence of jobs’ time intervals will not overlap on each facility. In other words, only one job can be processed at a time.

We formulate CP model by using the high-level modeling language of ILOG OPL Development Studio 6.1.1 for computational analysis. The code for CP version of minimization of total tardiness can be found in Section 7.2.
In logic-based Benders decomposition, we acquire Benders cuts by solving the inference dual of the subproblem. Firstly, primary variables (variables of master problem) are distinguished from secondary variables (variables of subproblem). Secondly, planners solve the master problem for fixing primary variables to some trial value. There may be two possible outcomes: either master problem is feasible or infeasible. If master problem is infeasible, planners stop trying to find the optimal solution as there is no optimal solution for this instance.

On the other hand, if the master problem is feasible, the primary variables are fixed and the solution of the inference duality becomes a proof of optimality. For each trial value of primary variables, schedulers solve the subproblem which is decoupled for each facility. If the solution is suboptimal, planners and schedulers search for its reasons and develop constraints, Benders cuts, by following the same proof and obtain a valid bound on the optimal value to reduce the number of possible solutions. Benders cuts may not only eliminate the current suboptimal solution but also may eliminate a class of solutions for the same reason. As the Benders cuts are added to the master problem, which is formulated as MILP, the Benders cuts have to be formulated as linear inequalities. Afterwards, planners resolve the master problem including the Benders cuts. Logic-based Benders method is represented in Figure 2.

![Figure 2: Logic-based Benders method.](image)
4.1 Master Problem and Subproblem

In the Benders approach by the master problem jobs are assigned to fictitious facilities and by the subproblem jobs are scheduled. Before mentioning the Benders cut and the subproblem relaxations proposed, we would like to introduce the master problem and subproblem of single-facility scheduling with long time horizons.

New decision variables are introduced in the master problem. \( T_{ij} \) and \( \tilde{T}_i \) represent tardiness of job \( j \) at facility \( i \), and total tardiness, respectively. Furthermore, let \( Z_i \) is the set of time periods representing time window of fictitious facility \( i \) \( (Z_i = \{ t : z_i \leq t \leq z_{i+1} - 1 \}) \). In iteration \( h \) of Benders method, the master problem is

\[
\min_{T} T \quad \text{(13)}
\]

s.t.

\[
\sum_{t} x_{jt} = 1 \text{ for all } j \quad \text{(14)}
\]

\[
x_{jt} + x_{j\tilde{T}} \leq 1 \text{ for all } j, \tilde{T}, t, \text{ and } \tilde{T} = t, ..., t + p_j - 1 \quad \text{(15)}
\]

\[
x_{jt} = 0 \text{ for all } t < r_j \quad \text{(16)}
\]

\[
x_{jt} = 0 \text{ for all } i, j, t, \text{ where } t = z_{i+1} - p_j + 1, ..., z_{i+1} - 1 \quad \text{(17)}
\]

\[
T_{ij} \geq \sum_{t \in Z_i} (t + p_j - d_j) \cdot x_{jt} \text{ for all } i, j \quad \text{(18)}
\]

\[
T \geq \sum_{i} \sum_{j} T_{ij} \quad \text{(19)}
\]

\[
x_{jt} \in \{0, 1\} \text{ for all } j, t \quad \text{(20)}
\]

\[
T_{ij} \geq 0 \text{ for all } i, j \quad \text{(21)}
\]

Benders cuts for iteration 1, ..., \( h - 1 \) (22)

relaxations of subproblem (23)

The objective function, (13), is represented as minimization of total tardiness. (14), (15), (16), (17), and (20) are same as the constraints in MILP model mentioned in Section 3.2. (18) and (21) are for defining \( T_{ij} \) according to tardiness definition. Instead of introducing decision variable \( T_{ij} \), defining a decision variable for tardiness with only facility index \( i \) \( (\tilde{T}_i) \), may seem enough. However, because of the nonnegativity aspect of tardiness, introducing \( \tilde{T}_i \) is not enough.

Suppose \( \tilde{T}_i \) is the decision variable in MILP formulation instead of \( T_{ij} \) as shown in (24). While calculating tardiness of job \( j \), we need to take maximum of 0 and \( \sum_{t \in Z_i} (t + p_j - d_j) \cdot x_{jt} \). In (24) we sum \( (t + p_j - d_j) \cdot x_{jt} \) over every job \( j \) and over every time period \( t \in Z_i \). Then, we consider the nonnegativity of the summation \( (\tilde{T}_i \geq 0) \).

Assume job \( k \) is not tardy and is assigned to facility \( i \) (If a job is not tardy, then \( \sum_{t \in Z_i} (t + p_j - d_j) \cdot x_{jt} \leq 0 \)). Also assume \( \sum_{t \in Z_i} (t + p_k - d_k) \cdot x_{kt} < 0 \). If we consider the nonnegativity after summing the terms \( \sum_{t \in Z_i} (t + p_j - d_j) \cdot x_{jt} \) over every job \( j \) assigned to facility \( i \), we will obtain wrong value for facility \( i \)’s tardiness as we decrease the real tardiness by \( \sum_{t \in Z_i} (t + p_k - d_k) \cdot x_{kt} \) amount.

\[
\tilde{T}_i \geq \sum_{j} (\sum_{t \in Z_i} (t + p_j - d_j) \cdot x_{jt}) \text{ for all } i \quad \text{(24)}
\]

The subproblem decouples into a single-facility scheduling problem for each facility \( i \). Let \( J_{hi} \) be a set of jobs
assigned to facility $i$ in iteration $h$. Then, the subproblem becomes

$$
\begin{align*}
\min & \sum_j \max\{\text{endOf}(y_j) - d_j, 0\} \quad \text{for all } j \in J_{hi} \\
\text{s.t.} & \quad \text{startOf}(y_j) \geq r_j \quad \text{for all } j \in J_{hi} \\
& \quad \text{startOf}(y_j) \geq z_i \quad \text{for all } j \in J_{hi} \\
& \quad \text{endOf}(y_j) \leq z_{i+1} \quad \text{for all } j \in J_{hi} \\
& \quad \text{noOverlap}(w_i) \quad \text{for all } i
\end{align*}
$$

(25) and (26) are same as in Section 3.2. Only, as we solve CP one by one for each facility, we consider all constraints will be valid for the jobs $j$ which are assigned to respective machine. (27) and (28) satisfy the time interval decision variables will be scheduled in the time windows of the respective fictitious machine.

4.2 Benders Cuts

Logic-based Benders cuts need to provide a valid bound for any given fixed primary variables’ value during iterations. For the sake of simplicity and tractability, let $\bar{T}_i = \sum_j T_{ij}$ where $j = 1, \ldots, n$. Additionally, let $J_{hi}$ be a set of jobs assigned to facility $i$ in iteration $h$ and let $Z_i$ is the set of time periods representing time window of fictitious facility $i$ ($Z_i = \{t : z_i \leq t \leq z_{i+1} - 1\}$).

Suppose the current tardiness of facility $i$ is $\bar{T}^0_i$ at iteration $h$ ($\bar{T}^0_i = \sum_{j \in J_{hi}} T^0_{ij}$). We propose Benders cut in (30) that will give bound on total tardiness of facility $i$ that can be stated after removing some jobs which are previously assigned to facility $i$.

Let $t^0_j$ be start time of job $j$ in the current schedule and suppose jobs are all indexed such that $t^0_1 \leq t^0_2 \leq \ldots \leq t^0_n$ (we know the previous schedule of the jobs from the scheduling problem). $x_{kt}$ is the decision variables of master problem (assignment variables). $x_{kt}$ takes value 1 when job $k$ starts at time $t$, otherwise 0.

$B_k$ is the maximum amount of decrease in tardiness of facility $i$ when we remove job $k$ from facility $i$. We consider each job $j$ which is performed after job $k$ ($j > k$) and we check how much the tardiness of each job $j$ will decrease if we remove job $k$. Then, we sum over all jobs which are performed after job $k$ ($j > k$ where $j = 1, \ldots, n$). $(1 - x_{kt})$ is multiplied with $B_k$ because if job $k$ is not removed ($\sum_{t \in Z_i} x_{kt} = 1$), then we do not want to decrease the current tardiness. Furthermore, $T^0_j$ and $T^0_k$ in (31) are current tardiness of job $j$ and current tardiness of the removed job (job $k$). Then, the Benders cut in iteration $h + 1$ becomes

$$
\bar{T}_i \geq \bar{T}^0_i - \sum_{k \in J_{hi}} B_k (1 - \sum_{t \in Z_i} x_{kt}) \quad \text{for all } i
$$

(30)

where

$$
B_k = \sum_{j > k} \min\{p_k, \min_{l=k+1}^j (t^0_l - r_l), T^0_j\} + T^0_k
$$

(31)

Let’s consider a job $j$ which is performed just after job $k$ and job $k$ is removed. There will be three possible conditions resulting from the removal of job $k$, represented in Figure 3. We choose to show the conditions in which we can observe tardiness and a positive decrease in tardiness if we remove job $k$. However, there are other possible conditions in which removing job $k$ does not decrease tardiness (minimum of the three components in
and $T_k$ may be zero).

In part a of Figure 3, the schedule before removing job $k$ is shown. In part b, the new schedule with the change in tardiness is represented. Red colored line shows the reduction in tardiness of job $j$ if we remove job $k$. We consider there are some other jobs performed before job $k$ (because in case ii and iii, job $k$ cannot start right after its release time). The brackets, “[” and “]”, show the release and due dates of jobs, respectively.

Figure 3: Examples for three cases realized while calculating $B_k$. Tardiness decreases by i. $p_k$ ii. $t^0_j - r_j$ iii. $T^0_j$.

4.3 Subproblem Relaxation

It is hard to find tight and valid relaxations. We have tried several candidate relaxations and checked if they are always valid. Unfortunately, we could not find a tight subproblem relaxation because of diverse release dates, therefore, we modify the previous relaxations for our problem.

We change the relaxation in [13] by adding $z_i$ to (33) and (34). $z_i$ is the start time of the fictitious facility $i$’s time interval (namely, respective time segment). $T_{ik}$ represents tardiness of job $k$ at facility $i$ where $T$ is total tardiness. Let $Z_i$ is the set of time periods representing time window of fictitious facility $i$ ($Z_i = \{t: z_i \leq t \leq z_{i+1} - 1\}$).

Let $\pi(j)$ be an ordering such that $p_{\pi(1)} \leq p_{\pi(2)} \leq p_{\pi(3)} \leq ... \leq p_{\pi(n)}$. Additionally, the jobs will be indexed according to their deadlines ($d_1 \leq d_2 \leq d_3 \leq ... \leq d_n$). Otherwise, this relaxation will not be valid. By this relaxation, a lower bound on tardiness incurred by the job with the $k^{th}$ latest deadline will be obtained by
assuming that the job will finish no sooner than the job interval of the $k$ jobs with the smallest processing times. We also add $z_i$ on the right hand side of (33) because if job $k$ is assigned to facility $i$, there is no chance that job $k$ may start earlier than the start time of the facility’s time window (namely, $z_i$).

Right-hand side of (32) is actually formulated as $\sum_i \sum_k T_{ik}(\sum_{t \in Z_i} x_{kt})$. However, if we formulate like multiplication of two decision variables, then the constraint will be nonlinear. Therefore, we linearize the bound by including $U_{ik}$ which is the maximum value that $z_i + \sum_{j=1}^{k} p_{\pi(j)} \sum_{t \in Z_i} x_{\pi(j)t} - d_k$ can take ($\sum_{t \in Z_i} x_{\pi(j)t} = 1$ for all first $k$ jobs in $\pi(j)$ ordering). Whenever job $k$ is not assigned to facility $i$ ($\sum_{t \in Z_i} x_{kt} = 0$, we subtract $U_{ik}$ from the right hand side of (33) so $T_{ik}$ becomes either equal to 0 or less than 0. $U_{ik}$ can be negative, however, right-hand side is never positive when $\sum_{t \in Z_i} x_{kt} = 0$). Lastly, nonnegativity feature of tardiness is represented in (35).

\[
T \geq \sum_i \sum_k T_{ik}
\]  
(32)

\[
T_{ik} \geq z_i + \sum_{j=1}^{k} p_{\pi(j)} \sum_{t \in Z_i} x_{\pi(j)t} - d_k - (1 - \sum_{t \in Z_i} x_{kt})U_{ik}
\]  
(33)

\[
U_{ik} = z_i + \sum_{j=1}^{k} p_{\pi(j)} - d_k
\]  
(34)

\[
T_{ik} \geq 0
\]  
(35)

The second relaxation is also similar to the relaxation in [13]. $J(0, d_k)$ is the set of jobs with time windows between 0 and $d_k$ and we do not assume that the jobs are indexed according to their deadlines. For any $k$, the last scheduled job in the set $J(0, d_k)$ can finish no earlier than time $t = \sum_{j \in J(0, d_k)} p_j$. The last task has due date no later than $d_k$. In our problem, we may have different release dates for jobs so we adjust the relaxation by adding the minimum of the release dates on the right-hand side of (37).

\[
T \geq \sum_i \tilde{T}_i^L
\]  
(36)

\[
\tilde{T}_i^L \geq \min\{r_j\} + \sum_{j \in J(0, d_k)} (p_j \sum_{t \in Z_i} x_{jt}) - d_k, \text{ for all } i
\]  
(37)

\[
\tilde{T}_i^L \geq 0
\]  
(38)

The third relaxation is from [12]. $J(t_1, t_2)$ is the set of the jobs with time windows between $t_1$ and $t_2$. With (39), we prevent over-scheduling of jobs between time $t_1$ and $t_2$. For $t_1$ and $t_2$, our current approach is considering the time windows of each fictitious facility, such as $t_1 = z_1$ and $t_2 = z_{i+1} - 1$. However, it is also reasonable to select some large sets $J(r_j, d_k)$. Let $\tilde{Z}_{t_1, t_2}$ is the set of time periods between $t_1$ and $t_2$ ($\tilde{Z}_{t_1, t_2} = \{ t : t_1 \leq t \leq t_2 \}$), then, the third relaxation becomes

\[
\sum_{j \in J(t_1, t_2)} p_j \sum_{t \in \tilde{Z}_{t_1, t_2}} x_{jt} \leq t_2 - t_1
\]  
(39)

9
5 Problem Generation

Random instances were generated from the uniform distribution on the intervals mentioned in Table 1. \( R \) is a large number representing the long time horizon of the single-facility (not fictitious facilities). Release date of each job, \( r_j \), were generated from the uniform distribution on the interval \([0, R]\) or \([0, \alpha \ast R]\). For every job \( j \), \( r_j \) had to be in the time window of the single facility. When we multiplied the time horizon of the facility by a small coefficient, \( \alpha \), each release date can at most take value \( \alpha \ast R \). Therefore, we can say that the interval \([0, \alpha \ast R]\) may be more denser than the case in which we generated each \( r_j \) from the interval \([0, R]\). Generating \( r_j \) more densely at the beginning of horizon may allow us to obtain different combinations of jobs’ schedules.

The length of time windows \((d_j - r_j)\) for every job were generated from the uniform distribution on the interval \([0, \alpha \ast R]\) or \([1/4 \ast \alpha \ast R, \alpha \ast R]\). We left space for jobs to be tardy as we did not allow the time windows to spread from 0 to \( R \). We multiplied \( R \) with small coefficients while generating time windows. In other words, we can say that we shortened the interval from which time windows’ length will be generated. After creating the time windows, we generated the processing time of every job in the respective time window. We can also multiply the length of the time windows with a small coefficient to let jobs to have several configurations without being tardy. Otherwise, their \( r_j \), \( d_j \) and \( p_j \) will define at which fictitious facility they will be assigned to, without solving the models, which will make the scheduling problem simpler.

| Data   | Interval                          |
|--------|-----------------------------------|
| \( r_j \) | \([0, R]\) or \([0, \alpha \ast R]\) |
| \( d_j - r_j \) | \([0, \alpha \ast R]\) or \([1/4 \ast \alpha \ast R, \alpha \ast R]\) |
| \( p_j \) | \([0, d_j - r_j]\) or \([0, \beta \ast (d_j - r_j)]\) |

6 Conclusions and Future Research

We integrate CP and MILP through a Benders scheme for single-facility scheduling problem with long-time horizons. By the nature of the problem, we can first decompose the scheduling problem into two: assignment, and scheduling. Then, we can decouple scheduling part of the problem which is the key motivation in implementing logic-based Benders decomposition. Thus, we have a chance to solve much smaller problems and add some cuts which will impose bounds on the objective function, minimization of total tardiness, to obtain the optimal solution by achieving some speed-ups.

Computational experiments will be done to analyze the performance of hybrid model compared to pure MILP and pure CP models for the single-facility scheduling problem with long time horizons.

One of the future research would be updating the time windows of facility at which there is a job overlapping to another facility. For example, job \( j \) is the last job at facility \( i - 1 \) overlapping facility \( i \). We propose shifting \( z_i \) to the finish time of job \( j \). Not only forward shifting but also backward shifting of \( z \) would be studied. Suppose the remaining time slot at the end of the facility \( i \)’s time window is not enough for any of the jobs to be scheduled on facility \( i \), therefore, shifting respective \( z \) (\( z_{i+1} \)) would allow us to gain the lost time slot. Addition of new constraints and new objective function term in the master problem become necessary when shifting is allowed. We would also consider updating \( z_i \) when preemptive scheduling is allowed.
7 APPENDIX

7.1 Inference Duality

In logic-based Benders decomposition, the linear programming dual of the subproblem is generalized to become an inference dual [10]. The inference dual of

\[ \text{[LP-1]} \min z = c^T x \]
\[ \text{s.t.} \quad Ax \geq b \]
\[ x \geq 0 \]

is finding the strongest inequality \( c^T x \) that can be inferred from (41) and (42) [14].

\[ \text{[Inference dual of LP-1]} \max \beta \]
\[ \text{s.t.} \quad \left( \begin{array}{l}
Ax \geq b \\
x \geq 0
\end{array} \right) \Rightarrow c^T x \geq \beta \]

7.2 OPL Studio code for the CP model

As mentioned in Section 3.2 decision variables are defined as interval variables (dojob, actOnRes) and sequence variable (seq). \( M \) is an upper bound on the finish time of interval variables dojob. machjob[1,j].duration corresponds to processing time of job \( j \) (namely, \( p_j \)). segmentLength is the length of each fictitious facility. With \( (i - 1) \times \text{segmentLength} \ldots (i) \times \text{segmentLength} \) we define the time windows for facility \( i \).

dvar interval dojob[j in Jobs] in job[j].release..M size machjob[1,j].duration;
dvar interval actOnRes[i in Machines][j in Jobs] optional
in (i-1)*segmentLength..(i)*segmentLength;
dvar sequence seq[i in Machines] in all(j in Jobs) actOnRes[i][j];

minimize sum (j in Jobs) maxl(endOf(dojob[j]) - job[j].deadline, 0);
subject to {
    forall(j in Jobs)
        alternative(dojob[j], all(i in Machines) actOnRes[i][j]);
    forall(i in Machines)
        noOverlap(seq[i]);
References

[1] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[2] H. Cambazard, Hladík, P. E., Deplanche, N. A. M. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. In *Principles and Practice of Constraint Programming (CP2004)*, *Lecture Notes in Computer Science* 3258, pages 153–167. Springer, 2004.

[3] Y. Chu and Q. Xia. Generating benders cuts for a class of integer programming problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*. *Lecture Notes in Computer Science* 3011, pages 127–141. Springer, 2004.

[4] A. I. Correa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*. *Lecture Notes in Computer Science* 3011, pages 370–378. Springer, 2004.

[5] J. Du and J. Y. T. Leung. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 15(3):483–495, 1990.

[6] S. French. *Sequencing and Scheduling*. John Wiley & Sons, 1982.

[7] A.M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[8] I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:16471660, 2002.

[9] I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.

[10] J. N. Hooker. Logic-based benders decomposition. Technical report, CMU, 1995.

[11] J. N. Hooker. Inference duality as a basis for sensitivity analysis. In *Constraints*, pages 224–236. Springer, 1996.

[12] J. N. Hooker. *Integrated methods for optimization*. Springer, 2007.

[13] J. N. Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55:588–602, 2007.

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

[15] J. N. Hooker, G. Ottosson, E. S. Thorsteinsson, and H. J. Kim. On integrating constraint propagation and linear programming for combinatorial optimization. In *AAAI ’99/IAAI ’99*, pages 136–141, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[16] J. N. Hooker and H. Yan. *Logic circuit verification by Benders decomposition*, pages 267–288. *Principles and Practice of Constraint Programming: The Newport Papers*, MIT Press (Cambridge, MA, 1995), 1995.
[17] V. Jain and I. E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. INFORMS Journal on Computing, 13(4):258–276, 2001.

[18] C. Koukamas. The single-machine total tardiness scheduling problem: Review and extensions. European Journal of Operational Research, In Press, 2009.

[19] C.T. Maravelias and I. E. Grossmann. A hybrid milp/cp decomposition approach for the continuous time scheduling of multipurpose batch plants. Computers and Chemical Engineering, 28:1921–1949, 2004.

[20] M. Pinedo. Scheduling: theory, algorithms, and systems. Prentice Hall, 1995.

[21] C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. OR Spectrum, 24:431–448, 2002.