Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Computer Science and Information Theory

Hungarian Academy of Sciences
Institute for Computer Science and Control
Research Laboratory on Engineering and Management Intelligence

Machine Scheduling with Additional Resource Constraints

by

Márton Drótos

PhD Thesis booklet

Supervisor:

Gábor Wiener

Department of Computer Science and Information Theory
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics
Budapest, Hungary

Advisor:

Tamás Kis

Research Laboratory on Engineering & Management Intelligence
Institute for Computer Science and Control
Hungarian Academy of Sciences
Budapest, Hungary

Machine Scheduling with Additional Resource Constraints

PhD Thesis booklet

Introduction

Machine scheduling is a popular topic in operations research with great practical relevance. In the last decades, several models and algorithms were proposed in order to model and solve problems that appear in today's industries. Although most of these problems are NP-hard, the increase in the computing power enables us to incorporate more and more details into our models. The environment is also changing: there is a considerable tendency from mass production to customized make-to-order production. With the evolution of industry, the models and algorithms must also be changed: the main objectives become different (e.g., customer service becomes more important than production throughput), and new types of bottlenecks are introduced (such as more expensive workforce, the need for reducing the stock size, etc.).

My work is motivated by these needs, and I investigate scheduling problems with renewable cumulative resources (such as workers/toolings), non-renewable resources (such as raw materials built into the products), and I try to address the need for a system that can be used in real-life environments. I have formulated my new results as Theses 1-3:

Thesis 1
I have investigated the scheduling of inventory releasing jobs on a single machine to minimize tardiness related criteria, and I have devised various complexity results as well as polynomial, pseudo-polynomial and approximation algorithms. In this model, jobs are supplying various resources that are required by deliveries with due dates.

I have proven that the problem is solvable in polynomial time if all jobs produce the same amount of resources, or there is only one resource type and the processing times of each job are equal (Theorem 1.1 and Theorem 1.2). I have also proven that the problem is NP-hard in the strong sense even if all jobs have unit processing times and there are at least two types of resources or two deliveries (Theorem 1.3 and Theorem 1.4), and it is NP-hard in the ordinary sense even if there is at least one type of resource and two deliveries (Theorem 1.5).

Furthermore, I have given a pseudo-polynomial time algorithm for the special case where the number of the resource types and the number of the deliveries are constants (Theorem 1.6), and a fully polynomial time approximation scheme for the special case where there is one type of resource and there are two deliveries (Theorem 1.7).

Thesis 2

I have devised a branch and bound algorithm for the resource leveling problem in a dedicated parallel machine environment (Algorithm 2.1). I demonstrate the performance of the algorithm by comparing its results on randomly generated test data to results provided by a commercial solver.

I have formulated the problem as an integer programming problem, and using this formulation, I have given an algorithm that calculates lower bounds using Lagrangian relaxation (Theorem 2.1).

In order to obtain improved lower bounds and to reduce the size of the search tree, I have provided a shaving technique based on the Lagrangian relaxation (Theorem 2.2 and Theorem 2.3).

I have given a method to calculate an upper bound in the nodes of the search tree (Lemma 2.1 and Algorithm 2.2).

I have also devised a novel branching scheme that improves the efficiency of the branch and bound method (Section 2.4.5).

Thesis 3

I have described a framework for solving complex industrial scheduling problems (Algorithm 3.1), and I have given various algorithms that can be used in its stages for different models of industrial problems, along with novel local search neighborhoods. I demonstrate the effectiveness of the framework by two industrial case studies.

The main focus of the framework is to handle renewable and non-renewable resource constraints. I have given an algorithm, using dynamic programming, for assigning non-renewable resources to jobs (Algorithm 3.2).

As the framework relies on local search algorithms, I have created two algorithms for the efficient calculation/recalculation of the start and end times of jobs in various scheduling models. Algorithm 3.3 focuses on the fast recalculation of the start and end times, while Algorithm 3.4 handles renewable cumulative resource constraints in complex models.

I have also given an algorithm that handles renewable cumulative resource constraints in certain simple models (Algorithm 3.5).

The results related to Thesis 1, 2 and 3 are explained in more details in sections 1, 2 and 3, respectively. To follow the stylistic conventions of the field, I use the first-person personal pronouns in their plural form throughout the rest of my work.

1 Scheduling of inventory releasing jobs to minimize tardiness related criteria

1.1 Motivations

Consider a manufacturing environment, where a set of inventory releasing jobs have to be sequenced on a single machine (or server) in order to meet a set of delivery requests. This means that instead of individual jobs having due dates, which is common in the practice of machine scheduling, the desired amount of resources at given times have to be provided.

A practical application would be a factory working with the Just-In-Time model, where the materials required for the production are transported to the factory by suppliers just before they are used. A material shortage would mean that either the production is delayed, or it has to be rescheduled. Therefore, it is desired to give a schedule for the unloading of the arriving materials, respecting constraints imposed by unloading time, capacities, etc. This way it is possible to ensure that production is delayed as little as possible.

1.2 Definition of the problem

There is a set of jobs J and a set of different product types S with $|J| = n^J$, and $|S| = n^S$. Each job $j \in J$ has a processing time $p_j > 0$, and produces an amount of $\tilde{c}_j^s \geq 0$ from product $s \in S$. It is permitted that the same job produces a positive amount from several distinct product types. The inventory level of each product is 0 initially. The inventory of products is consumed by a set of deliveries R with $|R| = n^R$. Each delivery R_r has a due-date d_r, and specifies a quantity of $\tilde{e}_r^s$ for each product type $s \in S$ to be withdrawn from the corresponding inventory. Like in the case of jobs, a delivery may specify positive requests for several product types simultaneously. It is assumed that $d_1 \leq d_2 \leq \cdots \leq d_{n^R}$. A delivery can be served only if the inventory level of each product type is above or equal to the requested quantity. Hence, a delivery can be *tardy*. The delivery requests must be served in increasing due-date order, and in case of ties, their order is fixed in advance. The indexing of delivery requests indicates the order in which they have to be served.

Let π be a sequence of the n^J jobs, i.e., $\pi(u) \in J$ is the job in position u $(1 \leq u \leq n^J)$, and $\pi(u) \neq \pi(v)$ for $u \neq v$. The total amount of product type $s \in S$ after completing the first h jobs in π is $\sum_{u=1}^{h} \tilde{c}_{\pi(u)}^s$, whereas the total requested quantity over the first r delivery requests is $e_r^s = \sum_{k=1}^{r} \tilde{e}_k^s$. Let h_r^π be the minimum number of jobs needed to reach the required inventory level by delivery r, i.e., $h_r^\pi := \min \left\{ h \mid \sum_{u=1}^{h} \tilde{c}_{\pi(u)}^s \geq e_r^s, \text{ for all } s \in S \right\}$.

The *completion time* of delivery r is $C_r^\pi := \sum_{u=1}^{h_r^\pi} p_{\pi(u)}$ (there is no idle time between the jobs). If $C_r^\pi \leq d_r$, then the delivery is *on time*, otherwise it is *tardy*. The *lateness* and *tardiness* of delivery r are defined as $L_r^\pi = C_r^\pi - d_r$ and $T_r^\pi := \max(0, L_r^\pi)$, respectively. The *completion time of job* $\pi(h)$ in a sequence π is $\tilde{C}_h^\pi = \sum_{k=1}^{h} p_{\pi(k)}$ (there are no idle times between jobs). The *total amount of product type* s produced up to time t in schedule π is $A_\pi^s(t) = \sum_{h|\tilde{C}_h^\pi \leq t} \tilde{c}_{\pi(h)}^s$. We will consider regular (non-decreasing) objective functions γ in the delivery completion times, e.g., maximal tardiness ($\max_r T_r$), maximal lateness ($\max_r L_r$), total weighted tardiness ($\sum_r w_r T_r$) or total weighted lateness ($\sum_r w_r L_r$). Let $\gamma(\pi)$ denote the objective function value of a schedule π. Since π, and the number of delivery requests are finite, γ is well defined.

Without loss of generality, we may assume that the due-date of the last delivery satisfies $d_{n^R} \geq \sum_j p_j$, and $e_{n^R}^s = \sum_j \tilde{c}_j^s$ for all $s \in S$. Namely, if $d_{n^R} < \sum_j p_j$, we can add a new delivery request to R, and set its due-date to $\sum_j p_j$. On the other hand, by that time, all the products are delivered, hence we lose no solution by assuming $e_{n^R}^s = \sum_{j \in J} \tilde{c}_j^s$ for $s \in S$.

An illustrative example is given in Figure 1.

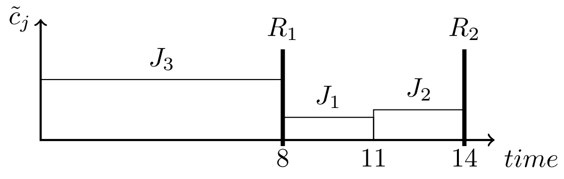
Parameters:

Param.	Value
n^J	3
n^S	1
n^R	2

Job	p_j	$\tilde{c}_j$
J_1	3	3
J_2	3	4
J_3	8	8

Delivery	d_r	$\tilde{e}_r$	e_r
R_1	7	6	6
R_2	14	9	15

Schedule $\pi(J_3, J_1, J_2)$:



Total production of $\pi(J_3, J_1, J_2)$:

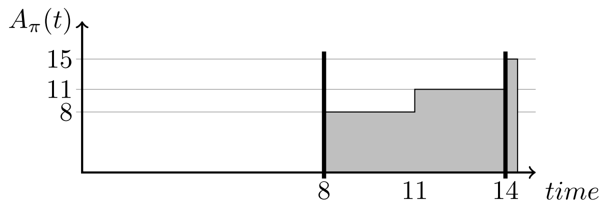


Figure 1: Example of the scheduling problem. The boxes in the schedule represent the jobs; the length and height of job j is proportional to p_j and $\tilde{c}_j$, respectively. The depicted schedule has a maximal tardiness of 1 time unit.

1.3 Background and related results

This work complements the recent results of Boysen et al. (2013). In that paper, the delivery requests have strict deadlines, and special cases where the existence of a feasible schedule is decidable in polynomial time are considered with the objective of minimizing some stock level related criteria. An optimal or suboptimal solution to our problem could be used to set hard deadlines, and then in a second round one can apply the methods of Boysen et al. to minimize the stock levels.

The opposite problem, i.e., consuming the inventory of raw materials subject to some constraints, is analyzed in (Grigoriev et al.; 2005). In that model raw materials of various types are supplied over time, and each job requires some of them in pre-specified quantities. A job may be started only if there is enough raw material on stock at the beginning of the job. If a job is started, the stock level of all the required materials are decreased by the quantities required by the job. The authors study several variants and provide NP-hardness proof, and approximation algorithms for the hard problems, or polynomial time algorithms for the tractable ones.

In a more general setting jobs may produce and consume a common set of materials, and a basic question is whether a feasible sequence of producers and consumers exists. This problem has been shown NP-hard in the strong sense, cf. (Kellerer et al.; 1998). This line of work has been taken up by Briskorn et al. (2010), where several variants are studied, and either an NP-hardness proof is presented, or a polynomial time algorithm is devised.

1.4 Results

We have given complexity proofs for special cases of the problem $1||\gamma$, and have devised polynomial and pseudo-polynomial algorithms for all the settings where this is possible (if $P \neq NP$). Furthermore, an FPTAS is given for one of the cases. It has been shown that even for very restricted models, this problem is very hard to solve. Our results are summarized in Table 1. For the sake of completeness, some results of other researchers are also shown.

Theorem 1.1. *The problem $1|p_j = p, n^S = 1|\gamma$ can be solved in $O(n^J \log n^J)$ time.*

Theorem 1.2. *The problem $1|\tilde{c}^s_j = \tilde{c}^s|\gamma$ can be solved in $O(n^J \log n^J)$ time.*

Theorem 1.3. *The problem $1|n^R = 2, p_j = 1|\gamma^T$ is NP-hard in the strong sense.*

Theorem 1.4. *The problem $1|n^S = 2, p_j = 1|\gamma^T$ is NP-hard in the strong sense.*

		$n^S = 1$		$n^S \geq 2$, const.		$n^S = *$	
		$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$	$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$	$\tilde{c}_j^s = \tilde{c}^s$	$\tilde{c}_j^s = *$
	$p_j = 1$	P (1.2)	P (1.1)	P (1.2)	o.NP[a]	P (1.2)	s.NP (1.3)
					p.poly. (1.6)		
$n^R \geq 2$,	$p_j = p$	P (1.2)	P (1.1)	P (1.2)	o.NP[a]	P (1.2)	s.NP (1.3)
const.					p.poly. (1.6)		
	$p_j = *$	P (1.2)	o.NP (1.5)	P (1.2)	o.NP[a]	P (1.2)	s.NP (1.3)
			p.poly. (1.6)		p.poly. (1.6)		
	$p_j = 1$	P (1.2)	P (1.1)	P (1.2)	s.NP (1.4)	P (1.2)	s.NP (1.3)
$n^R = *$	$p_j = p$	P (1.2)	P (1.1)	P (1.2)	s.NP (1.4)	P (1.2)	s.NP (1.3)
	$p_j = *$	P (1.2)	s.NP[b]	P (1.2)	s.NP (1.4)	P (1.2)	s.NP (1.3)

The following results of other researchers are also shown in the table:
[a] $1|p_j = 1, n^S = 2, n^R = 2|\gamma^T$ is NP-hard was shown in (Drótos and Kis; 2013a)
[b] $1|n^S = 1|\gamma^T$ is NP-hard in the strong sense was shown in (Boysen et al.; 2013)

Table 1: Overview of the complexity results for $1||\{\gamma, \gamma^T\}$. Each cell of the table corresponds to a variant of the problem with additional restrictions; a star ($*$) indicates that the problem parameter can be arbitrary positive integer value. The complexity can be polynomial (P), NP-hard in the ordinary sense (o.NP), or NP-hard in the strong sense (s.NP). The fact that a pseudo-polynomial algorithm exists is denoted by p.poly. The objective function is any regular function γ in the delivery completion times for those problems of polynomial time complexity, and it is any regular function γ^T in the delivery tardiness times for all NP-hard problems in the table. Numbers (in parenthesis) after the complexity classes refer to the corresponding theorems.

Theorem 1.5. *The problem* $1|n^S = 1, n^R = 2|\gamma^T$ *is NP-hard in the ordinary sense.*

Theorem 1.6. *The problem* $1|n^S = const., n^R = const.|\gamma$ *can be solved in pseudo-polynomial time.*

Theorem 1.7. *There exists an FPTAS for the problem* $1|n^S = 1,\ n^R = 2|T_{\max}$.

1.5 Publications

The results of Thesis 1 were published in (Drótos and Kis; 2013a,b; Drótos et al.; 2013).

2 Resource leveling in a parallel machine environment

2.1 Motivations

Renewable cumulative resources are very useful to model workforce requirements in a factory. Consider an environment where the workforce consists of permanent as well as temporary workers. If the load of the the workshop fluctuates over time, then in peak periods, when more workers are needed than the number of permanent employees, extra workers may be hired to meet the demand. To reduce hiring costs, a resource leveling problem has to be solved with appropriate objective function.

In a typical scheduling system, when considering workforce requirements, some part of the schedule is already fixed based on other measures (e.g. due dates). This means that tasks are already assigned to machines, and a time window is defined for them in which they can be performed such as the decisions on the higher level are not violated. In this part, we provide an exact method to find optimal or close-optimal solutions.

2.2 Definition of the problem

The main data and parameters of the scheduling problems we will study are as follows: There are m parallel machines, M_1 through M_m, L renewable resources R_1 through R_L, along with target levels $C_1, \ldots, C_L \in \mathbb{Q}$, and a set of n tasks, J, partitioned into m disjoint subsets $J = J_1 \cup J_2 \cdots \cup J_m$. Those tasks in J_i have to be processed exclusively on machine M_i. Each task j has a processing time p_j, release time e_j, a deadline d_j (all integral numbers), and resource requirements $b_{j1}, \ldots, b_{jL}$. Preemption of tasks is not allowed. No machine can process more than one task at a time, but there is no limitation on the maximum parallel usage of the resources. A *schedule* $\mathcal{S}$ specifies the starting time of every task, i.e, $\mathcal{S} = (S_1, \ldots, S_n)$, where S_j is the starting time of task j. A schedule is *feasible* if $e_j \leq S_j$ and $S_j \leq d_j - p_j$ hold for every task j, and the processing of any pairs of tasks j and k on the same machine is performed in disjoint time periods, i.e., $[S_j, S_j + p_j) \cap [S_k, S_k + p_k) = \emptyset$ for $j, k \in J_i$ with $j \neq k$.

In order to describe the objective functions considered in this chapter we need additional definitions. Let D denote the largest integer time point when some task may be started, i.e., $D = (\max_{j \in J} d_j) - 1$. The *resource profile* of schedule $\mathcal{S}$ is a function $R^{\mathcal{S}} = (r_1^{\mathcal{S}}, \ldots, r_L^{\mathcal{S}})$, where $r_\ell^{\mathcal{S}} : [0, D+1] \to \mathbb{Q}_+$ gives the total requirement from resource ℓ in time point t w.r.t. schedule $\mathcal{S}$, i.e., $r_\ell^{\mathcal{S}}(t) = \sum_{j \in J : S_j \leq t < S_j + p_j} b_{j\ell}$. We will consider the following type of objective functions:

$$f(R^{\mathcal{S}}) := \sum_{\ell=1}^{L} \sum_{t=0}^{D} \hat{f}_\ell(r_\ell^{\mathcal{S}}(t), C_\ell),$$

where the functions $\hat{f}_\ell : \mathbb{Q}_+ \times \mathbb{Q}_+ \to \mathbb{Q}_+$ satisfy $\hat{f}_\ell(x, y - z) = \hat{f}_\ell(x + z, y)$. For instance, with $\hat{f}_\ell(x, y) = w_\ell \max\{0, x - y\}$, where $w_\ell \geq 0$, we obtain

$$f_{\text{lin}}(R^{\mathcal{S}}) = \sum_{\ell=1}^{L} w_\ell \sum_{t=0}^{D} \max\{0, r_\ell^{\mathcal{S}}(t) - C_\ell\},$$

while $\hat{f}_\ell(x, y) = w_\ell(x - y)^2$ yields

$$f_{\text{quad}}(R^{\mathcal{S}}) = \sum_{\ell=1}^{L} w_\ell \sum_{t=0}^{D} (r_\ell^{\mathcal{S}}(t) - C_\ell)^2.$$

Of course, for distinct resources, the functions $\hat{f}_\ell$ may be different.

2.3 Background and related results

Traditionally resource leveling is a topic of project scheduling. One of the first heuristics for resource leveling is due to Burgess and Killebrew (1962). This procedure is applicable to CPM/PERT networks consisting of activities (nodes) and temporal relations between them (arcs). For a recent review of resource constrained project scheduling, see (Hartmann and Briskorn; 2010). Further information on the topic, including several models and algorithms, can be found in (Demeulemeester and Herroelen; 2002).

There is a considerable literature on machine scheduling with additional resources. Initially, Błażewicz et al. (1983) proposed the classification scheme and provide several complexity results for machine scheduling under resource constraints. For a comprehensive overview and references, see the textbook (Błażewicz; 2001).

However, to our knowledge, there are no results so far for resource leveling in machine scheduling.

In the presented model, as deciding whether a feasible schedule exists for even one machine is NP-hard in the strong sense ([SS1] in Garey and Johnson; 1979), the analysis of the complexity of our problem wouldn't reflect the additional complexity introduced by the resources. Therefore it is worth investigating the cases where finding a feasible schedule isn't an issue. The related complexity results can be found in (Drótos and Kis; 2011).

2.4 Results

In order to solve the general problem, we have devised a branch-and-bound method, in which the nodes in the branch-and-bound search tree represent a constrained version of the original problem, where the time windows of the tasks are narrowed, and the root node represents the original problem. An illustrative example of the branch-and-bound tree is given in Figure 2.

Algorithm 2.1. *The evaluation of a node in the branch and bound tree consists of the following steps:*

1. *Constraint propagation*
2. *Calculation of lower bound*
3. *Shaving*
4. *Calculation of upper bound*
5. *Branching*

2.4.1 Constraint propagation

We use some well-known one machine constraint propagation techniques to narrow the time windows of the tasks. If infeasibility is proven on any machine, then the actual search tree node is infeasible.

2.4.2 Calculation of lower bound

In order to provide a method for calculating a lower bound on the objective function value, we start by formulating the resource leveling problem as an integer programming problem. The decision variables are $x_{jt} \in \{0,1\}$, $j \in J$, $t \in \{e_j, \ldots, d_j - p_j\}$, and $y_{\ell t} \in \mathbb{R}$, $j \in J$, $t \in \{0, \ldots, D\}$. The x_{jt}'s determine a schedule $\mathcal{S}$ by indicating the starting time of each task, in which $y_{\ell t}$ represents the resource usage $r_\ell^{\mathcal{S}}(t)$.

$$\min \sum_{\ell=1}^{L} \sum_{t=0}^{D} \hat{f}_\ell(y_{\ell t}, C_\ell) \tag{1}$$

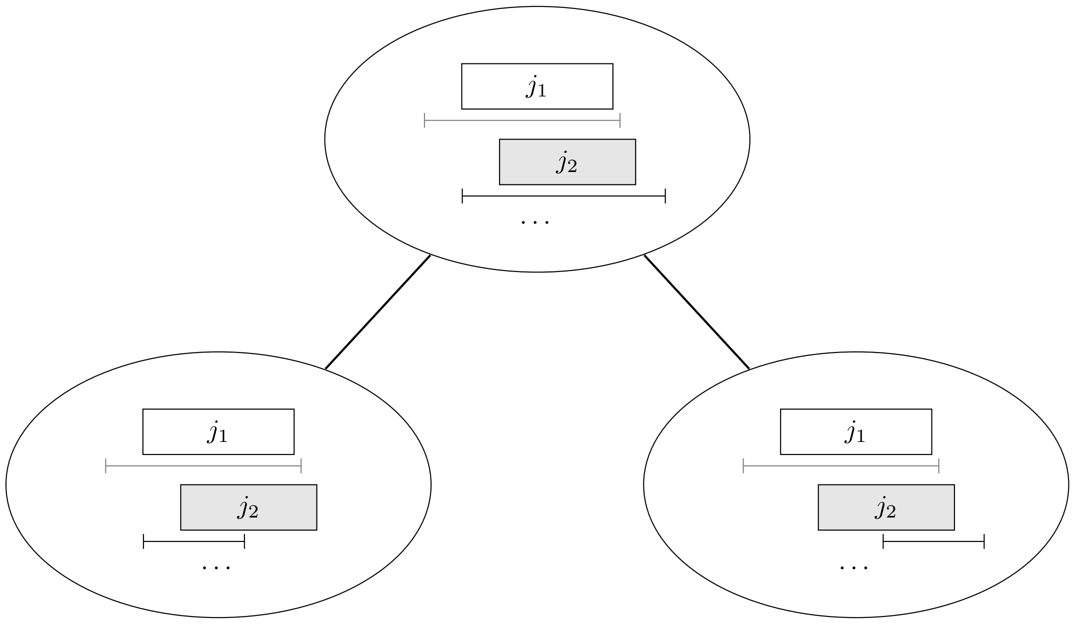


Figure 2: Illustration of the branch and bound tree. In each branching step, the time window of one task is partitioned into smaller intervals. The example shows a situation where the time window of task j_2 is divided into two smaller intervals, resulting in two, more constrained versions of the problem in the parent node.

subject to

$$\sum_{t=0}^{D} x_{jt} = 1, \quad \forall\, j \in J, \tag{2}$$

$$\sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} \leq 1, \quad \forall\, t \in \{0, \dots, D\},\ i \in \{1, \dots, m\} \tag{3}$$

$$\sum_{j \in J} \sum_{\tau=t-p_j+1}^{t} b_j x_{j\tau} - y_{\ell t} = 0, \quad \forall\, t \in \{0, \dots, D\},\ \ell \in \{1, \dots, L\} \tag{4}$$

$$x_{jt} \in \{0, 1\}, \quad \forall j \in J,\ t \in \{e_j, \dots, d_j - p_j\}. \tag{5}$$

To obtain the Lagrangian relaxation of the problem (1)-(5), we dualize (4). Since the objective function and the constraints are separable, we obtain the following mathematical program:

$$LB(\boldsymbol{\lambda}) = \sum_{i=1}^{m} LB_i(\boldsymbol{\lambda}) + \min_{y} \sum_{\ell=1}^{L} \sum_{t=0}^{D} \left(\hat{f}_\ell(y_{\ell t}, C_\ell) - \lambda_{\ell t} y_{\ell t} \right), \tag{6}$$

where

$$LB_i(\boldsymbol{\lambda}) = \min \sum_{\ell=1}^{L} \sum_{t=0}^{D} \sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} \lambda_{\ell t} b_{j\ell} x_{j\tau} \tag{7}$$

subject to

$$\begin{aligned} \sum_{t \in \{e_j,\ldots,d_j-p_j\}} x_{jt} &= 1, \quad \forall\, j \in J_i, & (8) \\ \sum_{j \in J_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad \forall\, t \in \{0,\ldots,D\} & (9) \\ x_{jt} &\in \{0,1\}, \quad \forall j \in J_i,\ t \in \{e_j,\ldots,d_j-p_j\}. & (10) \end{aligned}$$

The $LB_i(\boldsymbol{\lambda})$ are integer programs which would still be very hard to solve, so their linear relaxations are considered (by replacing the $x_{jt} \in \{0,1\}$ constraints with $0 \leq x_{jt} \leq 1$).

Theorem 2.1. *$LB(\boldsymbol{\lambda})$ is a lower bound of the problem for any $\boldsymbol{\lambda} = (\lambda_{\ell t} : \ell \in \{1,\ldots,L\}, t \in \{0,\ldots,D\})$, and for any fixed $\boldsymbol{\lambda}$ it can be calculated by solving m independent one machine problems.*

The advantage of this approach is that the structure and the size of the LP-s (identified by (7)-(10)) are independent of the number of resources and the objective function. Since the size of these LP's is $O(Dn_i)$, they can be solved in pseudo-polynomial time in the size of the scheduling problem instance.

The goal is to find the lower bound $LB = \max_{\boldsymbol{\lambda}} LB(\boldsymbol{\lambda})$. In order to approximate the optimal Lagrangian multipliers we use the subgradient method, while respecting constraints that may be imposed by certain objective functions.

2.4.3 Shaving

Shaving is an efficient propagation technique in constraint programming (introduced by Carlier and Pinson (1994) and Martin and Shmoys (1996)), which aims at narrowing the domain of some variables without generating choice points. This is done by fixing a decision variable to one of its possible values, then, after constraint propagation, excluding this value if it results in infeasibility. For an overview of shaving techniques, we refer the reader to (Torres and Lopez; 2000).

In the studied model, for each task j on machine i with a time window with non-empty interior we calculate the lower bound $\tilde{LB}_{ijt}$ (with $\boldsymbol{\lambda}^*$) for the LP problems where j is scheduled at time t ($e_j \leq t \leq d_j - p_j$). By analysing the results, it is possible that the time window of certain tasks may be narrowed without losing any solutions, and the lower bound may also be improved.

Theorem 2.2. *The value $\tilde{LB}_i = \max_{j \in J_i} \min_{e_j \leq t \leq d_j - p_j} \tilde{LB}_{ijt}$ provided by the shaving procedure is a valid lower bound for the corresponding branch and bound tree node.*

Theorem 2.3. *If the shaving procedure removes the time point t from the possible starting times of task j, then no such solution exists for the corresponding branch and bound tree node where task j is started at time t and the objective function value is better than the currently known best upper bound.*

2.4.4 Calculation of upper bound

In each node of the branch-and-bound tree, a feasible schedule is sought with respect to the actual time windows. As the machines are independent in terms of feasibility, it is possible to construct a solution on each machine individually. Because of deciding whether there exists a feasible schedule on a machine is NP-complete, we are limited to heuristic and approximation methods. The idea behind our approach is to use the solution of the Lagrangian relaxation to make an initial schedule, as it is expected to be similar to the optimal solution of the problem. In case this method fails, we fall back to an approximation algorithm. If a schedule is found, it is improved heuristically to obtain a good upper bound.

Lemma 2.1. *For any machine, the approximation algorithm of Hall and Shmoys (1982) can be used to obtain one of the following observations:*

(a) A feasible schedule exists and it is found.

(b) A feasible schedule may or may not exist, but no such schedule is found.

(c) A feasible schedule does not exist.

Algorithm 2.2. *A feasible schedule is sought on each machine as follows:*

1. *The Lagrangian relaxation provides a feasible schedule. This solution is used without any changes.*

2. *A feasible schedule can be devised from the solution of the Lagrangian relaxation by rounding. This is used as an initial schedule on the machine.*

3. *If no feasible schedule is found so far, then the results of Lemma 2.1 is used. There are three possible outcomes:*

   (a) *A feasible schedule is found. This is used as initial schedule on the machine.*

   (b) *A feasible sequence may exist, but it is not found. In this case no upper bound is calculated.*

   (c) *No feasible sequence exists on the machine, so the corresponding node in the branch-and-bound tree is infeasible.*

In order to improve the quality of the solution, a local search algorithm is performed on those machines for which a feasible schedule is found in Step 2 or Step 3 of the above algorithm. This local search procedure uses the polynomial right-shifting algorithm described in (Drótos and Kis; 2011).

2.4.5 Branching

In each node a predefined number of children are generated. This is done by choosing a task j, and dividing its time window into B sub-windows. We want to ensure that in each sub-window, the $\tilde{LB}_{ijt}$ values (defined in Section 2.4.3) are about the same. To this end, we define and solve an optimization problem that provides the desired partitioning.

Using this method the generated nodes represent the structure of the problem, by partitioning the time windows in a natural way (the tasks will have *good* and *bad* time windows in different nodes). Its other advantage is that the sibling nodes are likely to have different lower bounds, so the promising combinations of time windows can be evaluated at first.

2.4.6 Computational results

In order to assess the performance of our exact as well as the heuristic methods, we have implemented them in C++ and generated a series of test instances. We evaluated several variants of our exact algorithm and compared the results to those obtained by the commercial solver ILOG CPLEX 11.2 using the MIP formulation of the resource leveling problem (1)-(5). The test set contain problems with $m = 5, 10, 20$ machines and $t = 10, 15$ and 20 tasks per machine, with each type having 10 instances.

	m5		**m10**		**m20**		**avg**	
	BB	*CPX*	*BB*	*CPX*	*BB*	*CPX*	*BB*	*CPX*
t10	6.16%	3.10%	0.74%	0.24%	0.36%	0.37%	**2.42%**	**1.24%**
t15	12.94%	11.28%	5.08%	5.96%	0.41%	1.62%	**6.14%**	**6.29%**
t20	18.39%	17.15%	5.41%	7.48%	2.19%	10.39%	**8.66%**	**11.67%**
avg	**12.49%**	**10.51%**	**3.74%**	**4.56%**	**0.99%**	**4.13%**	**5.74%**	**6.40%**

Table 2: Average optimality gap for the linear objective function with $C_\ell = \lfloor \sum_{j \in J} b_{\ell j} p_j / D \rfloor$, and 3 resources.

	m5		**m10**		**m20**		**avg**	
	BB	*CPX*	*BB*	*CPX*	*BB*	*CPX*	*BB*	*CPX*
t10	2.31%	1.51%	0.85%	–	0.24%	–	**1.13%**	–
t15	3.77%	–	1.20%	–	0.60%	–	**1.86%**	–
t20	4.31%	–	2.71%	–	0.37%	–	**2.46%**	–
avg	**3.46%**	–	**1.59%**	–	**0.40%**	–	**1.82%**	–

Table 3: Average optimality gap for the quadratic objective function with $C_\ell = 0$, and 3 resources.

The average optimality gap (defined as $UB/LB - 1$, expressed in percents) for each algorithm is shown in Table 2 and Table 3 for the linear and quadratic objective function, respectively. A CPU time limit of 1800 seconds was imposed for each test run. For the test instances with quadratic objective function, CPLEX was only able to compute lower or upper bounds for the smallest instances within this time limit.

By analyzing the results, it can be clearly seen that our method overperforms the commercial solver on bigger test instances for the linear objective function, while it gives provably good quality solutions when using the quadratic objective function, in which case the commercial solver has difficulties in finding any lower or upper bound.

2.5 Publications

The results of Thesis 2 were published in (Drótos and Kis; 2011; Drótos et al.; 2007b; Drótos and Kis; 2008, 2010b).

3 A framework for solving complex industrial scheduling problems

3.1 Motivations

Although the topic of shop scheduling is very well studied, most approaches focus on simplified models. The main advantage of a simple, relaxed formulation is that it captures the important structure of the problem without including the less interesting technical details, but in a real industrial environment it may not be sufficient. Therefore it is very important to provide solution techniques that can be applied in a wide variety of production environments.

Our goal was to provide a customizable, modular framework that can be used in job-shop like environments with constraints such as limited renewable and non-renewable resource availability, machine and sequence dependent setup times, job hierarchy, etc., while optimizing multiple objective functions. An important aspect of the problem is that the solutions should satisfy the human decision-makers, i.e. the generated schedules should outperform those produced by hand.

3.2 Background and related results

A scheduling system not only consists of the actual algorithms that solve the problem, but it's a part of a business information system. As such, it is very important to follow guidelines that allow the successful development and integration of customized manufacturing scheduling systems. Framinan and Ruiz (2010) provide an overview of the requirements of such systems based on the literature of various fields from management to scheduling.

Job shop scheduling is a widely studied area of research. Some papers that consider similar extensions that are used in our work are e.g. (Schutten; 1998; Mason et al.; 2005; Mönch et al.; 2007). The main reasons that the existing, commercially available solutions (e.g. scheduling software for SAP) aren't sufficient in real life make-to-order manufacturing are (1) the renewable and non-renewable cumulative resources aren't handled simultaneously with machine constraints and (2) they may require a specific problem structure, e.g. parallel groups of machines with equivalent capabilities. On the other hand, scientific publications are generally concerned with specific, well-formed problems, therefore aren't always adaptable for other models.

Our solution approach relies heavily on heuristic procedures. An overview of these kinds of methods and the justification of their importance in practical optimization problems are given in e.g. (Silver; 2004).

3.3 Results

We provide a multi-phase algorithmic framework that can be applied for a wide variety of scheduling problems. We discuss various approaches that can be applied depending on the required features. Here we describe the main components of our approach. In each phase as more and more constraints are added to the problem, more and more parameters are fixed, allowing the algorithms to focus on their primary objectives while respecting the decisions made on previous phases.

Algorithm 3.1. *The proposed three phase framework for solving complex industrial scheduling problems consists of the following phases:*

1. *Making fundamental decisions and lower bound calculation*
2. *Providing a schedule optimized for the most important criteria*
3. *Adjusting the schedule to meet all requirements*

Algorithm 3.2. *An algorithm for assigning the non-renewable resources (materials) for problems with job priorities.*

Algorithm 3.3. *An algorithm for the fast recalculation of the start and end time of jobs in a job-shop like environment with certain properties is proposed, that can be used in local search heuristics.*

Algorithm 3.4. *An algorithm for assigning the renewable cumulative resources is proposed for environments where resource or location hierarchy has to be considered.*

Algorithm 3.5. *An algorithm for enforcing the renewable cumulative resource constraints is proposed in environments where the possible choices for doing this are to turn shifts on or off on machines.*

Systems based on this framework were implemented and deployed at GE Hungary Zrt. – Consumer & Industrial, Nagykanizsa and Bosch Rexroth Pneumatika Kft., Eger.

3.3.1 Phase I: Making fundamental decisions and lower bound calculation

In a wide variety of industrial problems, there are certain decisions that influence even the basic data of the underlying scheduling problem. A common example is a scenario where jobs may have several routing alternatives, and the choice of the routing alternative determines the actual operations that have to be carried out to produce the desired product. A series of bad choices could lead to a final bad solution that can't be improved by simple alterations of the schedule, while exploring other choices would introduce prohibitively high complexity to the algorithm.

Generally, it is possible to build a relaxed mathematical model small enough to be solved by standard solvers. The relaxation may even discard otherwise very important aspects of the original problems, e.g. in an actual application precedence constraints, setups, etc. are relaxed. There are two very important advantages of this approach:

- the highest level decisions are made and fixed based on an optimization process, so simpler models can be used in later phases without compromising the solution quality;
- the solution for the relaxed problem provides lower bound(s) for the objective function(s), so it will be possible the determine the quality of the final solution.

An other area where initial decisions can be safely made is material assignment. Problems that require non-renewable resource assignment and have a complicated machine environment (such as a flow shop or job shop) at the same time, can be prohibitively hard to solve. We overcome this problem by taking care of the material constraints in this phase, so we have proposed a method for the assignment of the non-renewable resources (materials) that provides good results in common industrial scenarios. As in most environments the incoming material supplies are fixed and cannot be altered for a predefined period, a schedule is only feasible if it does not generate material shortage (any required materials can be ordered for dates after this period). Algorithm 3.2 determines the earliest start times (release dates) of jobs based on their priorities in a way that no resource shortage will occur in any schedule respecting these start time constraints.

3.3.2 Phase II: Providing a schedule optimized for the most important criteria

Based on the previous decisions, it is now possible to construct an initial feasible schedule. The most promising approach to optimize it is the use of a guided local search algorithm, such as tabu search. Its main advantage is that it always maintains a feasible schedule, and the final result will be locally optimal, which means that with small changes one can not improve the final schedule. This aspect is very important in order to convince the users of the system of the quality of the solution. Furthermore, a number of constraints can be easily enforced on the schedule with only minor modifications of the algorithm.

By the appropriate choice of the neighborhood and exploiting the structure of the problem, it is possible to speed up the search. The main idea of Algorithm 3.3 is that when altering only a small part of the schedule during a local search procedure, it is not necessary to recalculate all the start end times of the jobs.

In our work, we evaluate different search strategies related to tardiness objective functions.

If the cumulative resources must be dealt with in complex manner, such as if a resource or location hierarchy is present, then Algorithm 3.3 can not be used because any set of otherwise non-related jobs may be in a resource conflict. Therefore we propose a different approach: we provide a general resource model along with an algorithm (Algorithm 3.4) for calculating the start and end times of the jobs in the schedule.

3.3.3 Phase III: Adjusting the schedule to meet all requirements

In this phase we have a good schedule based on the most important criteria, but it is possible that it may be improved in other aspects without the significant loss of its quality. A typical secondary objective function can be setup related, as frequent setup of the machines cost considerable amount of money, but it is generally not as important as job tardiness.

The main idea is that in this phase job-machine assignment is fixed and a tolerance on the primary objective function is introduced, and further optimizations are carried out while respecting these constraints. We provide a typical scenario where the elements of the previous local search algorithm can be used in this phase.

The other main topic of this phase is renewable cumulative resource handling. If the resource model is simple (such as that the available choice is to switch on or off shifts on machines or locations) then good results can be acquired with Algorithm 3.5. It calculates bounds with tools used in previous phases, and using these bounds, it iteratively formulates small, tractable integer programming problems. The solution of these problems determine which shifts have to be turned off in order to satisfy the resource constraints.

3.3.4 Computational experiments

We have conducted computational experiments on real industrial data as well as generated instances. On average, the problems had $\sim$130 machines and $\sim$3500 jobs. The most successful strategy provided results with total job tardiness on average at most 1.29 times the lower bound, while the other measures were also close to their optimal values.

3.4 Publications

The results of Thesis 3 were published in (Drótos et al.; 2009; Kis et al.; 2006; Drótos et al.; 2007a; Drótos and Kis; 2010a; Monostori et al.; 2008). Parts of these methods are covered in patent Nakano et al. (2012 – Japan and 2013 – WIPO).

References

Błażewicz, J. (2001). *Scheduling Computer and Manufacturing Processes*, 2^{nd} edn, Springer-Verlag, Berlin.
`http://www.google.hu/books?id=0rrq1xlJ1dsC`

Błażewicz, J., Lenstra, J. and Rinnooy Kan, A. (1983). Scheduling subject to resource constraints: classification and complexity, *Discrete Applied Mathematics* **5**(1): 11 – 24.
`http://www.sciencedirect.com/science/article/pii/0166218X83900124`

Boysen, N., Bock, S. and Fliedner, M. (2013). Scheduling of inventory releasing jobs to satisfy time-varying demand: an analysis of complexity, *Journal of Scheduling* **16**(2): 185–198.
`http://dx.doi.org/10.1007/s10951-012-0266-0`

Briskorn, D., Choi, B.-C., Lee, K., Leung, J. and Pinedo, M. (2010). Complexity of single machine scheduling subject to nonnegative inventory constraints, *European Journal of Operational Research* **207**(2): 605 – 619.
`http://www.sciencedirect.com/science/article/pii/S0377221710003929`

Burgess, A. R. and Killebrew, J. B. (1962). Variation of activity level on a cyclic arrow diagram, *Journal of Industrial Engineering* **13**: 76–83.

Carlier, J. and Pinson, E. (1994). Adjustment of heads and tails for the job-shop problem, *European Journal of Operational Research* **78**(2): 146 – 161.
`http://www.sciencedirect.com/science/article/pii/0377221794903794`

Demeulemeester, E. and Herroelen, W. (2002). *Project Scheduling: A Research Handbook*, International Series in Operations Research & Management Science, Kluwer Academic Publishers, Boston/Dordrecht/London.
`http://www.google.hu/books?id=pHetPl2LOYgC`

Framinan, J. M. and Ruiz, R. (2010). Architecture of manufacturing scheduling systems: Literature review and an integrated proposal, *European Journal of Operational Research* **205**(2): 237 – 246.
`http://www.sciencedirect.com/science/article/pii/S0377221709006419`

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

Grigoriev, A., Holthuijsen, M. and van de Klundert, J. (2005). Basic scheduling problems with raw material constraints, *Naval Research Logistics (NRL)* **52**(6): 527–535.
`http://dx.doi.org/10.1002/nav.20095`

Hall, L. A. and Shmoys, D. B. (1982). Jackson's rule for single-machine scheduling: making a good heuristic better, *Mathematics of Operations Research* **17**: 22–35.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem, *European Journal of Operational Research* **207**(1): 1 – 14.
`http://www.sciencedirect.com/science/article/pii/S0377221709008558`

Kellerer, H., Kotov, V., Rendl, F. and Woeginger, G. J. (1998). The stock size problem, *Operations Research* **46**(3): S1–S12.
`http://www.jstor.org/stable/3840850`

Martin, P. and Shmoys, D. (1996). A new approach to computing optimal schedules for the job-shop scheduling problem, *in* W. Cunningham, S. McCormick and M. Queyranne (eds), *Integer Programming and Combinatorial Optimization*, Vol. 1084 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 389–403.
`http://dx.doi.org/10.1007/3-540-61310-2_29`

Mason, S., Fowler, J., Carlyle, W. and Montgomery, D. (2005). Heuristics for minimizing total weighted tardiness in complex job shops, *International Journal of Production Research* **43**(10): 1943–1963.
`http://www.tandfonline.com/doi/abs/10.1080/00207540412331331399`

Mönch, L., Schabacker, R., Pabst, D. and Fowler, J. W. (2007). Genetic algorithm-based sub-problem solution procedures for a modified shifting bottleneck heuristic for complex job shops, *European Journal of Operational Research* **177**(3): 2100 – 2118.
`http://www.sciencedirect.com/science/article/pii/S0377221705008532`

Schutten, J. (1998). Practical job shop scheduling, *Annals of Operations Research* **83**: 161–178. 10.1023/A:1018955929512.
`http://dx.doi.org/10.1023/A:1018955929512`

Silver, E. A. (2004). An overview of heuristic solution methods, *The Journal of the Operational Research Society* **55**(9): 936–956.
`http://www.jstor.org/stable/4101930`

Torres, P. and Lopez, P. (2000). Overview and possible extensions of shaving techniques for job-shop problems, *CP-AI-OR '2000*, pp. 181–186.

Own publications

In international journals

Drótos, M., Erdős, G. and Kis, T. (2009). Computing lower and upper bounds for a large-scale industrial job shop scheduling problem, *European Journal of Operational Research* **197**(1): 296 – 306.
`http://www.sciencedirect.com/science/article/pii/S0377221708004700`

Drótos, M. and Kis, T. (2011). Resource leveling in a machine environment, *European Journal of Operational Research* **212**(1): 12 – 21.
`http://www.sciencedirect.com/science/article/pii/S037722171100097X`

Drótos, M. and Kis, T. (2013a). Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times, *Journal of Scheduling* **16**(3): 337–346.
`http://link.springer.com/article/10.1007%2Fs10951-012-0297-6`

In Hungarian journals

Monostori, L., Váncza, J., Kis, T., Kádár, B., Karnok, D., Drótos, M. and Egri, P. (2008). Valós időben együttműködő vállalatok: egy ipari-akadémiai projekt eredméyei, *Gépgyártás* **48**(4): 11 – 15.

In proceedings of international conferences

Drótos, M., Erdős, G. and Kis, T. (2007a). A lower bound and a tabu search algorithm for scheduling batches in a proportional job shop, *MAPSP 2007, 8th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Istanbul.

Drótos, M., Erdős, G. and Kis, T. (2007b). New techniques to handle workforce constraints in parallel machine scheduling, *MIM'07: IFAC workshop on manufacturing modelling*, Budapest, pp. 101–105.

Drótos, M., Györgyi, P. and Kis, T. (2013). Approximation algorithms for machine scheduling with non-renewable resources, *MAPSP 2013: 11th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Pont à Mousson.

Drótos, M. and Kis, T. (2008). Minimize overtime in a parallel machine environment, *VOCAL 2008: Veszprém Optimization Conference: Advanced Algorithms*, Veszprém.

Drótos, M. and Kis, T. (2010a). Challenges in planning and scheduling in discrete manufacturing and assembly, *Factory Automation 2010*, Kecskemét.

Drótos, M. and Kis, T. (2010b). Resource levelling in a parallel machine environment, *PMS2010: 12th International Conference on Project Management and Scheduling*, Tours, pp. 171–174.

Drótos, M. and Kis, T. (2013b). Scheduling of inventory releasing jobs, *EURO2013: 26th European Conference on Operational Research*, Rome.

Kis, T., Drótos, M., Erdős, G. and Kovács, A. (2006). The interval flow shop model, *PMS2006: Tenth International Conference on Project Management and Scheduling*, Poznan, pp. 217–221.

Patents

Nakano, T., Nonaka, Y., Drótos, M., Erdős, G., Kis, T., Kovács, A., Monostori, L. and Váncza, J. (2012 – Japan and 2013 – WIPO). Processing step planning device. App. number: 2012-054166 (Japan), PPCT/JP2013/056549 (WIPO).
`http://patentscope.wipo.int/search/en/WO2013137159`

Other

Drótos, M. (2012). *Algoritmuselmélet (BMEVISZA213) feladatgyűjtemény.*
`http://www.cs.bme.hu/~drotos/algfgy.pdf`