Decision Support and Optimization in Shutdown and Turnaround Scheduling

Nicole Megow*,‡ Rolf H. Möhring† Jens Schulz†

July 17, 2009

Abstract

Large-scale maintenance in industrial plants requires the entire shutdown of production units for disassembly, comprehensive inspection and renewal. It is an important process but causes high out-of-service cost. Therefore a good schedule for a shutdown and and an analysis of possible associated risks are crucial for the manufacturer.

We derive models and algorithms for shutdown scheduling that include different features such as time-cost tradeoff, precedence constraints, hiring external resources, resource leveling, different working shifts, and risk analysis. Our experimental results show that our methods solve large real-world instances very fast and yield an excellent resource utilization. A comparison with solutions of a mixed integer program on smaller instances proves the high quality of the schedules that our algorithms produce within a few minutes.

Our algorithms work in two phases. The first phase supports the manager in finding a good makespan for the shutdown. It computes an approximate project time cost tradeoff curve together with a stochastic evaluation of the risk for meeting a particular makespan t. Our risk measures are the expected tardiness at time t and the probability of completing the shutdown within time t. In the second, detailed planning phase, we solve the actual scheduling optimization problem for the makespan chosen in the first phase heuristically and compute a detailed schedule that respects all side constraints. Again, we complement this by computing upper bounds for the same two risk measures, but now for the detailed schedule.

The shutdown problem has many relationships with well established areas of scheduling, and we also give an overview on the large variety of scheduling problems involved.

Key words: project management; scheduling; resource leveling; time-cost tradeoff; resource constraints; working shifts; risk analysis

1 Introduction

In chemical manufacturing and petroleum refining, large-scale maintenance activities are conducted on a regular basis. Entire production units are shut down for disassembly, comprehensive inspection and renewal. Such a process is called *Shutdown and Turnaround* (or turnaround for short). It is an essential process but causes high out-of-service cost. Therefore a good schedule for the turnaround has a high priority to the manufacturer. A good schedule is not simply a short schedule. The project execution can be speeded up at the expense of adding resources, mostly in the form of additional workers. Thus, short projects cause high resource cost whereas cheap projects take a long time. Moreover, in practice, task execution times typically involve uncertainty. Such uncertainty arises due to unforeseen repair jobs, and, naturally, a short schedule is less robust against unexpected repair jobs or processing delays than a schedule with long duration that offers more flexibility for rescheduling. Such considerations are fundamental in the decision process of a turnaround project manager. We support this process by analyzing the tradeoff between project

---

*Max Planck Institute for Informatics, Saarbrücken, Germany, `nmegow@mpi-inf.mpg.de`.

†Institut für Mathematik, Technische Universität Berlin, Germany, `{moehring,jschulz}@math.tu-berlin.de`.

‡Partially supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

duration and project cost as well as the effects on the stability of schedules. Our main contribution is an optimization algorithm within a larger decision support framework that computes a detailed schedule of given project duration with the aim to minimize the total resource cost.

More explicitly, during a turnaround a huge number of precedence constrained operations or jobs must be executed by maintenance groups of different specializations. Scheduling these is already a complex task since various working shifts must be respected. However, in this particular problem another issue increases the complexity drastically: the duration of a job is flexible in the sense that a job can be accelerated by increasing the number of resource units (workers) allocated to it. Typically, technical reasons restrict the choice to a range between a maximum and minimum number of workers. We can assume that the duration of a job is a non-increasing discrete function of the number of workers allocated to it. Due to communication overhead between the workers, the duration of a job decreases at a smaller rate than the rate at which the number of assigned workers increases.

We will call workers with a particular specialization a *resource type* and an individual worker from such a group a *resource unit* of that type. Every job requires only resources from at most one type. Each resource unit causes a certain cost for each time unit it is used. In a turnaround project we may assume that the number of available resource units of any resource type is unlimited since resources can be hired externally. However, for most resource types resource units can only be hired for a certain minimum time period and must be paid for also when they are idle. That implies that the resource consumption of these resource types must be leveled over at least such a time period—in our model this is the entire turnaround period.

A feasible schedule consists of an allocation of resource units to jobs and a feasible temporal planning of jobs respecting given precedence constraints and working shifts. Ideally, we would like to minimize both, the project duration and its cost. However, there is the tradeoff mentioned above; fast project executions cause high cost whereas cheap project executions take a long time.

Determining a good project duration depends on several aspects that need to be balanced against each other. These are the total resource cost for hiring resources, the total production loss caused by the shutdown during the turnaround period, and a "risk cost" due to unexpected repairs and delays that are inherent in maintenance jobs and tend to become the more influential the more ambitious, i.e., the shorter the project duration is chosen. Let us neglect the risk cost for a moment. Then, for a given production loss per time unit, one would ideally like to find a schedule that minimizes the total cost, i.e., the sum of out-of-service cost and the cost for hiring resources over the turnaround period. If the out-of-service-cost cannot be quantified, than the manager defines a deadline for the turnaround and our goal is to find a feasible schedule of minimum resource cost that meets this deadline.

However, risk issues cannot be neglected in practice, in particular not in turnaround projects that contain many maintenance jobs that my cause unforeseen repair work or, even worse, delay the completion of the job until the delivery of a spare part. So the deadline for a turnaround can only be met with a certain probability which tends to decrease with an ambitious deadline. So information about the risk involved with a decision about the length of a turnaround is crucial to the manager. Fortunately, companies typically have stochastic information based on experiences with earlier turnaround projects. This information permits a stochastic evaluation of the risk of the computed schedule w.r.t. meeting the deadline. The risk measures we use are (i) the expected tardiness of a schedule w.r.t. the chosen deadline and (ii) the probability distribution of the project duration. Computing these measures is $\#P$-complete in general, which makes efficient computations unlikely. However, for problems without shifts and with unlimited resources we can apply known techniques to determine an upper bound on the expected tardiness for a given project schedule as a function of its completion time. Interestingly, the computation of this function is algorithmically strongly related to the algorithm we use to determine the deterministic tradeoff between project duration and project cost. Our stochastic evaluation of relevant schedules enables the project manager to decide for a particular schedule according to his own risk preferences.

In this paper, we introduce the problem of turnaround scheduling and develop techniques to solve such problems. We also give an overview on the large variety of related problems. As our main contribution, we report on our experience with solving real-world turnaround scheduling problems with our methods in a case study that was carried out in cooperation with the management consulting company T. A. Cook[1] and two of their customers at chemical manufacturing sites.

We implemented a two-phase solution method for turnaround problems that serves as a decision support tool. In the first phase, the *strategic planning phase*, a project manager has to decide on the desired makespan for the turnaround project and he has to quantify the number of workers and resources available for the project. To support this decision process, we provide an approximation of the tradeoff between project duration and cost as well as a stochastic evaluation of the risk for meeting the makespan. In a second phase, the *detailed planning phase*, we solve the actual scheduling optimization problem for the chosen deadline heuristically and compute a detailed schedule that we complement by evaluating upper bounds for the two risk measures expected tardiness and the probability of meeting the deadline.

Our methods can handle real-world instances with 100,000 – 150,000 jobs within a few minutes and yield solutions with a leveled resource consumption. To evaluate the performance of our methods, we compare our solutions with optimal solutions for problems with up to 50 jobs that are computed by solving a mixed integer program (MIP) formulation of the turnaround problem that includes all the deterministic features such as different shifts, variable resource allocation, resource leveling, and complex precedence constraints. Our MIP is time-indexed and thus much too large for typical problem sizes of turnarounds for chemical manufacturing. In contrast, our heuristic algorithm is fast and produces solutions of good quality as the comparison with the MIP shows.

To the best of our knowledge, this is the first time that the turnaround scheduling problem is treated with this combination of optimization techniques. The interest of our cooperation partner T. A. Cook in these methods has led to a commercial initiative to integrate them as a software tool into Microsoft Project.

2 Related work

Several variants of scheduling problems considered in the literature are related to the turnaround scheduling problem.

Time-cost tradeoff. Given a project network of jobs and precedence constraints, a job may be executed in different modes, each associated with a certain processing time and resource requirement. The *time-cost tradeoff problem* asks for the relation between the duration of a project and its cost, which is determined by the amount of non-renewable resources that is necessary to achieve the project duration. For a nice survey we refer to De et al. [6]. Fixing either the project duration or the cost leads to the closely related optimization problems with the objective to minimize the other parameter; these problems are referred to as the *deadline problem* and the *budget problem*, respectively.

When the resource cost for the jobs are continuous linear non-increasing functions of the job processing times, then the deadline and the budget problem can be solved optimally in polynomial time as has been shown independently by Fulkerson [15] and Kelley [21]. Later, Phillips and Dessouky [31] gave an improved version of the original algorithms in which iterative cut computations in a graph of critical jobs yield the project time-cost tradeoff curve which describes the tradeoff between project duration t and associated cost for all t. The running time is polynomial in the number of breakpoints of the optimal time-cost curve, which may, however be exponential in the input size, see Skutella [37]. (A breakpoint of such a piecewise linear function is a point in which the function is continuous but not differentiable.) Elmaghraby and Kamburowski [12] generalized previous algorithms to solve a more general problem variant in which jobs may have release dates and deadlines and arbitrary time lags between them. They provided a combinatorial algorithm that iteratively computes minimum cost flows in an iteratively transformed network modeling the time-lags. Also other cost functions have been considered such as convex [22, 20, 35, 1] and concave functions [13].

---

[1] T. A. Cook is a management consulting firm focusing on asset performance management; see `www.tacook.com`.

In practical applications, the discrete version of this problem plays an important role. Here the processing time of a job is a discrete non-increasing function of the amount of the renewable resource allocated to it. This problem is known to be $\mathcal{NP}$-hard [7]. Skutella [36] derived approximation algorithms for the deadline and budget problem as well as bicriteria approximations, while Deineko and Woeginger [8] gave lower bounds on the approximability of the problems.

Various exact algorithms and meta-heuristics have been implemented for the discrete time-cost tradeoff problem. For an overview we refer to the book by Demeulemeester and Herroelen [9, Chap. 8].

Time-cost tradeoff with capacity constraints. Motivated by restrictions on resource capacities in real-world applications, the time-cost tradeoff problem has been investigated in problem variants with renewable as well as non-renewable resources of limited capacity. Such problems are also known as *multi-mode (resource constrained) project scheduling problems*; see e.g. [9].

Various versions of linear and discrete time-cost tradeoff related problems have also been considered in the theory of machine scheduling . Besides the available machines, there is an additional resource that allows to accelerate the processing of jobs. Approximation algorithms and even polynomial time approximation schemes have been derived. Reviewing these results in detail is beyond the scope of this paper. We only mention scheduling with *controllable processing times* which concerns the allocation of non-renewable resources, see the recent survey of Shabtay and Steiner [34], scheduling jobs with *resource dependent processing times* which assumes a discrete renewable resource, see Grigoriev et al. [16] and references therein, and scheduling *malleable jobs* on one or several machines where the duration of a job is determined by the number of machines allocated to it, see e.g., Du and Leung [10], Lepére et al. [23] and Jansen and Zhang [19].

Resources with calendars and working shifts. In real world applications, resources are rarely continuously available for processing. Working shifts, machine maintenance or other constraints may prohibit the processing in certain time intervals. Also in machine scheduling, various problems with limited machine availability have been considered and we refer to the survey by Schmidt [33] for complexity and approximation results.

In project scheduling, such constraints are known as break calendars or shift calendars. Zhan [43] provides an exact pseudo-polynomial algorithm for computing earliest and latest start times in a generalized activity network that may contain minimum and maximum time lags, but no capacity bounds on the resources. His modified label-correcting algorithm respects jobs that may be preempted and those that must not. This algorithm has been modified into a polynomial time algorithm by Franck et al. [14]. In the same paper, they also provide priority-rule based heuristics for solving resource-constrained project scheduling problems where each job may require different resource types.

Yang and Chen [42] consider a job-based, more flexible version of calendars represented by *time-switch constraints*. These constraints specify for any job several time windows in which it may be processed. They also extend the classical critical path method in order to analyze project networks when resource capacities are unbounded. Time-switch constraints have also been incorporated by Vanhoucke et al. [40] in the deadline version of the discrete time-cost tradeoff problem in order to model different working shifts. They present a branch-and-bound algorithm which has later been improved by [38]. Experimental results were shown for instances with up to 30 jobs and up to 7 different processing modes. Recently, Vanhoucke and Debels [39] investigated the deadline problem with time-switch and other side constraints with the objective to minimize the net present value.

Resource leveling. Typical goals in project management are the minimization of the total project duration (makespan), the maximization of net present value or more service oriented goals such as minimizing waiting time or lateness. In certain applications the objective functions are based on resource utilization; see e.g., Neumann et al. [28]. In particular, when resources are rented for a fixed time period, then they should be utilized evenly over this time.

Harris [18] developed a critical path based heuristic for resource leveling of precedence constrained jobs with fixed processing times and no side constraints. Neumann and Zimmermann [29] presented a heuristic and exact algorithms for the resource leveling problem with temporal and resource constraints. In a number of earlier publications such as e.g. [4] and [11, 30, 3], heuristics and exact algorithms for simplified problem versions can be found.

Considering a variant of interval scheduling, Cieliebak et al. [5] aim for minimizing the maximum number of used resources. They derive approximation algorithms and hardness results.

Stochastic analysis of project networks. The importance of dealing with uncertainty in scheduling is reflected by the large number of publications on various aspects of this topic. These results mostly restrict to problems without resource constraints and without shift calendars. Several methods have been developed for analyzing the makespan $C_{\max}$ in project networks with random processing times, e.g., bounding the expected makespan or its distribution function. An exact computation of the makespan distribution—even just a single point of this function—is in general a $\#P$-complete problem, as shown by Hagstrom [17], which presumably rules out its efficient computation. For a general overview we refer to Adlakha and Kulkarni [2], and for a recent survey on methods for bounding the makespan distribution we refer to [26]. An experimental study comparing the performance of various such methods has been pursued by Ludwig et al. [24].

Particularly, we want to mention the works of Meilijson and Nadas [25] and Weiss [41]. They consider jobs with stochastically dependent processing times and determine an upper bound on the expected tardiness $\mathbb{E}[\max\{t - C_{\max}, 0\}]$ for a given project schedule with makespan $C_{\max}$ as a function of the completion time t (in the model without resource constraints and calendars). Interestingly, the computation of this bound is strongly related to solving a linear time-cost tradeoff problem.

3 Problem description

In turnaround scheduling we are given a set $\mathcal{J}$ of n jobs and a set $\mathcal{R}$ of renewable resource types. Each job needs a finite number of resource units of exactly one resource type $k \in \mathcal{R}$ for its processing.

Processing alternatives of job $j \in \mathcal{J}$ are characterized by the number r_j of allocated resources and its resulting processing time $p_j(r_j)$. We assume that r_j is integral and bounded from below and above by $r_j^{\min}$ and $r_j^{\max}$, respectively. Let $\mathcal{J}_k \subseteq \mathcal{J}$ denote the set of jobs that requires resource type $k \in \mathcal{R}$. Since each job requires exactly one resource type, we can partition the set of jobs $\mathcal{J}$ into disjoint subsets $\mathcal{J}_1, \mathcal{J}_2, \dots, \mathcal{J}_{|\mathcal{R}|}$.

The amount of work for processing a job $j \in \mathcal{J}$ is given by $w_j(r_j) = r_j \cdot p_j(r_j)$. We assume that the processing time is non-increasing and the work is non-decreasing in the number of resources. So the *monotonicity properties*

$$p_j(r_1) \geq p_j(r_2) \qquad \text{and} \qquad w_j(r_1) \leq w_j(r_2)$$

hold for any r_1, r_2 with $r_j^{\min} \leq r_1 \leq r_2 \leq r_j^{\max}$. We denote each processing alternative for a job $j \in \mathcal{J}$ given by the resource allocation r_j as a *mode* of job j and denote the set of feasible modes for job j by $\mathcal{M}_j$. So each feasible mode for job j defines a tuple (r_j, p_j) of allocated resources r_j and associated processing time p_j. Furthermore, each job $j \in \mathcal{J}$ has associated a release date $s_j \in \mathbb{Z}^+$ and a due date $d_j \in \mathbb{Z}^+$ which define the time-window $[s_j, d_j[$ in which j must be processed. Precedence constraints are given by a directed acyclic graph $G = (V, E)$ where the vertices correspond to jobs and there is an edge $(i, j) \in E$ if job i precedes job j. See the end of this chapter for generalized precedence constraints.

A vector of processing times $p = (p_1, \dots, p_n)$ is a *feasible realization* if for each $j = 1, \dots, n$ there is a resource allocation $r_j \in \{r_j^{\min}, \dots, r_j^{\max}\}$ such that $p_j = p_j(r_j)$. If the resource allocation r_j is clear from the context, we will simply write p_j instead of $p_j(r_j)$ and w_j instead of $w_j(r_j)$.

A *schedule* for a turnaround problem is given by a pair (S, r), where $r = (r_1, \dots, r_n)$ with $r_j \in \{r_j^{\min}, \dots, r_j^{\max}\}$ for each job j is a vector of feasible resource allocations, and S is a vector $S =$

$(S_1, \ldots, S_n)$ of start times for the jobs. A schedule (S, r) is *time-feasible* if it respects the release dates, due dates and precedence constraints, i.e.,

$$S_i + p_i \leq S_j, \qquad \text{for all } (i, j) \in E\,,$$

and

$$s_j \leq S_j \leq S_j + p_j \leq d_j, \qquad \text{for all } j \in \mathcal{J}.$$

The maximum completion time of all jobs, the *makespan*, is denoted by

$$C_{\max}(S, r) := \max_{j \in \mathcal{J}} \{S_j + p_j\}\,.$$

For a time-feasible schedule (S, r), we denote by

$$r_k(S, r, t) := \sum_{j \in J_k : S_j \leq t < S_j + p_j} r_j$$

the *resource utilization* of resource type $k \in \mathcal{R}$ at time t, and by

$$R_k := \max_t r_k(S, r, t), \qquad \text{for all } k \in \mathcal{R}$$

the *maximum resource utilization* of resource type $k \in \mathcal{R}$, i.e., the maximum number of resource units of type k utilized at any time during the turnaround. The maximum resource utilization of a resource type k may be bounded by a constant $\bar{R}_k \in \mathbb{Z}^+$, which we call the *capacity* of that resource type.

Each resource type $k \in \mathcal{R}$ has an individual *calendar* of working shifts, which represents the *availability periods* of k. To model calendars, we introduce an indicator variable δ_{kt} for any resource type k that is set to 1 if k is available at time $[t, t+1[$ and 0 otherwise. The condition

$$p_j = \sum_{t=S_j}^{S_j+p_j-1} \delta_{kt}$$

then expresses that no job is running in different availability periods. Given a project deadline T, we define the *working time* T_k of resource type k as the total time that resources of type $k \in \mathcal{R}$ are available, i.e.,

$$T_k := \sum_{t=1}^{T} \delta_{kt}.$$

We call a schedule *resource-feasible* w.r.t. calendars and bounded resource capacities $\bar{R}_k$ if

$$R_k \leq \bar{R}_k \qquad \text{and} \qquad p_j = \sum_{t=S_j}^{S_j+p_j-1} \delta_{kt}, \qquad \text{for all } k \in \mathcal{R},\ j \in \mathcal{J}_k.$$

For each $k \in \mathcal{R}$, we are given a cost rate c_k that represents the cost per unit of resource type k per time unit. The set of resource types is partitioned into two disjoint subsets $\mathcal{R}^\ell$ and $\mathcal{R}^f$ depending on their payment type. Resources of type $k \in \mathcal{R}^\ell$ have to be paid during the entire turnaround period for the maximum amount needed. These are mainly external workers that are hired for the complete turnaround period and they must be paid for even if they are temporally idle. Clearly, the goal of a project manager is to minimize the amount of paid idle time. In other words, the maximum resource utilization of those resource types should be minimized. We say that these resource types shall be *leveled*. In contrast, resource types from set $\mathcal{R}^f$ are paid for the actual work they perform. We say they are *free of leveling*. The job sets corresponding to $\mathcal{R}^\ell$ and $\mathcal{R}^f$ are denoted by $\mathcal{J}^\ell$ and $\mathcal{J}^f$, respectively.

Now, we can express the total cost of a schedule (S, r) as

$$\sum_{k \in \mathcal{R}^\ell} c_k \cdot R_k \cdot T_k + \sum_{k \in \mathcal{R}^f} \sum_{j \in J_k} c_k \cdot r_j \cdot p_j .$$

The first term is called the *resource availability cost* and represents the cost of resource types that must be leveled, while the second term, called *resource utilization cost*, represents the cost of jobs that do not need to be leveled.

The turnaround scheduling task is to find a schedule which is *time-* and *resource-feasible* and has minimum total cost. In practice the first term clearly dominates the cost function. If we neglect the cost for jobs that do not need to be leveled, we speak of the *resource leveling problem.* This is the problem on which we focus in this paper.

For later use, we introduce two additional parameters that are related to the goal of minimizing the resource availability cost. The first parameter indicates how “well” a single resource k is leveled and is called the *relative resource consumption* of resource type k and denoted by ν_k. It is defined for a schedule (S, r) as the total work done by resource type k relative to the maximum resource utilization R_k over the total available working time of resource type k, i.e.,

$$\nu_k := \frac{\sum_{j \in J_k} w_j}{T_k \cdot R_k} . \tag{1}$$

The second additional parameter μ_k is very useful within the resource leveling algorithm itself when we need to identify a resource type that is “badly” leveled and causes high cost. We call it the *relative idleness cost* μ_k and define it as the cost for the wasted available but not utilized work volume over the available work volume, i.e.,

$$\mu_k := (1 - \nu_k) \cdot c_k . \tag{2}$$

In the following we present a mixed integer programming formulation of the turnaround scheduling problem for a given project deadline T. It borrows from the classical time-indexed formulation based on start times for resource-constrained project scheduling by Pritsker, Watters, and Wolfe [32] and incorporates the multi-mode characteristics of jobs [?]. We use binary decision variables $x_{j\ell t}$ that indicate whether job $j \in \mathcal{J}$ starts in mode $\ell \in \mathcal{M}_j$ at time $t \in \{0, 1, \ldots, T-1\}$.

We model resource calendars implicitly using start-time dependent processing times. In a preprocessing step we compute for each job $j \in \mathcal{J}$ the processing time $p_{j\ell t}$ it has when starting at time t in mode $\ell \in \mathcal{M}_j$. If a job cannot be scheduled with respect to calendars at time t, then we set $p_{j\ell t}$ to the project deadline T, otherwise $p_{j\ell t}$ is the processing time of job j in mode ℓ. The resource requirements of job j in mode ℓ is denoted by $r_{j\ell}$.

$$\min \quad \sum_{k \in \mathcal{R}} c_k \cdot R_k \cdot T_k$$

$$\text{s.t.} \quad \sum_{\ell \in \mathcal{M}_j} \sum_{t=0}^{T-1} x_{j\ell t} = 1 \qquad \forall\, j \in \mathcal{J} \tag{3}$$

$$\sum_{\ell \in \mathcal{M}_j} \sum_{t=0}^{T-1} t \cdot x_{j\ell t} - \sum_{\ell \in \mathcal{M}_j} \sum_{t=0}^{T-1} (t + p_{i\ell t}) \cdot x_{i\ell t} \geq 0 \qquad \forall\, (i,j) \in E \tag{4}$$

$$\sum_{\ell \in \mathcal{M}_j} \sum_{t=0}^{T-1} (t + p_{i\ell t}) \cdot x_{i\ell t} \leq T \qquad \forall\, j \in \mathcal{J} \tag{5}$$

$$\sum_{j \in \mathcal{J}} \sum_{\tau=0}^{t} \sum_{\ell \in \mathcal{M}_j} r_{j\ell} \cdot x_{j\ell\tau} \cdot \mathbb{1}_{\{\tau + p_{j\ell\tau} > t\}}(\tau) \leq R_k \qquad \forall\, k \in \mathcal{R},\ t = 0, \ldots, T-1 \tag{6}$$

$$0 \leq R_k \leq \bar{R}_k \qquad \forall\, k \in \mathcal{R} \tag{7}$$

$$x_{j\ell t} \in \{0,1\} \qquad \forall\, j \in \mathcal{J}, \ell \in \mathcal{M}_j, t = 0, \ldots, T-1$$

Constraint (3) assures that each job starts exactly once in one of its modes, and because of (4) every two jobs i, j respect the precedence constraints. Constraint (5) avoids that a job starts at a time t that is in conflict with the resource calendars. Recall, that $p_{j\ell t} = T$ for such conflicting t. Finally, inequalities (6) and (7) guarantee that the capacity constraints are met.

Due to the time expansion, time-indexed formulations for scheduling problems are usually hopeless for large problem instances. However, small instances can be solved using integer programming solvers such as CPLEX, and we can thus evaluate the performance of our algorithm by comparing the computed solution with an optimal solution for such instances.

We conclude this section with a remark on further, so-called *generalized precedence constraints* that are important in some practical turnaround problems. Our algorithms can handle these requirements, but we do elaborate on this in the present paper.

Such additional constraints are:

- *Fixed start times:* A job $j \in \mathcal{J}$ must start at its release date $S_j = s_j$.
- *Parallel sets:* Two jobs $i, j \in \mathcal{J}$ must be executed in parallel for the processing time of the shorter job. W.l.o.g. we assume that $p_i(r_i) < p_j(r_j)$. The condition is fulfilled if $S_i \geq S_j$ and $S_i + p_i(r_i) \leq S_j + p_j(r_j)$.
- *Forbidden sets:* Two jobs $i, j \in \mathcal{J}$ that form a forbidden set must not be executed in intersecting time intervals. This is expressed by the following condition $S_j \geq S_i + p_i(r_i)$ or $S_i \geq S_j + p_j(r_j)$.
- *Zero maximum finish-start time-lags:* Job j has to be executed immediately after the completion of job i, i.e., $S_j = S_i + p_i(r_i)$ must hold.
- *Zero maximum start-start time-lag:* Two jobs $i, j \in \mathcal{J}$ have to start at the same time $S_i = S_j$.

4 Solution methods

We implemented a two-phase solution method for turnaround scheduling problems that serves as a decision support tool. The general outline is as follows.

Phase I. We compute a time-cost tradeoff curve that provides the approximate cost for any possible choice of duration T for the turnaround. To this end, we relax the integrality of the resource usage, the resource capacities, and working shifts. Then we solve a linear time-cost tradeoff problem with time-windows to optimality. Finally, we apply a heuristic scaling technique in order to approximate the true costs and to compute associated feasible schedules. This includes computing job modes for every T that results from scaling the breakpoints of the linear time-cost tradeoff curve.

Based on that curve (and information about the risk involved, see below), the decision maker chooses a particular makespan for the turnaround duration. This may be fixed by the decision maker based on his or her risk and other preferences but could also be the result of minimizing the total cost when out-of-service costs are available. The job modes computed for the chosen makespan are the basis for the second phase.

Phase II. In this phase, we solve the actual turnaround scheduling problem with all side constraints for the turnaround duration chosen in the first phase. We determine feasible start times for all jobs and adjust the resource allocation such that the temporal unavailabilities of resources as well as the given deadline T are respected. We find a feasible schedule with a resource profile that is leveled over the project duration, i.e., a schedule with minimized resource availability cost.

Stochastic support. The decision making process of the user is supported in both phases by a risk analysis of the respective solutions. We estimate the expected tardiness of relevant schedules for a deadline T and the probability of meeting it. In the first phase, this is done for each schedule corresponding to a breakpoint of the (relaxed) time-cost tradeoff curve, and, in the second phase, for the final schedule after resource leveling. We complement the expected tardiness of a schedule with confidence intervals which tell the project manager how likely it is to meet certain project deadlines.

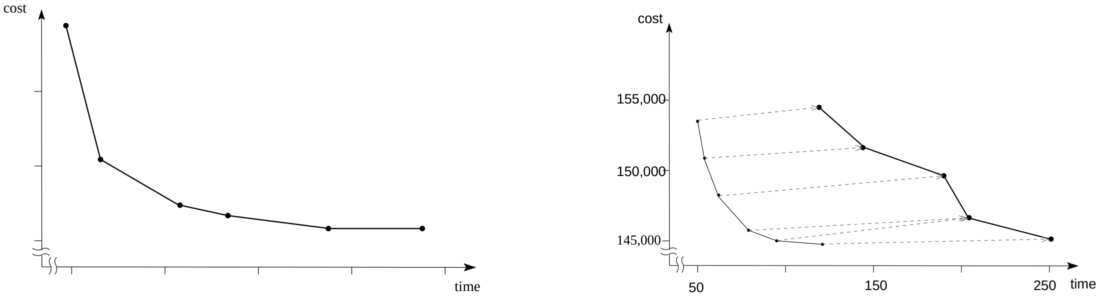
In the following sections, we describe the two main phases of our algorithm and the stochastic analysis in detail.

4.1 Phase I—The time-cost tradeoff curve

The tradeoff between project duration and project cost can be represented as a so-called *time-cost tradeoff curve.* For each possible project duration it gives the minimum cost. Such a curve can guide a manager when making the decision on the project deadline. Clearly, computing the exact curve is computationally intractable for a complex turnaround problem. However, an approximate curve is sufficient for decision support.

In this phase, we do not consider the resource availability cost as defined in Section 3. Instead, we compute the actual resource utilization cost without the additional cost for idle times when renting and leveling resources. The reason is that the cost of idle times is very sensitive to small changes in the schedule that occur only at project execution. Therefore, at this stage, when a project manager must decide on a project duration, he/she is interested only in the tradeoff between project duration and resource utilization cost, i.e., without the cost for idle times.

To compute such an approximate time-cost curve, we first compute an optimal curve for a relaxed problem and turn it into a feasible solution using rounding and scaling techniques. The general outline of the procedure is summarized below, followed by a detailed description. Figure 1 visualizes the effects of the procedure and the curves.



(a) Optimal solution for the linear time-cost tradeoff problem for a relaxation of the original turnaround problem.

(b) The optimal solution of the relaxation is turned into an approximate solution for the original turnaround problem by applying heuristics at the breakpoints. The time is given in hours and the cost is given in €.

Figure 1: Time-cost tradeoff curves.

```
Algorithm 1: Time-cost tradeoff algorithm with scaling
Input: Turnaround scheduling instance.
Output: Approximate time-cost tradeoff curve with respect to resource
        availability cost.
Compute the optimal time-cost tradeoff curve for the linearized problem version
without resource constraints (calendars, capacities).
for each point breakpoint $p$ of the curve do
    Round up fractional resource assignments to the nearest integral
    value $r_j := \lceil r_j \rceil$.
    Apply list scheduling heuristics that respect calendars and capacity constraints.
    Scale point $p$ and apply dominance rules.
```

First we relax our problem. We linearize the job modes $\mathcal{M}_j$ and allow non-integral resource allocations r_j. We convert the modes of each job into a linear function by linear interpolation. This cost function defines the dependency between processing time and cost. Furthermore, we assume that all resources are available continuously, and we do not level the resource usage. Then the problem reduces to assigning a (possibly non-integral) resource consumption r_j, and thus a processing time p_j, to each $j \in \mathcal{J}$, and finding a time feasible schedule of minimum resource utilization cost. This is the classical time-cost tradeoff problem with additional release dates and deadlines (time-windows) for jobs.

This problem without time-windows can be solved optimally in time that is polynomial in the input and the number of breakpoints of the curve [15, 21, 31]. In fact, most of our real-world instances do not require the more sophisticated and time consuming algorithm that respects also time windows by Elmaghraby and Kamburowski [12].

The optimal time-cost curve for the relaxed problem is a lower bound on the optimal time-cost curve for the tradeoff problem including all other side constraints.

Now, the schedules associated with points on the time-cost tradeoff curve need not be feasible. Usually, resource assignments are non-integral and resource calendars are not respected. In order to obtain a cost curve respecting these conditions, we consider all breakpoints T^i of the relaxed curve and turn the corresponding schedules into feasible schedules. Interpolation between the cost of these schedules then gives the new, approximate cost curve.

For a particular T^i, we round up a non-integral resource allocation r_j to the nearest integer. This increases the resource utilization cost, but it also guarantees that we do not exceed the job completion times from the linear relaxations. Once all jobs $j \in \mathcal{J}$ have integral values r_j, we greedily try to decrease the resource allocation and thus the cost without violating the deadline T^i.

Furthermore, schedules must be adopted to respect the given calendars. This will usually result in job deferrals and increased cost, but this way we obtain feasible solutions. We apply simple but fast list scheduling heuristics that re-schedule with respect to the resource availability (time and capacity wise). We refer to this as scaling a breakpoint T^i and the associated schedule. Within this heuristic approach, we may turn two infeasible schedules (S_1, r_1) and (S_2, r_2) with makespans $C_{\max}(S_1, r_1) > C_{\max}(S_2, r_2)$ and costs $\mathit{cost}(S_1, r_1) < \mathit{cost}(S_2, r_2)$ into two feasible schedules (S'_1, r'_1) and (S'_2, r'_2) with costs $\mathit{cost}(S'_1, r'_1) < \mathit{cost}(S'_2, r'_2)$ but with makespans $C_{\max}(S'_1, r'_1) < C_{\max}(S'_2, r'_2)$. In that case, schedule (S'_2, r'_2) is dominated by (S'_1, r'_1), and therefore, we do not store (S'_2, r'_2). The scaled time-cost tradeoff curve is obtained by interpolation between the points obtained after scaling and applying this dominance rule; see Figure 1(b). The difference between the linearized curve and the scaled curve obviously depends on the resource calendars and the resource capacities. In our real-world instances the project durations are scaled by a factor between two and three whereas the costs hardly differ.

The resulting curve is an approximation of the optimal time-cost tradeoff curve where the computed costs in the breakpoints are an upper bound for the optimal resource utilization cost. We guarantee that each point of the curve corresponds to a feasible schedule respecting all constraints. However, the resulting curve need not be convex anymore; see Figure 1(b).

While resource calendars and precedence constraints are considered, leveling the resource usage over time has not been addressed, yet. This objective is taken care of in the next phase when a project deadline is fixed. In the current stage, the approximated time-cost tradeoff curve together with the stochastic analysis as described in Section 4.3 guides a turnaround manager in its decision on the project duration T.

4.2 Phase II—Resource leveling and detailed scheduling

We enter the second phase with a maximum project duration T and a feasible choice of processing times $p^\star$ (with corresponding resource consumptions $r^\star_j$, for $j \in \mathcal{J}$) given by the chosen approximate solution of the first phase. While T remains fixed, the choice of job modes $(r^\star_j, p^\star_j)$ serves solely as a starting point for solving the resource leveling problem. We construct a schedule (S, r) that is time- and resource feasible and minimizes the resource availability cost $\sum_{k\in<<<<<<<.mine\mathcal{R}^\ell} c_k \cdot R_k \cdot T_k$.======= R$^\ell c_k \cdot R_k \cdot T_k$. In Figures 4 and 5 in Section 5 we illustrate the difference between a schedule with temporary high resource consumptions and a schedule with evenly used resource types that causes lower resource availability cost based on real turnaround data. ¿¿¿¿¿¿¿ .r6184

¡¡¡¡¡¡¡ .mine ======= ¿¿¿¿¿¿¿ .r6184

Our solution approach focuses on the minimization of the resource availability cost with respect to the feasibility of a schedule. To that end, we combine binary search on capacity bounds with different list scheduling procedures. Recall that only resource types $k \in \mathcal{R}^\ell$ need to be considered for leveling.

We compute initial lower and upper bounds, LB_k and UB_k, on the maximum resource utilization R_k for each resource type $k \in \mathcal{R}^\ell$. A lower bound is given by the minimum resource requirement of each job and by the minimum total workload divided by the working time of the corresponding resource. More formally,

$$R_k \geq \mathrm{LB}_k = \max\left\{ \max\{\, r_j^{\min} \,|\, j \in \mathcal{J}_k \}\,,\ \sum_{j\in\mathcal{J}_k} \frac{r_j^{\min} \cdot p_j(r_j^{\min})}{T_k} \right\}.$$

The upper bounds UB_k for $k \in \mathcal{R}$ might be part of the input, otherwise we compute an earliest start schedule (S, r) without limitations in the resource availability and use the resulting maximum resource utilization as upper bounds, i. e. $\mathrm{UB}_k := \max_t r_k(S, r, t)$.

In our algorithm we iteratively choose a resource type $k \in \mathcal{R}^\ell$ that is "badly leveled", meaning that a large fraction of the total availability cost is spent on idle times of resource type k with respect to its current bound UB_k. In Section 3 we introduced therefore the measure of relative idleness cost (2) for each resource type k depending on the maximum resource utilization R_k.

At this stage of the algorithm, the relative idleness cost is defined based on the current upper bound UB_k on the maximum resource utilization R_k, i.e., with a slight abuse of notation

$$\mu_k = \frac{T_k \cdot \mathrm{UB}_k - \sum_{j \in J_k} w_j(r_j)}{T_k \cdot \mathrm{UB}_k} \cdot c_k \, .$$

In each iteration, we choose a resource type $k^\star$ with maximum relative idleness cost, $\mu_{k^\star} = \max_{k \in \mathcal{R}^\ell} \mu_k$, and try to decrease the upper bound $\mathrm{UB}_{k^\star}$ on its capacity, while all other resource capacities remain fixed. We aim at decreasing $\mathrm{UB}_{k^\star}$ to $\alpha \cdot \mathrm{UB}_{k^\star} + (1-\alpha) \cdot \mathrm{LB}_{k^\star}$, for some $0 < \alpha < 1$. An upper bound can be decreased to a value u, if we find a time- and resource feasible schedule for the given total project duration T that utilizes at no time more than u units of resource type $k^\star$. We verify this property heuristically by using list scheduling procedures. Each of these procedures considers a different ordering (list) of jobs by which jobs are inserted into a partial schedule. With respect to the given precedence constraints, it is desirable to place jobs according to a topological ordering. Such orderings can be obtained by forward and backward computations of earliest start dates, earliest completion times, latest start dates and latest completion times of the jobs with respect to the given shifts and the given makespan T. We also use lists that have a random switch, i.e., the beginning of the list up to a randomly chosen position is sorted by increasing earliest start dates, while the jobs after that position are sorted by increasing latest completion times. We use five different such lists.

If the list scheduling procedures do not yield a feasible schedule, we find the lowest upper bound that allows a feasible schedule by binary search. If an upper bound UB_k cannot be decreased in this way, we do not consider resource type k for leveling anymore and set its lower bound to $\mathrm{LB}_k = \mathrm{UB}_k$.

Notice, that in this procedure we do not aim at decreasing one particular, badly leveled resource type immediately down to its lowest utilization limit. Instead, we consider all badly leveled resource types in a round robin fashion and decrease their utilization in each round to a certain fraction of the previous bound. The idea behind it is to balance the effects that the decreasing utilization of different resource types have on each other. Experiments have revealed that it is beneficial to focus not only on a single resource type but all of them one after another. The relative idleness cost μ_k steers the prioritization of resource types within the round robin selection.

A more formal description of our algorithm is as follows.

Algorithm 2: Resource Leveling

```
Input: Set $\mathcal{R}^\ell$ of resources that must be leveled, set $L$ of topologically sorted lists
       of jobs, maximum total project duration $T$, and parameter $\alpha \in (0,1)$.
Output: Leveled resource utilization $R_k$ for each resource type $k \in \mathcal{R}^\ell$.
1  Set $\mathrm{LB}_k$ and $\mathrm{UB}_k$ to initial values for each resource type $k \in \mathcal{R}^\ell$.
2  while $\exists\, k \in \mathcal{R}^\ell : \mathrm{LB}_k < \mathrm{UB}_k$ do
3      Choose resource type $k^\star \in \mathcal{R}^\ell$ with $\mathrm{LB}_{k^\star} < \mathrm{UB}_{k^\star}$ and $\mu_{k^\star}$ maximum.
4      Set a temporary upper bound $u_{k^\star} := \alpha \cdot \mathrm{UB}_{k^\star} + (1-\alpha) \cdot \mathrm{LB}_{k^\star}$.
5      for each list in L do
6          if List scheduling yields a feasible schedule with makespan at most T then
7              Set $\mathrm{UB}_{k^\star} := u_{k^\star}$.
8              goto Step 3.
9      // if List scheduling failed for each list:
10     Set $\mathrm{LB}_{k^\star} := u_{k^\star}$.
11     if $\mathrm{LB}_{k^\star} = \mathrm{UB}_{k^\star}$ then
12         $R_{k^\star} := \mathrm{LB}_{k^\star}$.
13         $\mathcal{R}^\ell := \mathcal{R}^\ell \setminus \{k^\star\}$.
14     else
15         goto Step 4.
```

At the end of Section 3 we have introduced additional, more complex *generalized* precedence relations between jobs. Our algorithm can handle these constraints: when inserting a job into a partial schedule during the list scheduling procedure (Step 6), this type of additional constraint can be respected. However, depending on their complexity, these constraints weaken the performance of the list scheduling heuristics and clearly slow down the computation process. In our real world instances, these constraints have mainly a common structure. In most cases they involve cranes that are required for a short time in parallel to a long job. It appears that cranes are no bottleneck resource in our instances and they do not need to be leveled. Therefore we can eliminate those precedence constraints in the resource leveling and handle them in a post-processing step without causing major conflicts. Similarly, we can substitute a group of jobs associated within generalized precedence relations by a single one if their time-windows and respective resource capacities are no bottleneck and reinsert them after the resource leveling procedure. In this way, generalized precedence constraints become rare and the increase in the running time is acceptable.

Another refinement of our list scheduling heuristic concerns the flexibility in the resource utilization and thus the influence on the processing times of jobs. Traditional list scheduling procedures deal with fixed realizations of processing times. Notice that so far we have only used the unchanged job modes as they were fixed in the first phase of the turnaround decision framework. Certainly, these first choices are based on an optimal solution for a relaxed problem version, and thus, they are a good starting point, but we would neglect optimization potential if we would keep them fixed. In particular, in such a complex scheduling situation as the turnaround problem, the "wrong" resource allocation for a single job may block the resource unfavorably and prohibit already a feasible solution for a certain resource capacity.

We address this issue by incorporating a local search on the resource allocation r_j of jobs $j \in \mathcal{J}$ into our list scheduling procedure. If a job j cannot be inserted because its current processing time exceeds the beginning of a non-available period of the respective resource for a small amount, then we increase the number of assigned resources r_j (if feasible), and thus, we decrease the processing time until the job eventually fits feasibly into the schedule. If this is not successful, we recursively include predecessors of j in this search. This refinement may speed up the resource leveling algorithm noticeably. The reason is that Step 5 and 6 of the binary search find a feasible solution much faster and accept a decrease of the upper bound on the capacity.

4.3 Risk analysis in Phase I and II

One of the essential features of turnaround scheduling is the fixed upper bound on the makespan T for the whole turnaround. This bound T determines the hiring of external resources, the loss of production of the factory unit that undergoes the turnaround, and also shifts of production to other factory units. It is therefore crucial to make a good decision about the makespan T in Phase I and to be aware of possible risks of the actual turnaround schedule computed in Phase II. Such risks are inherent in a turnaround project because of the many maintenance jobs that may involve unforeseen repair activities. So overambitious values of T will involve a high risk of exceeding T, while too pessimistic values will result in an unnecessary loss of production and possibly also higher external resource costs.

To aid the decision makers in choosing a good makespan T, we have implemented methods for evaluating two risk measures that are used in both phases. These are based on the assumption that the processing times of (some) jobs j are random variables X_j with (roughly) known probability distribution P_j and modal value p_j, where p_j is the deterministic processing time resulting from the given mode (r_j, p_j) of job j (either at a particular breakpoint of the time-cost tradeoff curve in Phase I, or the fixed mode of the final schedule in Phase II). For computational reason that will become obvious below, we also assume that the X_j are *discrete* random variables. They are so in our real world instances and have up to four values $p_j^1 < p_j^2 = p_j < pj^3 < p_j^4$ where p_j^1 denotes an early completion, and p_j^3 and p_j^4 model an increase in processing time due to repair work and waiting for spare parts plus repair, respectively.

The makespan $C_{\max}$ is then a function of the random processing times X_j and thus also a random variable that depends on the joint distribution P of the job processing times X_j. Since these are typically stochastically dependent (often with a positive correlation), an exact calculation of percentiles or other values of the distribution function of $C_{\max}$ is not feasible, as this is $\#P$-complete even for independent processing times [17]. We therefore compute worst-case measures that are valid for arbitrary dependencies among jobs.

The first such risk measure is an upper bound ψ on the expected tardiness of the makespan, which is defined as

$$\psi(t) := \sup_{Q:Q_j=P_j} \mathbb{E}_Q[\max\{C_{\max} - t, 0\}],$$

where Q ranges over all joint distributions of the processing times whose marginal distributions Q_j equals the given distribution P_j of X_j. So $\psi(t)$ is an upper bound on the expected time by which $C_{\max}$ will exceed the value t as a function of t.

This bound has extensively been investigated by Meilijson and Nadas [25] and Weiss [41]. As a function of t, $\psi(t)$ is convex decreasing with a slope of -1 for all t not exceeding a particular value t_0 and can, for all $t \geq t_0$ be computed as

$$\begin{array}{lcl} \psi(t) & = & \min_{(x_1,\ldots,x_n)} \sum_{j\in\mathcal{J}} \mathbb{E}[\max\{X_j - x_j, 0\}] \\ \text{s.t.} & & C_{\max}(x_1,\ldots,x_n) \leq t\,, \end{array}$$

where $C_{\max}(x_1,\ldots,x_n)$ is the makespan resulting from the processing times x_j of jobs j.

This minimization problem is the deadline version of a continuous time cost tradeoff problem in which the cost of job j as a function of its processing time x_j is just the expected tardiness $\mathbb{E}[\max\{X_j - x_j, 0\}]$ of that job and thus convex and even piecewise linear since the random processing times are discrete. These functions $\mathbb{E}[\max\{X_j - x_j, 0\}]$ are directly obtained from the values $p_j^1 < p_j^2 = p_j < pj^3 < p_j^4$ and their probabilities, and the standard algorithm for linear time cost tradeoff problems can straightforwardly be adapted to piecewise linear and convex cost functions. Altogether, this yields a very efficient computation of $\psi(t)$ as a function of t.

Meilijson and Nadas [25] also show that the bound $\psi(t)$ is tight in the sense that, for every $t \geq t_0$, there is a joint distribution Q such that $\psi(t) = \mathbb{E}_Q[\max\{C_{\max} - t, 0\}]$. In general, Q may depend on t, but if the precedence constraints form a series-parallel partial order (which is the case in our real life instances, see Section 5), then there is such a joint distribution Q attaining the worst case bound for all t. Moreover, the distribution function F of Q is then given by $F = 1 - \psi$. This implies that

$$\text{Prob}(C_{\max} \leq t) \geq F(t) = 1 - \psi(t)\,,$$

which means that the probability that the makespan $C_{\max}$ does not exceed time t is at least the value $1 - \psi(t)$, and that for all possible dependencies among jobs. This is exactly the second risk measure that we compute, and it can be directly obtained from the function $\psi(t)$.

Altogether, the solution of a continuous time cost tradeoff problem derived from the stochastic information about job processing times gives us directly two risk measures: the expected tardiness of exceeding the envisioned or chosen makespan T as $\psi(T)$, and the probability of exceeding it as $1 - \psi(T)$. Figure 2 shows the use of the second risk measure in Phase I, where the probabilities of exceeding an envisioned makespan T are indicated by different colors.

5 Computational results

In this section we report on our computational results obtained on the turnaround scheduling algorithm, that is, the resource leveling heuristic in Section 4.2 based on the project managers decision made after the time-cost tradeoff computation described in Section 4.1. We tested it on three real-world instances provided from chemical manufacturing sites and on additional smaller instances with similar characteristics that we have generated randomly. We evaluate the performance of the real-world instances by the means of the relative resource consumption ν_k which we introduced as equation (1) in Section 3 as an indicator for “how well” the resource is leveled. For a large class of artificially generated instances we can compare the resource availability cost of our

Figure 2: Evaluation of the risk in Phase I illustrated by a snapshot of our (German) software tool. The vertical percentage lines refer to the probability that the corresponding time t is met when the turnaround is carried out with the chosen value T. In the example, $T \approx 7$ days, and the probability of meeting T is only 45%.

heuristic solution with the cost of optimal solutions or lower bounds, both obtained by solving the mixed integer program (MIP) introduced in Section 3. All computations were done on a 64Bit Linux machine equipped with a 2.66GHz Intel Core 2 Duo processor with 2 GB RAM. To solve the integer programs we used ILOG CPLEX 11.

The real-world instances consist of about 1,000 and in one case 100,000 jobs. They require roughly 15 different resource types out of which 8 shall be leveled. The processing times per job are widely spread between 20 minutes and 2 days. The precedence relations between jobs form a series-parallel structure which is somewhat dominated by parallel chains. See Figure 3 for a visualization of such a structure for a part of a real-world input instance. To imitate a project managers decision on the total project duration (based on Phase I), we compute the time-cost tradeoff curve for a turnaround instance and choose a minimum, medium, and maximum feasible project duration. This yields three test instances per input instance to test the resource leveling heuristic. The chosen durations are between 4 and 15 days for the smaller instances.

We compute solutions for the huge real-world instance in less than 10 minutes, while solving the $1,000$ job instances takes only a few seconds. The relative resource consumption ν_k of the most important resource types in these examples lies between 95 and 99%. Other less costly resource types yield a lower relative resource consumption. (See Figures 4 and 5 for a visualization of the resource consumption of selected resource types before and after the leveling.) Nevertheless, this is a high degree of resource utilization, and the precedence constraints between jobs and resource calendars prohibit a much larger resource utilization. To quantify the optimality gap we consider instances of smaller size for which we can compute an optimal solution or obtain lower bounds from solving the MIP.

We generated such additional turnaround instances by randomly setting the parameters mainly within the bounds given by the real-world instances. We created test sets with 30, 40, 50 and 60 jobs and variable resource allocation of one or two resource units per job. The work volume of any job lies between 6 and 20. This is clearly a deviation from the real-world data we received, but this simplification enables us to compute optimal solutions. Additionally, we generated instances with six or seven resource units per job. To compare the results with the previous test sets we scaled the work volume for each job with a factor of 6.5. Each job requires one out of two resource types which must be leveled and have an upper bound on the resource capacity of 30 or 40. The resource

Figure 3: Series-parallel structure of precedence constraints between jobs; part of a real-world turnaround instance.

costs are chosen as in the real-world instances. The precedence relations are chosen randomly in such a way that the precedence graph is again series-parallel as in the real-world instances.

For each of these randomly generated instances, we determine deadlines, that is, turnaround makespans that a manager may choose in the same way as for the real-world instances. We compute for each instance the minimum and maximum project duration and a value in between, which gives three instances for the resource leveling heuristic. The total computation time of our algorithm is less than a second for each instance. We assess the quality of our solutions by comparing against CPLEX solutions for the corresponding MIP formulation stated in Section 3. As mentioned earlier, the model uses time-indexed variables, and is therefore not applicable to instances of the size as is typical in practice. For solving our smaller size test instances, we bound the computation time for CPLEX by one hour. If CPLEX has not found a provably optimal solution, then we use the current lower bound on the optimum value for the comparison with the results of our heuristic. Table 1 and Table 2 summarize the computational results for each of the two classes of instances, with resource requirements of one/two and six/seven units per job.

number of jobs	runtime heuristic	average runtime CPLEX	optimal solutions CPLEX	optimal solutions heuristic	gap heuristic vs. opt.	max. gap heuristic vs. opt.	gap heuristic vs. LB	max. cost gap heuristic vs. LB
30	< 1 sec	20 sec	100%	61%	8%	33%	–	–
40	< 1 sec	20 sec	100%	47%	10%	38%	–	–
50	< 1 sec	25 sec	100%	59%	7%	29%	–	–
60	< 1 sec	12 min	83%	40%	9%	33%	10%	33%

Table 1: Comparison of runtime and resource availability cost (or lower bounds) of optimal schedules obtained by CPLEX and schedules obtained by our heuristic. The number of resource units per job is between one and two.

As already mentioned, our heuristic solves all test instances in less than one second. In contrast, CPLEX needs for instances with sixty jobs on average nearly the total given time limit of one hour. The fourth column reveals that with increasing number of jobs CPLEX computations reach the given time limit, and thus, the computation process is aborted without having found an optimal solution. This situation occurs in particular if we set the resource allocation to 6 or 7 resource units. In a few cases, our heuristic actually found an optimal solution after a few seconds whereas CPLEX did not within one hour. The fifth column quantifies how often our heuristic yields provable optimal solutions. For resource allocations between one and two units, we solve a significant number of instances to optimality. This changes when the resource requirements

Figure 4: Visualization of the unleveled resource consumption for two selected resource types in a real world project, aggregated on a 4-hourly basis. The resource types are cleaners (on top, 100% = 3 units) and metal workers (bottom, 100% = 22 units), respectively.

number of jobs	runtime heuristic	average runtime CPLEX	optimal solutions CPLEX	optimal solutions heuristic	gap heuristic vs. opt.	max. gap heuristic vs. opt.	gap heuristic vs. LB	max. cost gap heuristic vs. LB
30	< 1 sec	20 min	68%	25%	12%	48%	15%	48%
40	< 1 sec	36 min	43%	0%	13%	24%	14%	30%
50	< 1 sec	36 min	41%	7%	16%	33%	17%	38%
60	< 1 sec	54 min	10%	3%	10%	19%	18%	36%

Table 2: Comparison of runtime and resource availability cost (or lower bounds) of optimal schedules obtained by CPLEX and schedules obtained by our heuristic. The number of resource units per job is between six and seven.

Figure 5: Visualization of the leveling for 4 selected resource types in a real world project, aggregated on a 4-hourly basis. The upper two res The upper two resource types are the same as in Figure 4, i.e., cleaners (3 units) and metal workers (22 units), respectively. The two resource types below correspond to two different cranes (14 ton and 22 ton) that are only used sporadically. Such resource types are excluded in the leveling.

increase to six and seven units. In those cases, in which a provably optimal solution is found (by CPLEX), we compare its cost with those of our heuristic. The sixth column shows that we obtain solutions that are close to optimum. We leave on average a gap of about $7-10\%$, and 38% in the worst case; see Table 1. Increasing the number of resource units yields somewhat worse results with an average gap of up to 16% and 48% in the worst case; see Table 2. The last columns in the tables compare the results of our heuristic to the lower bounds obtained by CPLEX. If CPLEX has solved all instances to optimality, we omit the last entries because they are equal to the sixth and seventh column. We compute the cost gap of our heuristic to the lower bound over all instances, also those where CPLEX found an optimal solution. In total this does not dramatically increase the average gap which shows that our solutions leave a gap of same size to the lower bounds as to the suboptimal solutions by CPLEX.

So far we have evaluated our heuristic on instances with up to two processing alternatives per job. Table 3 shows our computational results on instances where each job requires two, three or four resource units which are given in advance per job. The work volume per job lies between 20 and 50. The other parameters are equal to those of the instances before. With increasing number of jobs and also increasing project durations the number of optimal solutions found by CPLEX decreases and thus the number of proven optimal solutions found by our heuristic solutions also decreases. It turns out that the average optimality gap is again about 10%. The maximum gap has been 57% for a single instance.

number of jobs	runtime heuristic	average runtime CPLEX	optimal solutions CPLEX	optimal solutions heuristic	gap heuristic vs. opt.	max. gap heuristic vs. opt.	gap heuristic vs. LB	max. cost gap heuristic vs. LB
30	< 1 sec	29 sec	100%	43%	9%	31%	–	–
40	< 1 sec	19 min	73%	10%	11%	57%	14%	69%
50	< 1 sec	45 min	15%	0%	11%	17%	15%	25%

Table 3: Comparison of runtime and resource availability cost (or lower bounds) of optimal schedules obtained by CPLEX and schedules obtained by our heuristic. The number of resource units per job is fixed between two and four.

6 Conclusions and research perspectives

To the best of our knowledge, popular project management software does not support a time-cost tradeoff related analysis. Resource leveling packages do exist but seem to use very simple heuristics. Moreover, they have their limitations in the presence of working shifts, capacity bounds, or other specialized constraints such as conflicting job sets.

Motivated by applications in chemical manufacturing, we have formulated the shutdown and turnaround scheduling problem as an integrated problem containing various optimization problems as subproblems, such as the time-cost tradeoff problem, scheduling with resource capacities and working shifts, and resource leveling, which have been considered previously individually. We reported on our successful solution approach within a more comprehensive decision support tool that additionally provides tools for risk analysis during the decision process and for the final schedule. Our optimization algorithm yields near optimal solutions in a very short time. We hope that our work initiates more research on this general and integrated model to overcome the deficiencies of current project management tools.

Another very challenging line of research has come out of extensive discussions with practitioners about how to cope with uncertainty in turnaround scheduling. One question addresses the risk inherent in the whole planning process. So far, we have only implemented tools for risk evaluation of given schedules. We applied techniques to determine an upper bound on the expected tardiness and the probability of meeting the makespan for a given project schedule. This allows the project manager to choose a schedule according to his/her risk affinity. Nevertheless, this method is only applied after the schedule optimization. We expect that an integrated approach that combines risk analysis and scheduling yields scheduling policies as in e.g. Möhring, Radermacher and Weiss [27] and might even yield better decision support for turnaround projects.

Acknowledgments. We thank T.A. Cook Consultants for providing the historical data sets used to test our solution method, and for explaining the details of shutdown and turnaround scheduling projects. We also thank Eamonn Thorsten Coughlan, Elisabeth Günther and Ralf Hoffmann for their great support in implementing the algorithms into a software tool.

References

[1] S. Adel and S. Elmaghraby. Optimal linear approximation in project compression. *IIE Transactions*, 16(4):339–347, 1984.

[2] V. Adlakha and V. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966– 1987. *Information Systems and Operational Research*, 27(3):272–296, 1989.

[3] M. Bandelloni, M. Tucci, and R. Rinaldi. Optimal resource leveling using non-serial dynamic programming. *Journal of the Society for Industrial and Applied Mathematics*, 78(2):162–177, 1994.

[4] A. Bourges and J. Killebrew. Variation in activity level in a cyclical arrow diagram. *Journal of Industrial Engineering*, 13(2):76–83, 1962.

[5] M. Cieliebak, T. Erlebach, F. Hennecke, B. Weber, and P. Widmayer. Scheduling with release times and deadlines on a minimum number of machines. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science (TCS 2004)*, pages 217–230, 2004.

[6] P. De, E. Dunne, J. Ghosh, and C. Wells. The discrete time-cost tradeoff problem revisited. *European Journal of Operational Research*, 81:225–238, 1995.

[7] P. De, E. Dunne, J. Ghosh, and C. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research*, 45(2):302–306, 1997.

[8] V. Deineko and G. Woeginger. Hardness of approximation of the discrete time-cost tradeoff problem. *Operations Research Letters*, 29(5):207–210, 2001.

[9] E. Demeulemeester and W. Herroelen. *Project Scheduling: A Research Handbook*. Kluwer, 2002.

[10] J. Du and J.-T. Leung. Complexity of scheduling parallel task systems. *SIAM Journal on Discrete Mathematics*, 2(4):473–487, 1989.

[11] S. Easa. Resource leveling in construction by optimization. *Journal of Construction Engineering and Management*, 115(2):302–316, 1989.

[12] S. Elmaghraby and J. Kamburowski. The analysis of activity networks under generalized precedence relations (gprs). *Management Science*, 38(9):1245–1263, 1992.

[13] J. Falk and J. Horowitz. Critical path problems with concave cost-time curves. *Management Science*, 19:446–455, 1972.

[14] B. Franck, K. Neumann, and C. Schwindt. Project scheduling with calendars. *OR Spektrum*, 23:325–334, 2001.

[15] D. Fulkerson. A network flow computation for project cost curves. *Management Science*, 7:167–178, 1961.

[16] A. Grigoriev, M. Sviridenko, and M. Uetz. Machine scheduling with resource dependent processing times. *Mathematical Programming*, 110(1):209–228, 2007.

[17] J. Hagstrom. Computational complexity of PERT problems. *Networks*, 18:139–147, 1988.

[18] R. Harris. Packing method for resource leveling (PACK). *Journal of Construction Engineering and Management*, 116(2):331–350, 1990.

[19] K. Jansen and H. Zhang. An approximation algorithm for scheduling malleable tasks under general precedence constraints. *ACM Transactions on Algorithms*, 2(3):416–434, 2006.

[20] K. Kapur. An algorithm for the project cost/duration analysis problem with quadratic and convex cost functions. *IIE Transactions*, 5:314–332, 1973.

[21] J. Kelley. Critical-Path planning and scheduling: Mathematical basis. *Operations Research*, 9(3):296–320, 1961.

[22] L. Lamberson and R. Hocking. Optimum time compression in project scheduling. *Management Science*, 16(10):B597–B606, 1970.

[23] R. Lepére, D. Trystram, and G. Woeginger. Approximation algorithms for scheduling malleable tasks under precedence constraints. *International Journal of Foundations of Computer Science*, 13(4):613–627, 2002.

[24] A. Ludwig, R. Möhring, and F. Stork. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research*, 102:49–64, 2001.

[25] I. Meilijson and A. Nadas. Convex majorization with an application to the length of critical paths. *Journal of Applied Probability*, 16:671–677, 1979.

[26] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. In *Computational Discrete Mathematics*, volume 2122 of *Lecture Notes in Computer Science*, pages 79–97. Springer, 2001.

[27] R. Möhring, F. Radermacher, and G. Weiss. Stochastic scheduling problems I: General strategies. *ZOR – Zeitschrift für Operations Research*, 28:193–260, 1984.

[28] K. Neumann, C. Schwindt, and J. Zimmermann. *Project scheduling with time windows and scarce resources.* Springer, 2003.

[29] K. Neumann and J. Zimmermann. Procedures for resource leveling and net present value problems in projec scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127:425–443, 2000.

[30] D. Phillips and A. Garcia-Diaz. *Fundamentals of network Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[31] S. Phillips and M. Dessouky. Solving the project time/cost tradeoff problem using the minimal cut concept. *Management Science*, 24:393–400, 1977.

[32] A. Pritsker, L. Watters, and P. Wolfe. Multi project scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–108, 1969.

[33] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121:1–15, 2000.

[34] D. Shabtay and G. Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.

[35] N. Siemens and C. Gooding. Reducing project duration at minimum cost: A time-cost tradeoff algorithm. *OMEGA*, 3:569–581, 1975.

[36] M. Skutella. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, 23(4):909–929, 1998.

[37] M. Skutella. *Approximation and randomization in scheduling.* PhD thesis, Technische Universität Berlin, 1998.

[38] M. Vanhoucke. New computational results for the discrete time/cost trade-off problem with time-switch constraints. *European Journal of Operational Research*, 165(2):359–374, 2005.

[39] M. Vanhoucke and D. Debels. The discrete time/cost trade-off problem: extensions and heuristic procedures. *Journal of Scheduling*, 10(4–5):311–326, 2007.

[40] M. Vanhoucke, E. Demeulemeester, and W. Herroelen. Discrete time/cost trade-offs in project scheduling with time-switch constraints. *Journal of the Operational Research Society*, 53:1–11, 2002.

[41] G. Weiss. Stochastic bounds on distributions of optimal value functions with applications to PERT, network flows and reliability. *Operations Research*, 34:595–605, 1986.

[42] H.-H. Yang and Y.-L. Chen. Finding the critical path in an activity network with time-switch constraints. *European Journal of Operational Research*, 120(3):603–613, 2000.

[43] J. Zhan. Calendarization of time planning in MPM networks. *ZOR – Methods and Models for Operations Research*, 36(5):423–438, 1992.