A cutting plane approach for integrated planning and scheduling

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 in a parallel machine environment 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. The two approaches are compared with extensive computational testing.

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 jobs 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 types of decisions are strongly related, since both the overloading and the underloading of the weekly production capacities have undesired effects. Namely, if

*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)

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 incurs 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 job has a release time and a due-date, the release time being the first week of the time horizon where the job may be started and the due-date is the week where the job should be completed. Each job has to be assigned to a week and those jobs 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 jobs before or after their due-dates. Albeit this setting is a simplification of real-world planning and scheduling problems, where there may be additional constraints on feasible solutions, the decomposition approach proposed in this paper may be generalized to richer problem formulations, and our main purpose here is to asses its merits in a "laboratory" environment.

While most of the known hierarchical approaches for solving hard scheduling problems reduce the problem size by decomposing the problem along the resources, our approach decomposes the problem along the types of decisions: the upper level assigns the jobs to weeks, and the lower level schedules the jobs assigned to a given week. Though this is a very natural decomposition approach, the computational advantages are not apparent at once. We use a compact problem formulation in which the decision variables represent only the assignment of jobs to weeks; but there will be no explicit variables for representing the schedule of those jobs assigned to the same week. Instead, we verify whether those jobs assigned to the same week can be completed during one week by using cutting planes, or as a last resort, by solving a parallel machine scheduling problem. In contrast to most previous approaches, we generate not only infeasibility or "no-good" cuts, but other problem specific cuts as well, and we try 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 a compact one suitable for decomposition, respectively. To strengthen the second formulation, we derive cutting planes from lower bounds for the bin-packing problem (Section 5.1), along with 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 integrated planning and scheduling problem as a monolithic mathematical program in Section 7.

2. Literature review

2.1. Parallel machine scheduling and bin-packing

By the parallel machine scheduling problem we mean the minimization of the makespan of n jobs on m identical parallel machines. For parallel ma-

chine scheduling, the worst case performance of $4/3 - 1/(3m)$ of the LPT rule (longest processing time first) is derived by Graham [5]. We will heavily exploit the strong connection between the parallel machine scheduling and the bin-packing problems, see Coffman et al. [6]. In that paper a new algorithm, called MULTIFIT, is presented, which uses ideas from bin-packing algorithms, and it is shown that it produces a schedule of makespan at most 1.22 times the optimum. However, its running time is larger than that of LPT, since in each iteration the FIRST-FIT-DECREASING bin-packing algorithm is run, which takes as much time as a single run of the LPT heuristic for parallel machine scheduling, and the desired number of iterations is about 7 for a large number of machines (over 8). This connection is pushed further by Hochbaum and Shmoys by developing the first polynomial time approximation scheme for the parallel machine scheduling problem [7]. In contrast to previous approaches, no weight function over the jobs is applied when deriving the approximation ratio of the algorithm. A thorough survey of approximation algorithms for bin-packing can be found in [8]. The lower bounds L_1 and L_2 (for bin-packing) are proposed in [9] to be used in exact algorithms. These bounds are enhanced in [10]. These lower bounds will be used in Section 5.1, where we give their precise definitions.

A cutting plane based approach for solving the parallel machine scheduling problem is proposed by Mokotoff [11]. The novelty of the method is a cutting plane which is valid for a specific face of the single-node fixed charge network model, and in fact can be derived from the well-know flow-cover inequality [12].

2.2. Hierarchical decomposition

Hierarchical decomposition approaches are applied widely in the field of production planning and scheduling. 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, 13]. 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 [14, 3]. Sawik [15] compares monolithic and hierarchical MIP formulations of an assembly line scheduling problem. In the hierarchical model, the upper level assigns jobs to resources and the lower level sequences them. The two levels are joined in a single-pass heuristic, and computational experiments have shown that the quicker hierarchical decomposition approach finds optimal solution for most of the instances.

Subsequently, we focus on exact solution methods that use hierarchical decomposition. One of the problems frequently addressed is the *multi-machine assignment and scheduling problem* (MMASP): a set of jobs, 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 jobs to machines, while a separate subproblem belongs to each

machine, sequencing the jobs on that machine. Jain and Grossmann [16] apply a MILP/CP approach, and add an infeasibility or "no-good" cut for the complete set of jobs scheduled on the machine where infeasibility is detected. Hooker and Ottoson [17] 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 [18] 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 [18] 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 [19]. The second class consists of a completely different type of cuts based on the edge-finding constraint propagation rule. Bockmayr and Pisaruk [20] investigate the problem of generating infeasibility cuts by CP for MILP in a general setting. The application of these ideas to MMASP leads to infeasibility cuts. MMASP has been generalized to cumulative resources in [21], and solved by a hybrid MIP/CP approach following the above decomposition scheme. MMASP is extended to multi-stage processes in [22]. 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 jobs is solved, but the resulting cuts may not be valid and 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 [23]. A decomposition approach is proposed, where the upper level sets production levels and inventories for macro time periods, and the lower level sequences the production activities. If the lower-level problem proves infeasible, then integer and logic cuts are fed back to the upper level. Both levels are described by and solved as a MILP.

Artigues et al. [24] investigate a hybrid decomposition based approach for an integrated employee timetabling and job-shop scheduling problem which is an extension of the classical job-shop scheduling problem. A decomposition-based CP formulation is proposed, which assigns jobs (possibly partially) to time periods (shifts). Guyon et al. [25] study a similar problem. In the proposed solution approach, there is a master problem for creating a timetable for the employees, while the subproblem checks if a feasible job 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.

A review of solution approaches has been presented by Grossmann et al [26]. The possible ways of integrating production planning and scheduling are surveyed in [4].

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 $T = \{1, \ldots, \tau\}$ index these 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 job 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., [10]): when the jobs 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. Formulation as a monolithic integer program

In this section we define a mathematical program for solving our integrated 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. In a feasible solution, 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., [27, 12, 28], 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. A problem decomposition based approach

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 scheduled 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 set B_t can be described as follows:

$$B_t := \left\{ z \in \{0,1\}^{|N|\times|T|} \;\middle|\; \begin{array}{l} z_t^j - \sum_{k\in M} x_{k,t}^j = 0,\ \forall j \in N_t \\ \sum_{k\in M} x_{kt}^j \le 1, \quad \forall\, j \in N_t, \\ \sum_{j\in N_t} p_j x_{kt}^j \le P, \quad \forall\, k \in M, \\ x_{kt}^j \in \{0,1\}, \quad \forall\, j \in N_t, k \in M. \end{array} \right\},$$

where $N_t := \{j \in N \mid r_j \ge t\}$. The condition $z \in B_t$ is closely related 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, see e.g., [6]. 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 scheduled on $|M|$ 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 cut off points not in 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 $L_1(H) := \lceil \sum_{j\in H} \tilde{p}_j \rceil$ [9]. To turn it into a valid inequality, suppose we have a set of jobs $\bar{H}$ to be defined later. Since

the set of jobs H has to be scheduled on $|M|$ identical parallel machines, the inequality

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

is valid for B_t.

L_2 inequalities. Now consider the lower bound L_2 of [9]. For a set of items H, the L_2 lower bound is

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

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

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

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

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

Notice that for $\varepsilon = \frac{1}{3}$, the following inequality is also valid

$$\left(\sum_{j\in H:\tilde{p}_j>\frac{2}{3}} z_t^j\right) + \left(\sum_{j\in H:\frac{1}{3}<\tilde{p}_j<\frac{2}{3}} 0.5 z_t^j\right) + \left(\sum_{j\in H:\tilde{p}_j=\frac{1}{3}\vee\tilde{p}_j=\frac{2}{3}} \tilde{p}_j z_t^j\right) \le |M| \tag{13'}$$

Fekete&Schepers inequalities. The third class of valid inequalities is derived from the lower bound $L_*^{(p)}$ of [10]. 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

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

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

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

The validity of this lower bound for the bin-packing problem is verified in [10]. 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, then the *infeasibility* cut

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

is valid for B_t, 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 along with (13'). 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 $\bar{H}_t = \{j \in N \mid \bar{z}_j^t > 0.01\}$, where $\bar{z}$ is the current optimal solution of the LP relaxation. Then we try to find $\varepsilon \in P_{\frac{1}{2}}(\bar{H}_t)$ such that a constraint is violated from (13). 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. add the L_2 cut (13) to the LP relaxation.
4. **end-if**
5. **end-loop**

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

Prior to trying to separate a violated inequality, we solve the parallel machine scheduling problem on the set of jobs $\bar{H}_t$ by using a simple heuristic, like LPT (longest processing time first). (We may also solve a bin-packing problem seeking a solution using at most m bins.) If the heuristic finds a feasible schedule on the m machines with makespan P or less, 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 jobs in $\tilde{H}_t = \{j \in N \mid \bar{z}_j^t > 0.1\}$ (jobs with smaller contribution than 0.1 to the period t are omitted) can be scheduled on $|M|$ parallel machines such that no machine receives a workload more than P, which is equivalent to a bin-packing problem. This bin-packing problem can be solved in a number of ways [9], and if the result is that more than $|M|$ machines are needed to schedule all the jobs in $\tilde{H}_t$, 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:

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

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

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

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

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

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 $\tilde{H}_t$, 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 $\tilde{H}_t$ 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 for $\bar{H}_t = \{j \in N \mid \bar{z}_j^t = 1\}$ 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. Notice that the solution may be improved by some heuristics, which we discuss in the end of this section.

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 2 (20 in the root node) 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. Note that inactive cuts are deleted from the LP relaxation in every 5th level of the search tree.

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 find a feasible solution. The heuristic sorts the jobs by non-increasing lateness penalties ℓ_j, and assigns them one-by-one to a time period and a 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.

If an integral assignment of jobs to periods is found in a search-tree node, then a schedule of minimum makespan is sought in each period using the LPT rule. If the makespan is at most P in all periods, then the solution is feasible and the hill climbing heuristic is applied to improve it.

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 solving monolithic mathematical program. Two different versions of the hierarchical solver has been tested, using different types of cuts:

- *Hierarchical H1*, a hierarchical solver using all the presented classes of cuts, namely infeasibility (16), L_2 (13), and Fekete&Schepers (15). The initial MILP formulation contained the L_1 (12) inequalities and the L_2 inequalities for H_t and $\varepsilon \in \left\{\frac{1}{2}, \frac{1}{3}\right\}$.
- *Hierarchical H2*, a hierarchical solver making use of infeasibility cuts (16) only. The initial MILP was the same as above.

All the methods have been implemented in the Mosel modeling language of FICO Xpress. The experiments were run on a 1.86 GHz Intel Xeon computer with 2 GB of RAM under Windows Server 2003. The time limit of a single run was set to 1200 seconds.

7.1. Test instances

Problem instances with different characteristics have been generated. The instances contained a mixture of so-called *small jobs* with processing times p_j taken at random from $U[1, 33]$ and *large jobs* with p_j from $U[34, 100]$, where $U[a, b]$ denotes the discrete uniform random distribution over the integers in $[a, b]$. The period length P was fixed to 100. To generate time windows for the jobs, 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 problem size was controlled by generating instances with different number of machines $|M|$ (the considered values were 2, 6, and 10), and nominal number of periods τ_0 (2, 6, and 10). The value of the number of jobs, $|N|$, was determined separately for each set of instances according to their complexity:

- Set A contained *small* jobs only, and the considered values of $|N|$ were 100, 150, and 200;
- Set B, with half of the jobs being *small* and half of them *large*, $|N| = 50, 100, 150$;
- Set C containing *large* jobs only, $|N| = 40, 60, 80$.

Each set consisted of five instances generated for every combination of parameters $|N|$, $|M|$, and τ_0, resulting in 405 instances altogether.

7.2. Detailed evaluation

The results are presented in Table 1 (results by number of jobs and time periods) and Table 2 (by number of jobs and machines). Each row of the tables displays combined results for the same value of $|N|$ and τ_0 (Table 1) or $|M|$ (Table 2). Columns *Opt* display the number of instances solved to proven

optimality out of 15; columns *UBRE* and *LBRE* contain the relative error of the upper and lower bounds found by the given method in percent, calculated using the following formulas:

$$UBRE_S^\delta = \frac{\sum_{i \in S} \frac{UB_i^\delta - LB_i^*}{LB_i^*}}{|S|} \cdot 100$$

$$LBRE_S^\delta = \frac{\sum_{i \in S} \frac{UB_i^* - LB_i^\delta}{UB_i^*}}{|S|} \cdot 100$$

Here, $UBRE_S^\delta$ and $LBRE_S^\delta$ are the relative error of the upper and lower bounds computed by solver δ on instance set S; UB_i^δ and LB_i^δ are the upper and lower bounds found by δ on instance i, and UB_i^* and LB_i^* are the best known bounds for the same instance. Note that for the instances whose optimal solution value was 0, all methods found proven optimal solutions, and therefore the above formula does not lead to division by zero. Columns *Time* contain the average computation times (including those runs where the time limit of 1200 seconds 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 all solvers could cope with large instances of set A, medium-sized problems in set B, while even smaller instances in set C were challenging. Furthermore, problems with large $|M|$ or large τ_0 were easier to solve for all methods, although the effect of these parameters is more apparent on the hierarchical solvers than on the monolithic. The hierarchical solvers outperformed the monolithic on sets A and B, whereas they achieved poorer results on set C. The detailed results differ significantly for the three sets, and therefore we review them separately for each set.

On set A, with small jobs only, the two hierarchical solvers outperformed the monolithic approach (107–108 versus 68 optimal solutions). Even for the instances not solved to optimality, the gap between the UBs and LBs found by the hierarchical solvers was insignificant, typically below 0.01%. The gap was considerably greater in case of the monolithic solver, and this solver also required higher run times than the hierarchical ones, partly due to the higher number of timeouts. The advanced cuts in method *Hierarchical H1* did not lead to improved performance, instead, it found one less optimal solution than *Hierarchical H2*.

The difference of the two hierarchical methods becomes spectacular on set B, where method *Hierarchical H1* dominated the other two solvers (58 versus 47–48 optimal solutions). This method not only found proven optimal solutions, but also, the relative error of the UBs found was considerably smaller than for the other methods.

The results on set C are less favorable: the monolithic solver found significantly more optimal solutions (115 versus 67–77), and achieved smaller gaps than the hierarchical methods. Nevertheless, the gap was mainly due to the

poor LBs, while the relative error of the UB was typically below 1%. The run times were also lower for the monolithic solver than for the hierarchical ones.

7.3. Final assessment

It can be stated that the hierarchical decomposition approach dominates the monolithic one when the lower level problems are easy to solve, i.e., for small jobs or many time periods, and in applications where the emphasis is on finding good solutions rather than computing lower bounds. For the easiest cases, even the simpler method *Hierarchical H2* with feasibility cuts works suitably well, whereas adding more advanced cuts (*Hierarchical H1*) extends the applicability of the hierarchical approach to more complicated cases. Large job instances are easy for the method using the monolithic formulation, since for such instances (flow) cover cuts are rather effective for single machines. Unfortunately, we have not found an efficient way of using them in the upper level of our decomposition method.

8. Final remarks

In this paper we have devised a hierarchical decomposition based method using cutting planes for solving an integrated planning and scheduling problem. In our method we generate not only infeasibility cuts, but valid cuts derived from the bin-packing problem as well, even if the solution is fractional. Of course, the strength of these cuts depends on the instances to be solved, and by extensive testing we have characterized those problem instances on which our method is particularly effective.

Since the parallel machine environment considered may have limited applications in practice, we plan to extend the model with machine dependent processing times and precedence constraints.

Acknowledgements

The authors are grateful to the anonymous referees for their constructive comments that helped to improve the paper in several ways. 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.

[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] Graham RL. Bounds on multiprocessor timing anomalies. SlAM Journal on Applied Mathematics 1969;17:416–29.

[6] Coffman EG, Garey MR, Johnson DS. An application of bin-packing to multiprocessor scheduling. SIAM J Computing 1978;7:1–17.

[7] Hochbaum DS, Shmoys DB. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. Journal of the Association for Computing Machinery 1987;34:144–62.

[8] Coffman EG, Garey MR, Johnson DS. Approximation algorithms for bin-packing: A survey. In: Hochbaum DS, editor. Algorithms for NP-hard problems; chap. 2. PWS Publishing, Boston; 1996, p. 46–93.

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

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

[11] Mokotoff E. An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research 2004;152:758–69.

[12] Padberg M, Van Roy TJ, Wolsey LA. Valid linear inequalities for fixed charge problems. Operations Research 1985;33:842–61.

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

[14] 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.

[15] Sawik T. Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. European Journal of Operational Research 2002;143:115–24.

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

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

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

			Hierarchical H1					Hierarchical H2					Monolithic			
Set	$\|N\|$	τ_0	Opt	UBRE	LBRE	Time	Cuts	Opt	UBRE	LBRE	Time	Cuts	Opt	UBRE	LBRE	Time
A	100	2	11	0.01	0.00	335.0	327	11	0.00	0.00	330.9	404	9	1.17	0.13	568.0
		6	15	0.00	0.00	5.8	5	15	0.00	0.00	4.4	7	10	1.67	0.36	400.3
		10	15	0.00	0.00	0.5	0	15	0.00	0.00	0.4	0	15	0.00	0.00	53.4
A	150	2	10	0.03	0.00	405.3	178	10	0.03	0.00	404.5	245	2	3.93	0.16	1040.3
		6	12	0.02	0.01	315.2	151	13	0.00	0.00	222.6	134	7	4.40	0.75	644.0
		10	15	0.00	0.00	40.0	20	15	0.00	0.00	33.9	19	10	5.73	0.43	400.8
A	200	2	9	0.01	0.00	549.0	178	9	0.02	0.00	532.1	189	0	3.75	0.20	1200.0
		6	10	0.05	0.00	401.2	131	10	0.02	0.00	401.1	155	5	4.46	0.42	800.5
		10	10	0.00	0.00	400.8	95	10	0.02	0.00	400.8	121	10	5.79	0.25	402.6
Subtotal A			107	0.01	0.00	272.5	120	108	0.01	0.00	259.0	141	68	3.43	0.30	612.2
B	50	2	11	0.68	0.17	390.8	5945	6	1.61	0.55	766.4	806	11	1.21	0.31	475.7
		6	15	0.00	0.00	19.0	848	14	0.00	0.03	101.4	236	15	0.00	0.00	40.3
		10	15	0.00	0.00	1.3	55	15	0.00	0.00	2.2	13	15	0.00	0.00	2.3
B	100	2	1	1.05	1.24	1162.4	14184	0	7.20	1.77	1200.0	347	0	6.28	0.16	1200.0
		6	4	29.57	0.62	1023.5	9911	1	36.31	1.40	1120.2	230	0	12.57	1.10	1200.0
		10	9	14.55	1.15	481.5	8176	10	0.25	0.73	576.7	214	5	18.73	1.65	800.6
B	150	2	0	2.49	1.36	1200.0	17435	0	3.18	1.56	1200.0	176	0	1.84	0.00	1200.0
		6	0	4.08	0.79	1200.0	14260	0	7.88	1.72	1200.0	342	0	8.26	0.51	1200.0
		10	3	1.48	0.63	961.0	8586	2	29.99	3.55	1043.0	141	1	31.97	2.95	1125.0
Subtotal B			58	5.99	0.66	715.5	8822	48	9.60	1.26	801.1	278	47	8.98	0.74	804.9
C	40	2	11	0.00	0.92	432.7	34446	8	0.02	1.96	609.1	7101	15	0.00	0.00	2.8
		6	14	0.09	0.25	96.4	7872	13	0.01	0.22	168.6	533	15	0.00	0.00	1.6
		10	13	0.01	0.25	161.0	7767	14	0.00	0.12	142.4	458	15	0.00	0.00	1.1
C	60	2	4	1.04	2.90	985.3	44914	3	1.77	4.50	1055.2	6102	13	0.00	0.00	217.3
		6	10	0.36	0.66	447.1	18591	8	0.75	1.94	625.7	2339	15	0.00	0.00	5.7
		10	12	0.26	0.81	252.1	11820	12	0.27	1.20	316.5	846	15	0.00	0.00	8.9
C	80	2	0	0.88	3.08	1200.0	35366	0	0.97	4.70	1200.0	3805	7	0.04	0.00	730.0
		6	4	2.04	2.95	895.9	40129	2	2.03	6.33	1160.8	3508	7	0.00	0.20	662.2
		10	9	0.37	1.14	495.9	15798	7	0.91	2.42	680.0	1496	13	0.00	0.33	189.1
Subtotal C			77	0.56	1.44	551.8	24078	67	0.75	2.60	662.0	2909	115	0.00	0.06	202.1
Total			242	2.19	0.70	513.3	11007	223	3.45	1.29	574.0	1109	230	4.14	0.37	539.7

Table 1: Experimental results by number of jobs and time periods.

			Hierarchical H1					Hierarchical H2					Monolithic			
Set	$\|N\|$	$\|M\|$	Opt	UBRE	LBRE	Time	Cuts	Opt	UBRE	LBRE	Time	Cuts	Opt	UBRE	LBRE	Time
A	100	2	11	0.01	0.00	340.6	332	11	0.00	0.00	335.1	411	5	2.78	0.44	853.1
		6	15	0.00	0.00	0.6	0	15	0.00	0.00	0.5	0	14	0.05	0.05	158.3
		10	15	0.00	0.00	0.1	0	15	0.00	0.00	0.1	0	15	0.00	0.00	10.2
A	150	2	7	0.05	0.01	754.8	349	8	0.03	0.00	656.1	397	0	10.17	0.65	1200.0
		6	15	0.00	0.00	5.0	1	15	0.00	0.00	4.2	1	7	2.94	0.66	643.8
		10	15	0.00	0.00	0.7	0	15	0.00	0.00	0.7	0	12	0.96	0.03	241.3
A	200	2	0	0.06	0.01	1200.0	379	0	0.06	0.00	1200.0	440	0	9.11	0.36	1200.0
		6	14	0.00	0.00	139.4	25	14	0.00	0.00	122.5	25	5	3.30	0.40	802.0
		10	15	0.00	0.00	11.5	0	15	0.00	0.00	11.4	0	10	1.59	0.12	401.1
Subtotal A			107	0.01	0.00	272.5	120	108	0.01	0.00	259.0	141	68	3.43	0.30	612.2
B	50	2	12	0.68	0.17	285.8	6800	10	0.37	0.43	436.0	721	14	0.00	0.00	215.0
		6	15	0.00	0.00	45.1	47	12	0.62	0.15	273.8	334	13	1.01	0.31	201.8
		10	14	0.00	0.00	80.3	0	13	0.63	0.00	160.2	0	14	0.20	0.00	101.5
B	100	2	0	1.30	1.70	1200.0	26464	0	0.76	2.12	1200.0	556	0	3.88	0.00	1200.0
		6	7	14.95	1.06	784.9	4986	5	10.19	1.28	896.7	223	0	29.24	2.64	1200.0
		10	7	28.90	0.25	682.6	821	6	32.81	0.49	800.2	12	5	4.47	0.26	800.6
B	150	2	0	0.53	1.70	1200.0	26365	0	0.05	1.81	1200.0	289	0	3.64	0.00	1200.0
		6	0	0.54	0.70	1200.0	13226	0	11.64	2.04	1200.0	358	0	16.63	0.96	1200.0
		10	3	6.98	0.38	961.0	691	2	29.35	2.98	1043.0	12	1	21.80	2.51	1125.0
Subtotal B			58	5.99	0.66	715.5	8822	48	9.60	1.26	801.1	278	47	8.98	0.74	804.9
C	40	2	9	0.10	1.09	555.3	36163	8	0.04	0.92	633.7	3197	15	0.00	0.00	2.3
		6	14	0.00	0.32	113.3	10883	13	0.00	0.69	161.1	1975	15	0.00	0.00	1.8
		10	15	0.00	0.00	21.5	3039	14	0.00	0.70	125.3	2920	15	0.00	0.00	1.3
C	60	2	3	0.91	2.76	997.2	34644	3	0.95	3.46	1069.9	2213	15	0.00	0.00	41.0
		6	11	0.35	0.83	384.9	26411	9	0.64	2.46	574.3	4253	14	0.00	0.00	102.6
		10	12	0.39	0.78	302.3	14269	11	1.20	1.71	353.3	2821	14	0.00	0.00	88.3
C	80	2	0	0.86	3.34	1200.0	35699	0	0.79	3.71	1200.0	1437	8	0.00	0.00	587.6
		6	4	1.57	2.90	894.2	36361	2	2.13	6.02	1058.1	3033	10	0.04	0.00	487.8
		10	9	0.86	0.92	497.6	19233	7	0.99	3.73	782.6	4339	9	0.00	0.54	505.9
Subtotal C			77	0.56	1.44	551.8	24078	67	0.75	2.60	662.0	2909	115	0.00	0.06	202.1
Total			242	2.19	0.70	513.3	11007	223	3.45	1.29	574.0	1109	230	4.14	0.37	539.7

Table 2: Experimental results by number of jobs and machines.