On hierarchical planning and scheduling in a parallel machine environment

T. Kis[a,*], A. Kovács[a]

[a] *Computer and Automation Institute, Kende str. 13-17, H-1111 Budapest, Hungary*

Abstract

In this paper we propose a branch-and-cut algorithm for solving an integrated production planning and scheduling problem in a parallel machine environment. The planning problem consists of assigning each job to a week over the planning horizon, whereas in the scheduling problem those jobs assigned to a given week have to be scheduled on the parallel machines such that all jobs are finished within the week. We solve this problem in two ways: (1) as a monolithic mathematical program, and (2) using a hierarchical decomposition approach in which only the planning decisions are modeled explicitly, and the existence of a feasible schedule for each week is verified by using cutting planes. Our computational results show that the hierarchical decomposition approach outperforms the monolithic model for this problem.

Keywords: Production planning and scheduling, Mathematical Programming, Cutting planes, Parallel Machine Scheduling.

1. Introduction

Hierarchical production planning and scheduling deals with tactical and operational decisions. The two types of decisions differ in their scope and time horizon [1]. We focus on *planning* on a weekly basis the objective being to determine the most cost effective way of distributing the workload between the weeks, while *scheduling* is concerned with allocating resources to tasks to be performed during the same week. The main advantage of hierarchical planning and scheduling is that at each decision level, only the most relevant information is used. E.g., when taking planning decisions, resource capacities are aggregated and the fine details of dealing with single resources are neglected. In contrast, when solving scheduling problems, only the weekly or daily assignments have to be scheduled [2]. It is often mentioned that these decisions are worth to be separated to ease the work of decision makers at either level. However, the two

*Corresponding Author: Tamás Kis, tel: +36-1-2796156, fax: +36-1-4667503.
Email addresses: tamas.kis@sztaki.hu (T. Kis), akovacs@sztaki.hu (A. Kovács)

types of decisions are strongly related, since both the overloading and the underloading of the weekly production capacities have undesired effects. Namely, if the weekly assignment cannot be met, then the plan has to be reworked. On the other hand, a loose plan may cause unnecessary delays and thus incur penalties which could be avoided by more careful planning. To remedy this situation, *integrated planning and scheduling* has been suggested by various authors [3, 4].

We will study a scheduling problem in a parallel machine environment, where each task has a release time and a due-date, the release time being the first week of the time horizon where the task may be started and the due-date is the week where the task should be completed. Each task has to be assigned to a week and those tasks assigned to a given week must be scheduled on the parallel machines so that the load of every machine is no more than one week. The objective is to minimize the earliness/tardiness penalty costs incurred by completing some of the tasks before or after their due-dates. This setting is suitable to study various problem formulations and to demonstrate that it pays off to pursue a hierarchical decomposition based approach.

While most of the hierarchical approaches for solving hard scheduling problems studied in the literature reduce the problem size by decomposing the problem along the resources, our approach reduces the problem size by decomposing along the types of decisions: the upper level assigns the tasks to weeks, and the lower level schedules the tasks assigned to a given week. Though this is a very natural decomposition approach, the computational advantages are not apparent at once. We will model explicitly only the planning decisions (assignment of tasks to weeks), while the schedulability of those tasks assigned to a given week will be verified by cutting planes. In contrast to most previous approaches, we generate not only infeasibility or "no-good" cuts, but other problem specific cuts as well, and we attempt to generate violated cuts not only when an integer solution is found, but in all search-tree nodes.

After a brief literature review (Section 2), we provide a formal problem statement in Section 3. In Section 4 and Section 5 we propose two alternative formulations: a monolithic mathematical program, and one suitable for decomposition, respectively. To strengthen the second formulation, we derive cutting planes from lower bounds for the bin-packing problems (Section 5.1), and corresponding separation algorithms (Section 5.2). The cutting planes are used in a Branch-and-Cut algorithm (Section 6), whose effectiveness is compared to solving the production planning and scheduling problem as a monolithic mathematical program in Section 7.

2. Literature review

Hierarchical decomposition approaches are applied widely in the field of production planning and scheduling. Classically, three hierarchy levels are distinguished: the *strategic*, the *tactical* (also called the medium-term production planning), and the *operational* (detailed production scheduling) levels [1]. Below we focus on the tactical and operational levels, as well as on their integration. A review of solution approaches applicable on these levels has been presented by

Grossmann et al [5]. The possible ways of integrating the production planning and scheduling are surveyed in [4].

Although the decisions made on the different levels are strongly related, solving these problems in an integrated way is often considered to be computationally intractable. It is therefore typical to apply single- or multi-pass heuristics. In the single-pass case, one fixed upper level plan is unfolded on the lower level, see e.g., [2, 6]. Obviously, a shortcoming of this approach is that bad planning decisions may result in situations where no detailed schedules can meet all production goals. Multi-pass heuristics aim at relieving such situations by iterating between the two levels, and modifying the upper level plan according to the problems identified in the previous iteration [7, 3]. A classical multi-pass approach to integrated job-shop planning and scheduling has been presented by Lasserre [3]. Given the demand, planning determines production and inventory levels for each product and time period, subject to the fixed product sequences determined by the previous scheduling step. On the other hand, scheduling solves the job-shop problem with sequence-independent setup times to minimize makespan, derived from a fixed plan. It is shown that the iteration converges to a local optimum.

In the sequel we focus on exact solution methods that use hierarchical decomposition. For the efficiency of such approaches, the strength of the cuts fed back from the lower to the upper level is crucial. One of the problem models frequently addressed using such methods is the *multi-machine assignment and scheduling problem* (MMASP): a set of tasks, characterized by individual time windows, are to be scheduled on unrelated parallel machines to minimize the total assignment cost. In all of the following papers, the master problem assigns tasks to machines, while a separate subproblem belongs to each machine, sequencing the tasks on that machine. Jain and Grossmann [8] apply a MILP/CP approach, and add an infeasibility or "no-good" cut for the complete set of job scheduled on the machine where infeasibility is detected. Hooker and Ottoson [9] introduce logic-based Benders decomposition, and illustrate the approach on MMASP. The same type of infeasibility cuts is used, though an indication is made that these cuts can be strengthened based on the CP proof of infeasibility. Sadykov and Wolsey [10] compare several monolithic and MIP/CP hybrid decomposition approaches. The new results include a tight MILP formulation. Their decomposed approaches detect infeasibility or "no-good" cuts in internal nodes of the branch-and-bound tree, after a suitable rounding of the LP solutions. Sadykov [10] investigates the solution of the one-machine subproblem of the above multi-machine assignment problem, which corresponds to $1|r_j|\sum w_j U_j$. Two new classes of cuts are introduced for this problem. The first class is infeasibility cuts of low cardinality, which are found by a modified version of Carlier's branch-and-bound algorithm [11]. The second class is a completely different type of cuts based on the edge-finding constraint propagation rule. Bockmayr and Pisaruk [12] investigate a the problem in the general context of generating infeasibility cuts by CP for MILP. The application of the generic idea to MMASP leads to infeasibility cuts. MMASP has been generalized to cumulative resources in [13], and solved by a hybrid MIP/CP approach

following the above decomposition scheme.

MMASP is extended to multi-stage processes in [14]. The same assign/schedule decomposition approach is taken. The main difference due to the multi-stage processes is that the single-machine subproblems are no longer independent, hence, a single subproblem involving all machines and tasks is solved. Infeasibility cuts for a subset of the tasks are generated by solving an optimization version of the subproblem for minimizing the number of late tasks. The cut includes a late job and a no-wait path from the last to the first stage. Note that the cuts are in fact invalid, and may cut off the optimal solution.

A different, multi-product continuous plant scheduling problem with a single processing unit, subject to sequence-dependent setup times, is discussed in [15]. A decomposition approach is proposed, where the upper level sets production levels and inventories for macro time periods. The lower level sequences the production activities subject to the sequence-dependent setup times. If this lower-level problem proves to be infeasible, then integer and logic cuts are fed back to the upper level. Both levels are described by and solved as a MILP.

Guyon et al. [16] propose a decomposition and cut generation approach to an integrated timetabling and job-shop scheduling problem. In this problem, the interruptible tasks are characterized by time windows, durations, and one required skill, whereas employees are described by their set of skills and their potential calendar assignments. The master problem composes a timetable for the employees, while the subproblem checks if a feasible task schedule exists for the given timetable. It is exploited that the subproblem corresponds to a maximum flow problem, and hence, a minimum cut is fed back to the master problem upon infeasibility. An initial set of cuts is generated in a pre-processing step using energetic reasoning for heuristically selected time periods and sets of skills.

Artigues et al. [17] investigate a hybrid decomposition solution approach for an integrated employee timetabling and job-shop scheduling problem. The problem is an extension of the classical job-shop scheduling problem, where beyond machine resources, tasks also require one or more employees, performing specific types of activities. Employees have different sets of skills, and incur different assignment costs, if perform a given type of activity. Each employee is available in a given subset of the shifts. The objective is to minimize the lexicographical combination of the makespan and the assignment cost. A decomposition-based CP formulation is proposed, which performs the rough-cut assignment of tasks to time periods: it determines the shifts in which a task will be (partially) processed.

3. The integrated production planning and scheduling problem

In this section we give a formal definition of the scheduling problem studied in the paper. Suppose that the time horizon is divided into τ equal length periods. The common length of the periods will be denoted by P, and let $T = \{1, \ldots, \tau\}$ be the set of all time periods. There is a set of jobs N to be scheduled on a set of parallel identical machines M. Each job $j \in N$ has a

release date r_j and a due-date d_j, both expressed in terms of periods. Namely, $r_j \in T$ is the earliest time period where the job may be processed, and $d_j \in T$ is the period when the job should be finished without paying a penalty. On the one hand, if job j is finished early in some period $C_j < d_j$, the penalty incurred is $(d_j - C_j)e_j$. In contrast, if it is finished late in some period $C_j > d_j$, the penalty to be payed is $(C_j - d_j)\ell_j$. The processing time of job j is p_j on all machines. Each job has to be processed on exactly one machine, and the preemption of jobs is not allowed. No machine may process more than one jobs at a time. Furthermore, we make the following assumptions about jobs.

Assumption 1. *The jobs are shorter than P, the common length of the periods.*

Assumption 2. *Each job has to be processed in a single period, i.e., no job may cross the boundary of two consecutive periods.*

These two assumptions are met in a number of practical applications. For instance, if periods represent weeks of 5 working days each, then the first assumption says that no job takes more than 5 working days, and the second assumption means that no job may be left unfinished over the weekend.

The ultimate goal is to assign jobs to periods and to machines in such a manner that the total processing time of those jobs assigned to the same period and to the same machine is at most P, and the total penalty incurred by completing some of the jobs early or late is minimized.

Note that this problem is NP-hard, because it contains the NP-hard bin packing problem (see, e.g., [18]): when the tasks have the same release times and due-dates, then there exists a schedule with zero cost if and only if the corresponding bin packing problem with items of size p_j and bin capacity P has a solution with at most $|M|$ bins. In the next two sections we present two alternative approaches for solving the integrated planning and scheduling problem just defined.

4. Solution as a monolithic integer program

In this section we define a mathematical program for solving our combined planning and scheduling problem. The decision variables are x^j_{kt}, for $j \in N$, $k \in M$ and $t \in T$, representing the assignment of jobs to machines and time periods. For each job j, precisely one of the x^j_{kt}, $k \in M$, $t \in T$, takes the value 1, and all other variables belonging to the same job take value 0. Therefore, our

planning problem can be formulated as follows:

$$\min_x \sum_{j\in N}\sum_{k\in M}\sum_{t\in T} w_t^j x_{kt}^j \tag{1}$$

s.t.

$$\sum_{k\in M}\sum_{t\in T} x_{kt}^j = 1, \quad \forall\ j \in N, \tag{2}$$

$$\sum_{j\in N} p_j x_{kt}^j \le P, \quad \forall\ k \in M,\ t \in T, \tag{3}$$

$$x_{kt}^j = 0, \quad \forall\ j \in N, k \in M, t < r_j \tag{4}$$

$$x_{kt}^j \in \{0,1\}, \quad \forall\ j \in N, k \in M, t \in T. \tag{5}$$

The weights in the objective function are given by

$$w_t^j = \max\{d_j - t, 0\}e_j + \max\{t - d_j, 0\}\ell_j, \tag{6}$$

which expresses that if job j finishes early in period $t < d_j$, i.e., $x_{kt}^j = 1$, the penalty is $(d_j - t)e_j$, whereas if it finishes late in period $t > d_j$, i.e., $x_{kt}^j = 1$, the penalty is $(t - d_j)\ell_j$. Constraints (2) ensure that each job is assigned to precisely one machine and to one period. The inequalities (3) are the capacity constraints for each machine and each period. The equations (4) set all the x_{kt}^j variables to 0 for all periods t before the release date of job j. Notice that these variables can be eliminated in actual computations.

The feasible solutions X of the above mathematical program are binary vectors satisfying all of the constraints. Let $\mathrm{conv}(X)$ denote the convex hull of these vectors. It is a convex polytope and the optimal solutions correspond to a subset of its vertices. Since our planning problem is NP-hard, $\mathrm{conv}(X)$ is unlikely to admit a representation with a polynomial number of inequalities in the size of the problem data. Therefore, we can solve it by, e.g., branch-and-cut type methods, which combine branch-and-bound and the generation of valid inequalities violated by fractional solutions in search-tree nodes.

To generate valid inequalities, observe that the formulation contains $|M||T|$ knapsack sets of the form $Y = \{y \in \mathbb{R}^n \mid \sum_{i=1}^n a_i y_i \le P\}$. There are many classes of valid inequalities for Y, see e.g., [19, 20], and state-of-the-art integer programming solvers contain the most effective ones. In addition, Gomory's mixed integer cuts can always be generated to cut off fractional solutions.

5. Solution as a two-level planning and scheduling problem

Our second formulation is more compact than the first one as the decision variables do not represent explicitly the assignment of jobs to machines. Namely, the decision variables are z_t^j, where $z_t^j = 1$ if and only if job j is assigned to period t. Clearly, for each job j, precisely one of the variables z_t^j, $t \in T$, takes value 1, all other variables in this set take value 0.

$$\min_{z} \sum_{j \in N} \sum_{t \in T} w_t^j z_t^j \tag{7}$$

s.t.

$$\sum_{t \in T} z_t^j = 1, \quad \forall\, j \in N, \tag{8}$$

$$z \in B_t, \quad \forall\, t \in T, \tag{9}$$

$$z_t^j = 0, \quad \forall\, j \in N, t < r_j, \tag{10}$$

$$z_t^j \in \{0,1\}, \quad \forall\, j \in N, t \in T. \tag{11}$$

The weights w_t^j in the objective function are defined by formula 6. The constraints (8) ensure that each job is assigned to precisely one time period. The sets B_t in the set of constraints (9) consists of those binary vectors that satisfy the following condition: $\bar{z} \in B_t$ if and only if the set of jobs $\{j \in N \mid \bar{z}_t^j = 1\}$ can be distributed on $|M|$ parallel machines such that the total processing time of those jobs assigned to any of the machines is not more than P.

The condition $z \in B_t$ is equivalent to a bin-packing problem. Recall that in the bin-packing problem there is given a set of n "items" with sizes $s_1, \ldots, s_n$, and a supply of identical containers of capacity C, and the objective is to determine the minimum number of containers to pack all the items [18]. To simplify notation, let $\tilde{p}_j = p_j/P$. Then each bin is of size 1, and the item sizes are between 0 and 1. Deciding whether all jobs assigned to time period t can be distributed among $|M|$ identical, parallel machines such that no machine receives more work than P is equivalent to verifying whether the minimum number of bins needed to pack all the items is not more than $|M|$. We will derive valid inequalities for bin-packing problems and apply them to describe partially the convex hull of B_t.

5.1. Valid inequalities from bin-packing problems

In this section we derive various classes of valid inequalities from lower bounds for the bin-packing problem with a set of items H and item sizes $\tilde{p}_j \in (0,1]$.

L_1 inequalities. There are several lower bounds in the literature for the minimum number of bins to fit all the items in H. For instance, the well-known L_1 lower bound for a set of items H is $L_1(H) := \lceil \sum_{j \in H} \tilde{p}_j \rceil$ [21]. To turn it into a valid inequality, suppose we have a set of jobs H to be defined later. Since the set of jobs H has to be scheduled on $|M|$ identical parallel machines, the inequality

$$\sum_{j \in H} \tilde{p}_j z_t^j \leq |M| \tag{12}$$

is valid for B_t.

L_2 inequalities. Now consider the lower bound L_2 of [21]. For a set of items H, the L_2 lower bound is

$$L_2(H) := \max_{\varepsilon \in [0,\frac{1}{2}]} |\{j \in H \mid \tilde{p}_j > 1-\varepsilon\}| + L_1(\{j \in H \mid \varepsilon \leq \tilde{p}_j \leq 1-\varepsilon\}).$$

Therefore, for any $\varepsilon \in [0, \frac{1}{2}]$ the inequality

$$\left(\sum_{j \in H: \tilde{p}_j > 1-\varepsilon} z_t^j\right) + \left(\sum_{j \in H: \varepsilon \leq \tilde{p}_j \leq 1-\varepsilon} \tilde{p}_j z_t^j\right) \leq |M| \tag{13}$$

is valid for B_t, Clearly, it is enough to consider ε from the set

$$P_{\frac{1}{2}}(H) = [0, \frac{1}{2}] \ \cap \ (\{\tilde{p}_j \mid j \in H\} \cup \{1-\tilde{p}_j \mid j \in H\}). \tag{14}$$

Fekete&Schepers inequalities. The third class of valid inequalities is derived from the lower bound $L_*^{(p)}$ of [18]. Define for any $k \in \mathbb{N}$

$$L_2^{(k)}(H) := \max_{\varepsilon \in [0,\frac{1}{2}]} L_1(U^\varepsilon u^{(k)}(H)),$$

where $u^{(k)} : [0,1] \to [0,1]$ with

$$u^{(k)}(x) := \begin{cases} x & \text{for } x(k+1) \in \mathbb{Z} \\ \lfloor (k+1)x \rfloor \frac{1}{k}, & \text{else} \end{cases}$$

and $U^\varepsilon : [0,1] \to [0,1]$ with

$$U^\varepsilon(x) := \begin{cases} 1, & \text{for } x > 1-\varepsilon \\ x, & \text{for } \varepsilon \leq x \leq 1-\varepsilon \\ 0, & \text{for } x < \varepsilon \end{cases}$$

For $p \geq 2$, the lower bound $L_*^{(p)}$ for the set of items H is

$$L_*^{(p)}(H) := \max\{L_2(H), \max_{k=2,\ldots,p} L_2^{(k)}(H)\}.$$

The validity of this lower bound for the bin-packing problem is verified in [18]. We can turn this into a family of valid inequalities for B_t as follows. Using the definition of $L_2^{(k)}$, for fixed $\varepsilon \in P_{\frac{1}{2}}(H)$ and $k \geq 2$, the inequality

$$\left(\sum_{j \in H: \tilde{p}_j > 1-\varepsilon} z_t^j\right) + \left(\sum_{j \in H: \varepsilon \leq \tilde{p}_j \leq 1-\varepsilon} U^\varepsilon u^{(k)}(\tilde{p}_j) z_t^j\right) \leq |M| \tag{15}$$

is valid for B_t.

Infeasibility or "no-good" cuts. Finally, if a set of items H cannot be scheduled in the same time period t, e.g., if they violate the cuts (13) or (15), then the *infeasibility* cut

$$\sum_{j \in H^{ext}} z_j^t \leq |H| - 1, \tag{16}$$

is violated as well, where H^{ext} is the set $H \cup \{j \in N \mid p_j \geq \max_{k \in H} p_k\}$.

5.2. Separation algorithms

In order to find violated inequalities (cuts) in classes (12), (13), and (15), we have to define the set H.

As for (12), for each period t we define the set $H_t := \{j \in N \mid t \geq r_j\}$, and add the corresponding L_1 inequality to the initial formulation. Additional inequalities of this type are not separated during the branch-and-cut algorithm, because those would be dominated by the L_1 inequality for H_t.

Concerning (13), we add the L_2 cuts using H_t and $\varepsilon = 0.5$ to the initial formulation. To separate additional cuts of type L_2 in the course of the branch-and-cut search, firstly we define for each period t the set $\bar{H}_t = \{j \in N \mid \bar{z}_j^t > 0.5\}$, where $\bar{z}$ is the current optimal solution of the LP relaxation. Then we try to find heuristically a minimal subset of $\bar{H}_t$ that violates a constraint from (13). Note that it may happen that $\cup_{t=1}^{T} \bar{H}_t \neq N$, i.e., some fractionally assigned jobs may not belong to any of the sets $\bar{H}_t$. However, as our computational experiments show, the cuts generated are quite effective in solving the hierarchical scheduling problem. The pseudo-code of the separation algorithm for finding violated L_2 cuts is given below:

1. **for** $\varepsilon \in P_{\frac{1}{2}}(\bar{H}_t)$ **loop**
2. **if** the L_2 inequality with ε and $\bar{H}_t$ is violated **then**
3. Determine the minimal subset $\bar{H}'_t$ of largest elements in $\bar{H}_t$ such that the L_2 inequality is violated for ε and $\bar{H}'_t$.
4. add the L_2 cut (13) as well as the infeasibility cut (16) with respect to $\bar{H}'_t$ and ε to the LP relaxation.
5. **end-if**
6. **end-loop**

Finally, the cuts (15) are generated only in the search-tree nodes (including the root), in the same way as the L_2 cuts for $k = 2, \ldots, 10$.

Prior to trying to separate a violated inequality, we try to solve the scheduling problem determined by the set of jobs $\bar{H}_t$ by using a simple heuristic, such as first-fit decreasing for bin-packing. If the heuristic finds a feasible schedule using at most $|M|$ machines, then none of the cuts may be violated. Otherwise, we try to find violated (13) and (15) cuts. If no violated cut in these classes is found, we solve the NP-complete decision problem whether the job in $\bar{H}_t$ can be scheduled on $|M|$ parallel machines such that no machine receives a workload of more than P, which is equivalent to a bin-packing problem. This bin-packing problem can be solved in a number of ways [21], and if the result

is that more than $|M|$ machines are needed to schedule all the jobs in $\bar{H}_t$, then we add the infeasibility cut (16) to the LP relaxation. In our implementation, we checked feasibility by solving the following mathematical program (without any objective function) by a MIP solver:

$$\sum_{k \in M} y_k^j = 1, \quad \forall\, j \in \bar{H}_t, \tag{17}$$

$$\sum_{j \in \bar{H}_t} p_j y_k^j \leq P, \quad \forall\, k \in M, \tag{18}$$

$$y_k^j = 0, \quad \forall\, j \in \bar{H}_t, k \in \{k_j + 1, \ldots, |M|\}, \tag{19}$$

$$\left(\sum_{j \in \bar{H}_t : p_j > \frac{P}{2}} y_k^j \right) + \left(\sum_{j \in \bar{H}_t : p_j = \frac{P}{2}} \frac{y_k^j}{2} \right) \leq 1, \quad \forall\, k \in M, \tag{20}$$

$$y_k^j \in \{0, 1\}, \quad \forall\, j \in \bar{H}_t, k \in M. \tag{21}$$

The constrains (17) ensure that each job is assigned to exactly one machine. Inequalities (18) enforce the capacity constraints on the machines. The symbol k_j in constraints (19) stands for the position of job j within an arbitrary ordering of the jobs in the set $\bar{H}_t$, and therefore, the constraints break the symmetries of machine assignment. Finally, line (18) corresponds to the L_2 inequalities for the bin packing subproblem with items $\bar{H}_t$ and $\varepsilon = \frac{1}{2}$.

6. The branch-and-cut algorithm

Branch-and-cut is an extension of branch-and-bound where valid inequalities (cuts) violated in a search-tree node are sought and added to the LP relaxation corresponding to the node. Each node in the course of our branch-and-cut algorithm is processed as follows:

1. Let $\bar{z}$ be the solution of the LP relaxation corresponding to the node.
2. **if** $\bar{z}$ is integral **then**
3. **if** $\bar{z} \in B_t$ for each period t **then**
4. fathom the node
5. **else if** there exists a violated (13) or (15) cut **then**
6. add the violated cut along with the corresponding infeasibility cut to the LP relaxation and reoptimize
7. **else** add the infeasibility cut with respect to $\bar{H}_t = \{j \in N \mid \bar{z}_j^t = 1\}$ to the LP relaxation and reoptimize
8. **end-if**
9. **else** apply the separation algorithms to the LP-relaxation, and if any cuts are added to the LP-relaxation, reoptimize
10. **end-if**

Several comments are in order. In step 3, $\bar{z} \in B_t$ is verified in several phases. Firstly, the bin-packing problem with items $\bar{H}_t = \{j \in N \mid \bar{z}_j = 1\}$ is solved heuristically (using the first-fit decreasing algorithm). If at most $|M|$ bins (machines) suffice, then the jobs assigned to period t can be scheduled on $|M|$ parallel machines within P time units. Otherwise, the separation algorithms are applied using the set $\bar{H}_t$. Notice that step 7 ensures that infeasible integral solutions are always cut off by a valid inequality and the node is reoptimized afterwards. In step 4, fathoming a node consists of dropping the node and the entire subtree below it. Moreover, if the solution represented by $\bar{z}$ is better than the best solution found so far, then $\bar{z}$ is recored as the best solution.

In step 9, the set $\bar{H}_t$ is defined with respect to the fractional solution $\bar{z}$ as in Section 5.2, and it may well be the case that no violated cuts are found.

After reoptimization the same node is processed again. This is repeated until the node is fathomed in step 4, or fractional solutions are obtained during at most 5 consecutive reoptimization steps. In the latter case, some variable z_j^t with $0 < \bar{z}_j^t < 1$ is selected for branching and it is set to 0 and 1 in the two descendants of the node, respectively.

Finally, we mention that we implemented a quick constructive heuristic, and ran it before the branch-and-cut algorithm to ensure that the solver always finds a feasible solution. The heuristic sorts the jobs by non-increasing lateness penalty ℓ_j, and assigns them one-by-one to a time period and machine that has enough free capacity to process job j and which incurs minimal penalty. This initial solution is improved using a hill climbing search with two types of moves: (i) reassign a job to a different period, and (ii) interchange two jobs belonging to different periods. In either case the resulting schedule has to be feasible, and the algorithm chooses a move which strictly decreases the objective function. This is repeated until a local optimum is reached, i.e., the schedule cannot be further improved by any of these moves.

7. Computational results

In this section we compare the computational performance of the proposed hierarchical decomposition approach with cutting planes to the results achieved by the monolithic model. Three different versions of the hierarchical solver has been tested, each exploiting different subsets of the cuts investigated. In particular, the following versions of the solver has been experimented with:

- *Hierarchical A*, a hierarchical solver using all the investigated classes of cuts (infeasibility, L_2, and Fekete&Schepers);
- *Hierarchical B*, a hierarchical solver exploiting infeasibility and L_2 cuts only;
- *Hierarchical C*, a hierarchical solver making use of infeasibility cuts only;
- *Monolithic*, the baseline monolithic model;

We note that the initial MILP formulation contained the L_1 inequalities and the L_2 inequalities for H_t and $\varepsilon = 0.5$ in all versions. All the models and algorithms have been implemented in Xpress-MP using the Mosel programming language. The experiments were run on a 1.86 GHz Intel Xeon computer with 2 GB of RAM under Windows Server 2003.

Two sets of problem instances were considered that differ in the average p_j/P value, and which we call the *large-task* and the *small-task* instances, respectively. For the large-task problems, the time period length P was fixed to 100, and processing times p_j were taken uniformly at random from the set $\{35, \ldots, 100\}$ ($p_j \in U[35, 100]$). To generate time windows for the tasks, two integers, α and β were taken from $U[1, \tau_0]$, and we set $r_j = \min(\alpha, \beta)$ and $d_j = \max(\alpha, \beta)$. Here, τ_0 is the nominal length of the time horizon, which characterizes the variance of the release and due-dates. To ensure that all instances are solvable, a longer time horizon with $\tau = \tau_0 + \lceil \frac{2 \cdot \sum_{j \in N} p_j}{P \cdot |M|} \rceil$ was used in the models. Earliness and tardiness penalties e_j and ℓ_j were randomized from $U[1, 10]$. The size of problem instances was controlled by varying the number of tasks $|N|$ between 20 and 80 with increments of 10, the number of machines $|M|$ and also the nominal number of periods τ_0 between 2 and 10 with increments of 2. Ten instances were generated for every combination of these three parameters, resulting in 1750 instances in this set.

The small-task instances were generated in the same way, but using $P = 30$ and taking p_j from $U[1, 10]$. Since these instances were easier to solve, in this case $|N|$ varied between 50 and 200, with increments of 25. This set contained 1750 further instances. The time limit was set to 120 seconds per instance.

The results are presented in Tables 1-4, separately for the two instance sets and for the different views of the results. Each row of the tables contain combined results for the same value of $|N|$ and τ_0 (Tables 1 and 2), or $|N|$ and $|M|$ (Tables 3 and 4). Columns *Opt* display the number of instances solved to proven optimality by the different solver versions; columns *Gap* show the average gap in percent between the upper and lower bounds, computed as $(UB-LB)/LB*100$; columns *Time* contain the average computation times, or 120 where the time limit was hit; finally, column *Cuts* displays the average number of cuts generated during the solution of an instance, including all infeasibility, L_2, and Fekete&Schepers cuts.

The results show that instances with high $|N|$, low $|M|$, and low τ_0 were the most challenging for all variants of the solver. With many machines (high $|M|$) the bin packing problems corresponding to individual time periods were easier to solve. The case was similar with higher variance of the deadlines (high τ_0). The results also indicate that large-task instances were significantly more difficult to solve (solvable to optimality with up to 20 tasks only) than small-task instances (up to 100 tasks).

However, a closer look to the results show essential differences on the two instance sets. On large-task instances, solver *Hierarchical A* outperformed all other solver versions. Both L_2 and Fekete&Schepers cuts added significant strength to the solver. The hierarchical solver operating with infeasibility cuts

only performed even weaker than the monolithic approach. There were considerable gaps between the upper and lower bound found, especially for the monolithic results, which in for certain combinations of parameters (e.g., $|N| = 80$ and $|M| = 10$) cases computed solution with cost two or three times higher than the lower bound.

In contrast, on the small-task instances, all the three hierarchical versions produced the same results, with a minor shift in favor of the simpler approaches due to the lack of complicated cut generation procedures (1579 vs. 1588 optimal solutions). Hence, the complicated L_2 and Fekete&Schepers cuts did not add significant strength to the hierarchical approach. Still, all hierarchical solver versions outperformed the monolithic approach. Overall, these instances were much easier to solve, which is also shown in the insignificant gap between the upper and lower bounds.

8. Conclusions

In this paper we have supported by computational evidence the common wisdom of working with the right level of details at the various levels of the production planning and scheduling process. Albeit the parallel machine environment considered in this paper may have some practical applications, we plan to extend the model with machine depended processing times and precedence constraints.

Acknowledgements

The work reported here has been supported by OTKA grant K76810, and by the research grant "Digital, real-time enterprises and networks", OMFB-01638/2009. A. Kovács acknowledges the support of the János Bolyai scholarship No. BO/00138/07.

References

[1] Vollmann TE, Berry WL, Whybark DC. Manufacturing Planning and Control Systems. McGraw-Hill; 1997.

[2] Fontan G, Merce C, Hennet JC, Lasserre J. Hierarchical scheduling for decision support. Journal of Intelligent Manufacturing 2005;16:235–42.

[3] Lasserre JB. An integrated model for job-shop planning and scheduling. Management Science 1992;38(8):1201–11.

[4] Maravelias CT, Sung C. Integration of production planning and scheduling: Overview, challenges and opportunities. Computers & Chemical Engineering 2009;33(12):1919–30.

[5] Grossmann IE, van der Heever SA, Harjunkoski I. Discrete optimization methods and their role in the integration of planning and scheduling. AIChE Symposium Series 2002;326:150–68.

[6] Roe B, Papageorgiou LG, Shah N. A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. Computers & Chemical Engineering 2005;29:1277–91.

[7] Das BP, Rickard JG, Shah N, Macchietto S. An investigation on integration of aggregate production planning, master production scheduling and short-term production scheduling of batch process operations through a common data model. Computers & Chemical Engineering 2000;24:1625–31.

[8] Jain V, Grossmann IE. Algorithms for hybrid MILP/CLP models for a class of optimization problems. INFORMS Journal on Computing 2001;13:258–76.

[9] Hooker JN, Ottosson G. Logic-based benders decomposition. Mathematical Programming, Ser A 2003;96(1):33–60.

[10] Sadykov R, Wolsey LA. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. INFORMS Journal on Computing 2006;18(2):209–17.

[11] Carlier J. The one-machine sequencing problem. European Journal of Operational Research 1982;11(1):42–7.

[12] Bockmayr A, Pisaruk N. Detecting infeasibility and generating cuts for mixed integer programming using constraint programming. Computers & Operations Research 2006;33(10):2777–86.

[13] Chu Y, Xia Q. A hybrid algorithm for a class of resource constrained scheduling problems. In: Proceedings of the 2nd International Conference CPAIOR'2005 (Springer LNCS 3524). 2005, p. 110–24.

[14] Harjunkoski I, Grossmann IE. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers & Chemical Engineering 2002;26(11):1533–52.

[15] Erdirik-Dogan M, Grossmann IE. Simultaneous planning and scheduling of single-stage multi-product continuous plants with parallel lines. Computers & Chemical Engineering 2008;32(11):2664–83.

[16] Guyon O, Lemaire P, Pinson E, Rivreau D. Cut generation for an integrated employee timetabling and production scheduling problem. European Journal of Operational Research 2010;201:557–67.

[17] Artigues C, Gendreau M, Rousseau LM, Vergnaud A. Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. Computers & Operations Research 2009;36(8):2330–40.

[18] Fekete S, Schepers J. New classes of fast lower bounds for bin packing problems. Mathematical Programming, Ser A 2001;91:11–31.

[19] Balas E, Zemel E. Facets of the knapsack polytope from minimal covers. SIAM Journal on Applied Mathematics 1978;34:119–48.

[20] Gu Z, Nemhauser GL, Savelsbergh MWP. Sequence independent lifting in mixed integer programming. Journal of Combinatorial Optimization 2000;4:109–29.

[21] Martello S, Toth P. Lower bounds and reduction procedures for the bin-packing problem. Dicrete Applied Mathematics 1990;28:59–70.

		Hierarchical A			Hierarchical B			Hierarchical C			Monolithic	
$\|N\|$	τ_0	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap
20	2	49	0.02	1946	48	0.15	1057	45	0.49	565	50	0.00
	4	50	0.00	599	49	0.16	397	49	0.33	190	50	0.00
	6	50	0.00	9	50	0.00	10	50	0.00	8	50	0.00
	8	50	0.00	7	50	0.00	8	50	0.00	6	50	0.00
	10	50	0.00	1	50	0.00	1	50	0.00	1	50	0.00
30	2	32	2.86	16488	26	5.24	4682	20	9.52	1534	47	0.44
	4	44	1.05	5248	41	2.00	3101	37	4.63	975	50	0.00
	6	46	0.56	3112	46	0.86	1800	40	2.06	688	50	0.00
	8	50	0.00	453	50	0.00	469	49	0.12	329	50	0.00
	10	50	0.00	401	50	0.00	289	49	0.20	223	50	0.00
40	2	16	9.16	17128	7	13.00	4936	5	19.29	1671	34	1.56
	4	29	6.07	10662	24	8.70	3461	19	13.69	1319	46	0.28
	6	36	4.00	6418	35	5.79	2253	30	8.09	973	49	0.40
	8	42	2.73	3771	41	3.54	1628	39	4.67	674	49	0.25
	10	44	1.16	2632	43	1.80	1416	42	2.47	561	49	0.01
50	2	2	17.62	20012	0	21.98	4935	0	28.85	1585	4	22.91
	4	18	12.12	13460	10	18.11	3820	9	25.86	1338	5	45.96
	6	24	10.68	10733	22	15.45	3370	17	22.63	1260	6	88.63
	8	34	5.44	6338	32	7.69	2379	30	10.98	963	20	37.14
	10	38	4.99	4092	37	5.94	1555	34	7.84	761	21	26.01
60	2	1	20.23	17536	1	25.97	3961	1	28.33	1392	0	33.76
	4	3	20.19	15483	2	28.03	3639	2	39.28	1331	0	53.41
	6	20	13.19	10591	13	18.28	3596	8	30.61	1367	0	90.90
	8	24	11.18	8399	22	14.78	2917	17	20.84	1255	2	118.32
	10	27	8.89	7406	25	11.72	2931	22	17.97	1052	8	80.39
70	2	0	23.81	14214	0	26.21	3466	0	26.59	1245	0	31.07
	4	1	22.15	13031	1	28.99	3333	1	32.83	1273	0	53.93
	6	8	23.72	11657	3	33.57	3086	3	44.04	1251	0	87.23
	8	15	19.05	10057	7	32.27	3257	7	43.49	1259	0	165.36
	10	23	14.02	7456	20	20.53	2668	18	24.67	1139	1	104.83
80	2	0	21.38	13318	0	23.61	3046	0	24.18	1155	0	27.22
	4	3	28.70	10591	1	33.08	2742	0	35.32	1067	0	55.86
	6	1	28.13	10619	0	37.27	2767	0	47.19	1069	0	75.33
	8	5	22.99	10112	3	33.31	2816	3	47.60	1098	0	140.91
	10	16	18.03	7405	13	26.71	2442	11	35.48	1037	0	161.19
$\sum$		901	10.69	8325	822	14.42	2521	757	18.86	960	791	42.95

Table 1: Results on *large-task* instances by number of tasks and time periods.

		Hierarchical A			Hierarchical B			Hierarchical C			Monolithic	
$\|N\|$	τ_0	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap
50	2	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	4	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	6	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	8	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
75	2	50	0.00	9	50	0.00	13	50	0.00	15	45	0.09
	4	50	0.00	4	50	0.00	4	50	0.00	6	49	0.01
	6	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	8	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
100	2	49	0.00	72	49	0.01	72	49	0.00	60	39	0.06
	4	48	0.02	130	49	0.01	126	49	0.01	104	40	0.13
	6	50	0.00	3	50	0.00	7	50	0.00	4	44	0.12
	8	50	0.00	2	50	0.00	2	50	0.00	2	47	0.08
	10	50	0.00	1	50	0.00	1	50	0.00	1	48	0.10
125	2	44	0.18	167	45	0.17	151	44	0.18	153	29	0.19
	4	43	0.08	182	44	0.06	187	43	0.06	163	33	0.25
	6	49	0.01	80	49	0.01	75	49	0.01	58	40	0.22
	8	50	0.00	26	50	0.00	26	49	0.01	31	40	0.27
	10	50	0.00	19	50	0.00	19	50	0.00	11	43	0.28
150	2	39	0.10	199	39	0.12	189	40	0.10	188	18	0.58
	4	43	0.11	157	45	0.08	150	44	0.07	150	28	0.47
	6	44	0.06	142	44	0.04	142	44	0.04	151	33	0.36
	8	42	0.13	154	43	0.13	152	45	0.09	144	36	0.60
	10	45	0.06	122	46	0.05	121	46	0.05	123	40	0.47
175	2	34	1.06	159	34	1.04	178	34	1.07	169	5	1.43
	4	36	0.63	135	36	0.58	145	37	0.58	149	19	1.22
	6	40	0.12	167	40	0.15	175	40	0.13	170	26	0.64
	8	42	0.09	148	43	0.13	144	41	0.14	151	30	1.01
	10	42	0.11	159	44	0.07	172	43	0.08	162	35	0.82
200	2	34	0.20	155	33	0.18	151	32	0.19	164	2	1.89
	4	33	1.23	120	33	1.17	134	33	1.19	136	15	1.91
	6	40	0.20	113	40	0.19	145	40	0.15	131	21	1.79
	8	41	0.13	146	41	0.10	151	41	0.13	141	26	3.17
	10	41	0.18	120	41	0.15	122	43	0.15	114	29	1.41
$\sum$		1579	0.13	83	1588	0.13	84	1586	0.13	81	1260	0.56

Table 2: Results on *small-task* instances by number of tasks and time periods.

		Hierarchical A			Hierarchical B			Hierarchical C			Monolithic	
$\|N\|$	$\|M\|$	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap
20	2	49	0.02	2373	47	0.31	1262	45	0.72	603	50	0.00
	4	50	0.00	184	50	0.00	199	49	0.10	156	50	0.00
	6	50	0.00	5	50	0.00	8	50	0.00	6	50	0.00
	8	50	0.00	1	50	0.00	3	50	0.00	3	50	0.00
	10	50	0.00	0	50	0.00	1	50	0.00	1	50	0.00
30	2	33	2.55	11157	31	3.35	6272	23	6.07	2197	50	0.00
	4	44	1.25	8599	39	2.61	2594	35	4.97	968	50	0.00
	6	45	0.66	5772	43	2.13	1216	42	3.70	380	50	0.00
	8	50	0.00	115	50	0.00	192	47	0.76	170	49	0.13
	10	50	0.00	59	50	0.00	67	48	1.02	35	48	0.31
40	2	9	11.86	15891	7	14.70	6441	5	17.19	2428	46	0.02
	4	30	4.59	11971	26	6.94	3591	21	10.05	1469	47	0.04
	6	38	3.13	7804	34	4.94	2384	31	9.36	782	47	0.13
	8	44	1.09	3107	41	2.50	863	37	5.03	392	46	0.55
	10	46	2.43	1838	42	3.76	416	41	6.58	127	41	1.76
50	2	4	14.94	12897	4	16.85	5201	4	18.44	2173	0	29.74
	4	10	16.96	18126	8	21.95	5022	5	28.13	1808	2	44.02
	6	26	9.77	12012	22	14.70	3106	20	22.06	1090	13	38.46
	8	37	4.78	6778	32	7.22	1663	28	13.29	602	18	63.78
	10	39	4.41	4823	35	8.44	1067	33	14.24	235	23	44.66
60	2	1	18.41	10073	1	19.77	4452	1	20.86	1941	0	29.35
	4	1	22.04	16927	0	27.26	4875	0	32.73	1738	0	55.64
	6	13	14.34	17710	8	21.41	4622	4	34.14	1641	0	82.47
	8	29	11.93	7782	26	16.30	1980	20	24.71	809	2	94.59
	10	31	6.98	6923	28	14.05	1116	25	24.59	268	8	114.73
70	2	0	19.94	8463	0	20.71	4070	0	21.82	1876	0	30.92
	4	1	27.48	14287	1	33.61	4137	0	36.63	1567	0	54.03
	6	2	25.34	15875	1	33.97	3824	0	41.50	1442	0	74.18
	8	18	13.72	11793	13	20.85	2743	13	26.86	959	0	114.92
	10	26	16.28	5997	16	32.42	1035	16	44.81	322	1	168.38
80	2	0	19.98	6413	0	20.92	3064	0	21.60	1539	0	30.07
	4	0	27.36	12296	0	31.45	3274	0	33.68	1385	0	55.91
	6	0	27.28	13620	0	34.01	3554	0	42.87	1304	0	67.98
	8	7	25.23	11608	5	35.34	2739	4	47.62	904	0	126.63
	10	18	19.39	8109	12	32.27	1183	10	43.99	293	0	179.91
$\sum$		901	10.69	8325	822	14.42	2521	757	18.86	960	791	42.95

Table 3: Results on *large-task* instances by number of tasks and machines.

		Hierarchical A			Hierarchical B			Hierarchical C			Monolithic	
$\|N\|$	$\|M\|$	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap	Cuts	Opt	Gap
50	2	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	4	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	6	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	8	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
75	2	50	0.00	14	50	0.00	17	50	0.00	21	47	0.02
	4	50	0.00	0	50	0.00	0	50	0.00	0	48	0.04
	6	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	8	50	0.00	0	50	0.00	0	50	0.00	0	49	0.05
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
100	2	47	0.03	208	48	0.02	207	48	0.02	171	22	0.43
	4	50	0.00	0	50	0.00	0	50	0.00	0	46	0.05
	6	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	8	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
125	2	38	0.10	470	40	0.07	451	37	0.08	411	4	0.89
	4	49	0.01	3	49	0.01	6	49	0.01	6	36	0.21
	6	49	0.16	0	49	0.16	0	49	0.16	0	47	0.07
	8	50	0.00	0	50	0.00	0	50	0.00	0	48	0.03
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
150	2	14	0.45	761	18	0.42	739	20	0.35	740	0	1.48
	4	49	0.01	13	49	0.00	14	49	0.00	16	23	0.65
	6	50	0.00	0	50	0.00	0	50	0.00	1	38	0.19
	8	50	0.00	0	50	0.00	0	50	0.00	0	44	0.16
	10	50	0.00	0	50	0.00	0	50	0.00	0	50	0.00
175	2	4	0.54	719	7	0.54	767	5	0.51	750	0	1.52
	4	44	0.07	47	44	0.04	46	44	0.09	48	7	1.87
	6	47	0.48	1	47	0.48	1	47	0.48	2	26	1.16
	8	49	0.91	0	49	0.91	0	49	0.91	0	40	0.44
	10	50	0.00	0	50	0.00	0	50	0.00	0	42	0.13
200	2	2	0.90	573	2	0.77	621	4	0.77	613	0	2.01
	4	40	0.97	73	39	0.94	73	39	0.95	63	0	2.94
	6	48	0.01	8	48	0.01	9	47	0.02	10	19	4.11
	8	49	0.07	0	49	0.07	0	49	0.07	0	34	0.84
	10	50	0.00	0	50	0.00	0	50	0.00	0	40	0.27
$\sum$		1579	0.13	83	1588	0.13	84	1586	0.13	81	1260	0.56

Table 4: Results on *small-task* instances by number of tasks and machines.