Briskorn, Dirk; Leung, Joseph; Pinedo, Michael

Working Paper

Robust scheduling on a single machine usinge time buffers

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 639

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Suggested Citation: Briskorn, Dirk; Leung, Joseph; Pinedo, Michael (2008) : Robust scheduling on a single machine usinge time buffers, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 639, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/147557

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 639

Robust Scheduling on a Single Machine using Time Buffers

Dirk Briskorn[1,3,*], Joseph Leung[2], Michael Pinedo[3]

October 2008

[1]: Christian-Albrechts-Universität zu Kiel
Institut für Betriebswirtschaftslehre
Olshausenstr. 40, 24098 Kiel, Germany
http://www.bwl.uni-kiel.de/bwlinstitute/Prod
briskorn@bwl.uni-kiel.de

[2]: Department of Computer Science
New Jersey Institute of Technology
University Heights, Newark, NJ 07102, USA
http://web.njit.edu/~leung/
leung@cis.njit.edu

[3]: Stern School of Business
New York University
44 West 4^{th} Street, New York, NY 10012, USA
http://www.stern.nyu.edu/~mpinedo
mpinedo@stern.nyu.edu

*: supported by a fellowship within the Postdoc-Program
of the German Academic Exchange Service (DAAD)

Abstract

This paper studies the allocation of buffer times in a single machine environment. Buffer times are a common tool to protect the schedule against disruptions such as machine failures. We introduce new classes of robust machine scheduling problems. For an arbitrary scheduling problem $1|\beta|\gamma$, $prmt \notin \beta$, we obtain three corresponding robust problems: maximize overall (weighted) buffer time while ensuring a given schedule's performance (regarding γ), optimize the schedule's performance (regarding γ) while ensuring a given minimum overall (weighted) buffer time, and finding the trade off curve regarding both objectives.

We outline the relationships between the different classes of problems and the corresponding underlying problem. Furthermore, we analyze the robust counterparts of three fundamental problems.

Keywords: Single machine scheduling, robustness, buffer time allocation

1 Introduction

In recent years robust optimization has attained more and more attention. While there are lots of formal definitions of robust solutions to an optimization problem (and still there seems not to be one single approach) the basic idea can be put in an abstract way like this: Find a feasible solution to the optimization problem at hand that is not necessarily optimal but remains feasible and has a good performance if parameter values of the problem change. Depending on the specific problem the changes of parameters may be restricted to a given domain or to a subset of parameters. Furthermore, it may depend on the underlying problem to which extent the focus is on solutions that remain feasible or solutions that have good performance, respectively. Usually, a solution is called solution-robust if it remains feasible for certain changes in the problem's parameters. If, even more, the solutions performance does not change for certain changes, the solution is called quality-robust.

There are several reasons why robust solutions may be important. Data used in the optimization problem may be uncertain. Hence, while solving the problem we cannot consider the problem that turns out to be the challenge (or at least there is no guarantee that we do so). Moreover, realization of the solutions derived from the optimization problem may be uncertain and, thus, the realization may differ from the plan derived from the optimization problem. In both cases it adds considerably to the predictability of the result of the real world process if our solution is robust.

A vast amount of literature on robust optimization can be found, see Ben-Tal and Nemirovski [3, 4] and Kouvelis and Yu [15] for overviews. Most approaches of robust optimization belong to one of the following classes as proposed by Davenport and Beck [10]: First, redundancy based approaches aim at protecting solutions against uncertainty by using less resources or time slots than are available. In case of disruptions this surplus can be used to follow the actual plan. Second, probabilistic approaches are based on the analysis of probability and the extension of disruptions providing a tool to evaluate solutions' robustness. Third, contingent solution approaches come up with a set of solutions being alternatives chosen – and possibly being switched in between – during execution.

Robust machine scheduling has been considered in several papers, see Aytug et al. [2] for a recent survey. Daniels and Kouvelis [9] propose an approach based on worst-case analysis. Daniels and Carrillo [8] consider the problem to find the schedule that reaches a given performance (minimization of flow-time) with the maximum likelihood under uncertain processing

times. Kouvelis et al. [16] consider a two-machine flow shop where processing times are uncertain and the makespan should be minimized. These approaches are scenario based and aim at minimizing the regret of choosing a specific schedule caused by an other schedule performing better for a certain scenario.
Goren and Sabuncuoglu [11], Herroelen and Leus [13], Leon et al. [19], Leus and Herroelen [20, 21], Mehta and Uzsoy [22], and Wu et al. [26] employ probabilistic analysis to construct predictive schedules. Predictiveness here means that the expected deviation of the realized schedule to the intended one is minimized. The deviation is measured either in the schedule's performance or in the schedule's specification (e.g. start times of jobs).
Briand et al. [6] consider $1|r_i|L_{max}$ and find a set of solutions of equal performance which allow to switch from one solution to an other while processing. Therefore, it is possible to adapt the schedule to be executed online.
A specific redundancy based approach to protect schedules against disruptions is to insert time buffers between jobs. More specifically, a buffer between two jobs protects the start time of the latter job (and, therefore, its finishing time) against delays of the finishing time of the former one. Time buffer allocation has been studied in the context of project scheduling (e.g. in Al-Fawzan and Haouari [1], Kobylanski and Kuchta [14], Shi et al. [24], and Van de Vonder et al. [25]) as well as in the context of machine scheduling (e.g. in Leus and Herroelen [21]). Leus and Herroelen [21] discuss the problem to allocate buffer such that the sum of weighted expected deviation of start times is minimum when the makespan (and, therefore, the overall buffer time) is limited.
In this paper we consider a similar concept. However, we consider the minimum (weighted) buffer inserted between a pair of consecutive jobs on a single machine. The insertion of buffer times affects two opposed properties of the schedule. We reasonably assume that robustness is increasing if the inserted buffers are larger. However, the schedules performance (e.g. makespan) may then worsen. Informal speaking, this gives us two types of optimization problems:

- Given a required robustness (measured as minimum (weighted) buffer time) what is the best performance we can reach?
- Given a requirement for the schedule's performance what is the maximum robustness (hence the maximum minimum (weighted) buffer time) we can reach?

The remainder of the paper is organized as follows. In Section 2 we formalize a framework concerning buffer allocation for one machine problems. Section 3 focuses on performance optimization if a certain degree of robustness is required. The results obtained here are used in the following Section 4 as a tool box to obtain results for robustness maximization if a certain performance is required. We conclude the paper with a summary of our insights and an overview of future directions for research.

2 Problem Specification

In what follows we restrict ourselves to single machine environments without preemption of jobs. We consider robust counterparts of machine scheduling problems that can be classified according to the well-known notation introduced by Graham et al. [12] as $1|\beta|\gamma$, $\beta \subset \{prec, r_i, p_i = 1, p_j = p, d_i\}$, $\gamma \in \{f_{max}, \sum C_i, \sum w_i C_i, \sum T_i, \sum w_i T_i, \sum U_i, \sum w_i U_i\}$.
For a given schedule σ of n jobs let $\sigma(l)$, $1 \leq l \leq n$, denote the job in the lth position. We define the buffer time b_i related to job $i = \sigma(l)$, $1 \leq l \leq n-1$, as the amount of machine

idle time between jobs i and $\sigma(l+1)$. Hence $b_{\sigma(l)} = C_{\sigma(l+1)} - C_{\sigma(l)} - p_{\sigma(l+1)}$, $1 \leq l \leq n-1$. Furthermore, $b_{\sigma(n)} := 0$.
The main idea of a buffer time b_i between job i and its successor is to protect the starting time of i's successor. If completion of i is delayed by p_i^+ for whatever reason, then the starting time of its successor is not affected if $p_i^+ \leq b_i$. If $p_i^+ > b_i$ then the successor cannot start on time but still its starting time is delayed less than it would be without a buffer. Hence, time buffers are convenient because they do not only provide a certain degree of quality-robustness but first and foremost the robustness of each single starting time is enhanced. This may be important if we think of a single machine embedded in an JIT-environment, for example, where tools or material to process jobs are delivered right on time.
We propose three surrogate measures for robustness that have to be maximized:

- the minimum buffer time of schedule σ that is defined as $B_\sigma^m = \min_{1 \leq l < n} b_{\sigma(l)}$,
- the minimum relative buffer time of schedule σ that is defined as $B_\sigma^p = \min_{1 \leq l < n} \left(b_{\sigma(l)} / p_{\sigma(l)} \right)$, and
- the minimum weighted buffer time of schedule σ that is defined as $B_\sigma^w = \min_{1 \leq l < n} \left(b_{\sigma(l)} / w^b_{\sigma(l)} \right)$.

Each buffer protects the following jobs from disruption. Considering the minimum buffer time as a robustness measure is motivated by the idea to balance the degree of protection from disruption by the preceding job. If we can assume that the probabilities of jobs to be finished later than expected as well as the amount of delay are similar, then this minimum buffer time seems to be an appropriate surrogate measure of robustness.
Minimum relative buffer time is motivated by the assumption that the protection for the following job should be correlated with the processing time of the actual job. There may be two reasons for that. First, we may assume that procedures required to process jobs are more or less homogeneous and mainly differ in the amount of time necessary. Then, the risk of malfunctions at each point of time may be quite the same. However, a job that is processed for a longer period of time bears a higher risk of failure during its processing. Second, if we assume that procedures of jobs differ reasonably, then more complicated procedures (leading to larger processing times) may entail a higher probability of malfunction. In both cases it is appropriate to postulate a fixed relation between processing times and corresponding buffer as a surrogate measure.
In order to consider the individual risk for each job we propose a minimum weighted buffer time. Here, a buffer weight w_i^b is associated with each job i giving the planner an idea of how the buffers of different jobs should be related to each other. Of course, as by setting $w_i^b = 1$ or $w_i^b = p_i$ we can cover both cases mentioned above.
In each case no protection is necessary for disruptions caused by the last job and, hence, we exclude this job from the robustness measures. We want to emphasize that we are aware of the fact that the proposed measures seem to be methodically inferior to several other proposals in literature. However, the simplicity of the concept may imply opportunities to solve corresponding optimization problems while for many other robustness measures optimization problems are intractable even for the most simplest settings.
Moreover, the buffer insertion can be seen as an application of the robustness concept in Bertsimas and Sim [5] to machine scheduling. Bertsimas and Sim [5] consider robust solutions to a linear program. They assume that an interval for each parameter is known where its actual value is randomnly drawn from. Informally speaking, for given protection parameter Γ_k

for constraint k, the authors consider the set S_k of Γ_k parameters having the worst impact on feasibility of a solution if they deviate from the expected value. The authors then formulate the problem to find the optimal solutions that is feasible even if for each constraint k each parameter in S_k differs from the expected value to a maximum amount.
Although we cannot formulate most scheduling problems as a linear optimization problem we can apply this idea. Consider the following set of constraints where s_i and S_i is the starting time and the successor, respectively, of job i:

$$s_i + p_i \leq s_{S_i} \quad \forall i$$

Since we have only one parameter in each constraint there is no choice to be made which parameter affects feasibility of a solution most. Suppose that the realization of p_i is drawn from the interval $[\overline{p}_i - p_i^-, \overline{p}_i + p_i^+]$ where $\overline{p}$ is the expected value. Then, obviously, $\overline{p}_i + p_i^+$ is the worst case regarding feasibility of a solution. To protect the solution against this case we have to choose $s_{S_i} \geq s_i + \overline{p}_i + p_i^+$. Note that this can be seen as inserting buffer time $b_i \geq p_i^+$ to protect the start time of job S_i.
There is an other buffer time related surrogate measure for robustness of a schedule that has been mentioned in the literature, e.g. Al-Fawzan and Haouari [1], namely the sum of all buffer times. However, in accordance with the reasoning in Kobylanski and Kuchta [14] we refuse to follow this idea. If we consider total buffer time as a robustness measure, the distribution of total buffer time on single buffers is not concerned. Hence, a schedule having only a single buffer before the last job is considered as robust as a schedule where the same amount of total buffer time is evenly distributed on buffers between each pair of consecutive jobs. However, in the first schedule no job but the last one is protected against disruptions by preceding ones while in the second schedule each job is protected (to a certain amount).

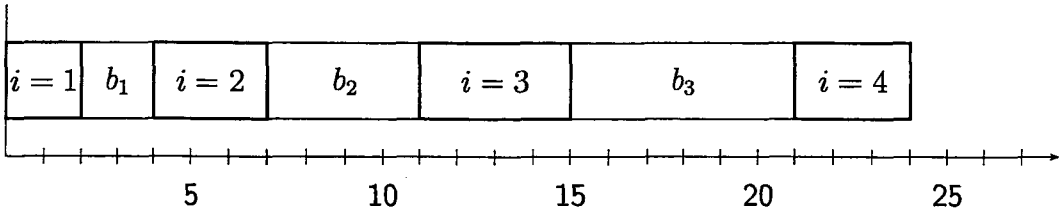

Figure 1: Schedule σ with Time Buffers

Figure 1 presents an arbitrary schedule σ of 4 jobs. Job 4 is scheduled last. We observe that $B_\sigma^m = 2$ since $\min_{1 \leq l < n} b_{\sigma(l)} = \min\{2, 4, 6\} = 2$. Furthermore, $B_\sigma^p = 1$ since

$$\min_{1 \leq l < n} \frac{b_{\sigma(l)}}{p_{\sigma(l)}} = \min\left\{1, \frac{4}{3}, \frac{6}{4}\right\} = 1.$$

To illustrate B_σ^w let us assume that buffer weights $w_1^b = 1$, $w_2^b = 5$, and $w_3^b = 6$ are given. Then, $B_\sigma^p = 0.8$ since

$$\min_{1 \leq l \leq n} \frac{b_{\sigma(l)}}{w_{\sigma(l)}^b} = \min\{2, 0.8, 1\} = 0.8.$$

For a given scheduling problem $1|\beta|\gamma$ we define $\mathcal{S}_{\underline{B}}^m$, $\mathcal{S}_{\underline{B}}^p$, and $\mathcal{S}_{\underline{B}}^w$ as the subsets of feasible schedules such that if and only if $\sigma \in \mathcal{S}_{\underline{B}}^m$, $\sigma \in \mathcal{S}_{\underline{B}}^p$, and $\sigma \in \mathcal{S}_{\underline{B}}^w$ we have $B_\sigma^m \geq \underline{B}$, $B_\sigma^p \geq \underline{B}$, and $B_\sigma^w \geq \underline{B}$, respectively. Additionally, we introduce $\mathcal{S}_{\overline{\gamma}}^\gamma$ as the subset of feasible jobs such that if and only $\sigma \in \mathcal{S}_{\overline{\gamma}}^\gamma$ the objective value of σ does not exceed $\overline{\gamma}$.

For the scheduling problems considered in the paper at hand, there is a trade off between robustness of a schedule and its performance. On the one hand, if we increase a schedule's robustness we may reduce its performance (e.g. the number of late jobs may go up). On the other hand, if we improve the schedule's performance (e.g. decrease the makespan) overall time buffer is reduced which may result into a reduced robustness. Consequently, we introduce robust counterparts concerning these trade-off effects for a given scheduling problem $1|\beta|\gamma$:

- Given a lower bound $\underline{B}$ of robustness find a schedule $\sigma \in \mathcal{S}^m_{\underline{B}}$, $\sigma \in \mathcal{S}^p_{\underline{B}}$, or $\sigma \in \mathcal{S}^w_{\underline{B}}$, respectively, that optimizes γ. We denote the corresponding problem by $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, and $1|\beta, B^w|\gamma$, respectively.
- Given an upper bound $\overline{\gamma}$ of performance find a schedule $\sigma \in \mathcal{S}^\gamma_{\overline{\gamma}}$ having $B^m_\sigma = \max\{B^m_\pi \mid \pi \in \mathcal{S}^\gamma_{\overline{\gamma}}\}$, $B^p_\sigma = \max\{B^p_\pi \mid \pi \in \mathcal{S}^\gamma_{\overline{\gamma}}\}$, and $B^w_\sigma = \max\{B^w_\pi \mid \pi \in \mathcal{S}^\gamma_{\overline{\gamma}}\}$, respectively. We denote the corresponding problem by $1|\beta, \gamma|B^m$, $1|\beta, \gamma|B^p$, and $1|\beta, \gamma|B^w$, respectively.
- Find the trade off curve according to objectives (i) minimize γ and (ii) maximize B^m_σ, B^p_σ, and B^w_σ, respectively. We denote the corresponding problem by $1|\beta|(\gamma, B^m)$, $1|\beta|(\gamma, B^p)$, and $1|\beta|(\gamma, B^w)$, respectively.

The motivation for these problem formulations can be derived from real world application. Suppose a planner can specify the robustness requirement (using the surrogate measures given above) of a schedule. Regarding this he still is interested in the best performance. Even in a single machine environment the sequence of jobs may significantly differ from the optimal sequence of the underlying problem (when no robustness measure is considered). The other way round, let us assume the planner has an upper bound on the performance of the schedule. If this upper bound exceeds the optimal performance without time buffers, the planner may want to use this surplus to protect the schedule against uncertainty. Last but not least, regarding the reasons above it is only natural to ask for the set of schedules being not dominated in a way that there is schedule better in one objective and at least as good in the other.

3 Performance Optimization for given Levels of Robustness

In this section we establish relations between an underlying scheduling problem $1|\beta|\gamma$ and its three robust scheduling counterparts $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, and $1|\beta, B^w|\gamma$. We distinguish between two classes of objective functions: *min sum* objectives (i.e. $\sum U_j$, $\sum w_j U_j$, $\sum T_j$, $\sum w_j T_j$, $\sum C_j$, $\sum w_j C_j$) and min max objectives (i.e. $f_{\max}$ or more specifically $C_{\max}$ and $L_{\max}$). For our reductions in this section we may have to modify processing times p_i, due dates d_i, and functions f_i:

- For min sum objectives we modify processing times and due dates (if given). The basic idea is to represent a job i and its corresponding buffer by job i' whose processing time comprises the processing time and buffer time of job i. However, when doing so we have to take care of the following situation. Job i''s part representing i's processing time may be finished before i's due date while the part representing the buffer is finished after d_i. In this case i' should be considered early and, thus, we have to modify due dates also.

- For min max objectives we do the same modifications as mentioned above. Additionally, we have to consider function f_i. Function f_i is a function of job i's completion time and, moreover, i's due date may be involved. So, in the first case we have to ensure that $f_{i'}(C_{i'}) = f_i(C_{i'} - (p_{i'} - p_i))$ and in the second case $f_{i'}(C_{i'}, d_{i'}) = f_i(C_{i'} - (p_{i'} - p_i), d_i)$ must hold. Basically, this says that the functions values for i and i' for the same starting points have to be the same.

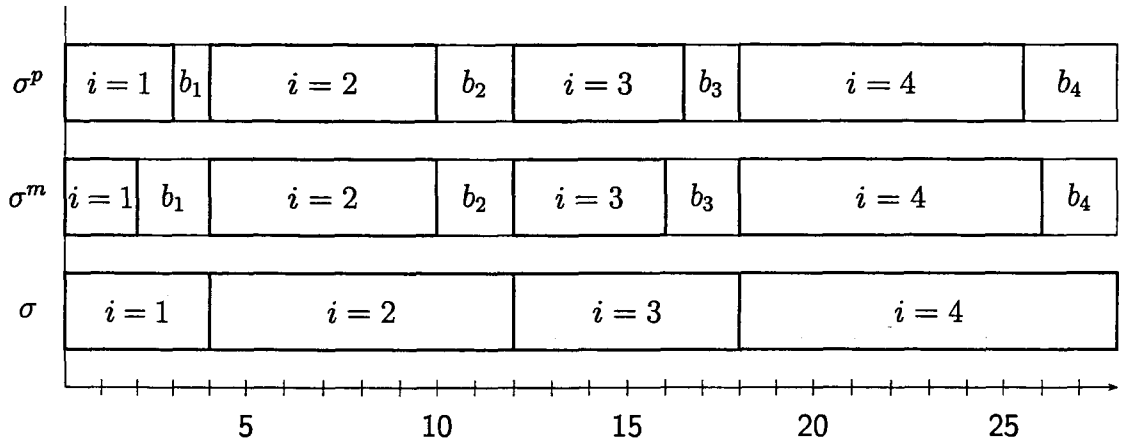

Figure 2: Sketch of reduction technique

Figure 2 presents three schedules that are related to one another. Schedule σ is the solution to an underlying scheduling problem having no time buffers. Schedule σ^m represents the corresponging solution to the robust counterpart considering minimum buffer time. In this case $\underline{B}^m = 1$. Schedule σ^p represents the corresponging solution to the robust counterpart considering minimum relative buffer time ($\underline{B}^m = 0.75$).
First, to provide some intuition for the basic technique used in all proofs in this section we determine as an example the complexity status of $1|p_j = p; r_j; B^m|\sum T_j$ and $1|p_j = p; r_j; B^w|\sum T_j$. Afterwards, we give proofs for more general results.

Lemma 1. $1|p_j = p; r_j; B^w|\sum T_j$ *is strongly* NP*-hard.*

Proof. We prove the complexity of $1|p_i = p; r_i; B^w|\sum T_i$ by reduction from $1|r_i|\sum T_i$ which, as a generalization of $1|r_i|L_{max}$, is known to be strongly NP-hard.
For a given instance P of $1|r_i|\sum T_j$ we construct an instance P^w as follows. We retain all parameters but p_i and d_i and set $p'_i = p' = \min_j\{p_j\}$ as well as $d'_i = d_i - p_i + p'$. Note that we obtain identical processing times. Furthermore, we set $w_i^b = (p_i - p')/p'$ and $\underline{B}^w = 1$.
We find an optimal solution σ to P from σ^w to P^w as follows. We set $C_i = C'_i - p' + p_i$. First, we obeserve that σ is feasible to P. Let i be an arbitrary job that is not scheduled last and let $s(i)$ be i's immediate successor. Then,

$$\begin{aligned} C_{s(i)} - C_i &= C'_{s(i)} - p' + p_{s(i)} - C'_i + p' - p_i \\ &\geq C'_i + p' w_i^b \underline{B}^w + p' - p' + p_{s(i)} - C'_i + p' - p_i \\ &= p_i - p' + p_{s(i)} + p' - p_i = p_{s(i)} \end{aligned} \tag{1}$$

Suppose there is a better schedule $\overline{\sigma}^w$ for P^w. We construct a better schedule $\overline{\sigma}$ for P by setting $C'_i = C_i + p' - p_i$. For corresponding solutions for P^w and P objective values are identical since

$$C_i - d_i = C'_i - p' + p_i - d'_i - p_i + p' = C'_i - d'_i.$$

Hence, $\overline{\sigma}$ is better than σ and σ can therefore not be optimal. □

In the following, we use the reduction techniques to determine relationships between underlying problems and their robust counterparts in a setting that is more general than the one in Lemma 1.

Theorem 1. *Problems $1|\beta, B^m|\gamma$ and $1|\beta, B^p|\gamma$ are equivalent even if $\underline{B}^m > 0$ and $\underline{B}^p > 0$.*

Proof. The proof is done in two steps: First, we reduce $1|\beta, B^m|\gamma$ to $1|\beta, B^p|\gamma$ and we do the reverse afterwards. We distinguish between two cases.

- **Case (i):** Processing times are arbitrary, that is $\beta \cap \{p_i = p, p_i = 1\} = \emptyset$.
- **Case (ii):** Processing times of all jobs are equal, that is $\{p_i = p\} \in \beta$ or $\{p_i = 1\} \in \beta$.

Let P^m be an arbitrary instance of $1|\beta, B^m|\gamma$. We construct an instance P^p of $1|\beta, B^p|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. In case (i) we set $p'_i = (p_i + \underline{B}^m)/2$, $d'_i = d_i + (\underline{B}^m - p_i)/2$, $f'_i(C'_i, d'_i) = f_i(C'_i - (\underline{B}^m - p_i)/2, d_i)$, and $\underline{B}^p = 1$. In case (ii) we set $p' = p$, $d'_i = d_i$, and $f'_i = f_i$ as well as $\underline{B}^p = (\underline{B}^m)/p$.
We derive a solution σ^m to P^m from an optimal solution σ^p to P^p by setting $C_i = C'_i + p'_i - \underline{B}^m$ in case (i) and $C_i = C'_i$ in case (ii).
Note that in case (i) we have $b'_i/p'_i \geq B^p_{\sigma^p} = 1$ for each $i \neq \sigma^p(n)$ while in case (ii) we have $b'_i/p'_i \geq B^p_{\sigma^p} = \underline{B}^m/p_i$ for each $i \neq \sigma^p(n)$. Now we can see that solution σ^m is feasible to P^m.
Let i be an arbitrary job that is not scheduled last and let $s(i)$ be i's immediate successor.
In case (i),

$$\begin{aligned} C_{s(i)} - C_i &= C'_{s(i)} + p'_{s(i)} - \underline{B}^m - C'_i - p'_i + \underline{B}^m \\ &\geq C'_i + p'_i + p'_{s(i)} + p'_{s(j)} - C'_i - p'_i \\ &= 2p'_{s(i)} = p_{s(i)} + \underline{B}^m \end{aligned} \tag{2}$$

while in case (ii),

$$\begin{aligned} C_{s(i)} - C_i &= C'_{s(i)} - C'_i \\ &\geq C'_i + p_i \frac{\underline{B}^m}{p_i} + p_{s(i)} - C'_i \\ &= p_{s(i)} + \underline{B}^m. \end{aligned} \tag{3}$$

Furthermore, in case (i)

$$\begin{aligned} C_i - d_i &= C'_i + p'_i - \underline{B}^m - d'_i + \frac{\underline{B}^m - p_i}{2} \\ &= C'_i + \frac{p_i + \underline{B}^m}{2} - \underline{B}^m - d'_i + \frac{\underline{B}^m - p_i}{2} \\ &= C'_i - d'_i \end{aligned} \tag{4}$$

while, trivially, $C_i - d_i = C'_i - d'_i$ in case (ii). Note that lateness $C_i - d_i$ may concern feasibility or performance. So, feasibility of σ^m is given. If the γ performance is based on lateness,

obviously the performances of σ^m and σ^p are identical. The same holds for general $f_{\max}$ due to the modification of f_i. Finally, if $\gamma \in \{\sum C_i, \sum w_i C_i\}$, solutions σ^m and σ^p have objective values that differ by a constant that does not depend on the solutions, that is

$$\begin{aligned} \gamma(\sigma^p) &= \sum_j w_j C'_j \\ &= \sum_j w_j C_j - \sum_j (p'_j - \underline{B}^m) w_j \\ &= \gamma(\sigma^m) - \sum_j (p'_j - \underline{B}^m) w_j \end{aligned} \tag{5}$$

in case (i) and $\gamma(\sigma^p) = \sum_j w_j C'_j = \sum_j w_j C_j = \gamma(\sigma^m)$ otherwise.
Suppose there is a solution $\overline{\sigma}^m$ that is better than σ^m. Then, we can construct a feasible solution $\overline{\sigma}^p$ by letting $C'_j = C_j - p'_j + \underline{B}^m$ in case (i) and $C'_j = C_j$ otherwise. Using (2), (3), (4), and (5) it is easy to show that $\overline{\sigma}^p$ is better than σ^p.
Now, let P^p be an arbitrary instance of $1|\beta, B^p|\gamma$. We construct an instance P^m of $1|\beta, B^m|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. In case (i), we set $\underline{B}^m = \min_i \{p_i\}$, $p'_i = p_i\,(1 + \underline{B}^p) - \underline{B}^m$, $d'_i = d_i + p_i \underline{B}^p - \underline{B}^m$, and $f'_i(C'_i, d'_i) = f_i(C'_i + \underline{B}^m - p_i \underline{B}^p, d_i)$. In case (ii), we set $p' = p$, $d'_i = d_i$, and $f'_i = f_i$ as well as $\underline{B}^p = \underline{B}^m / p$.
We derive a solution σ^p to P^p from an optimal solution σ^m to P^m by setting $C_i = C'_i + \underline{B}^m - p_i \underline{B}^p$ in case (i) and $C_i = C'_i$ otherwise. Based on the same arithmetics as in (2), (3), (4), and (5) it is easy to show that σ^p must be optimal for P^p. □

Theorem 2. *If not all processing times are equal to 1, then problems* $1|\beta|\gamma$, $1|\beta, B^m|\gamma$ *and* $1|\beta, B^p|\gamma$ *are equivalent even if* $\underline{B}^m > 0$ *and* $\underline{B}^p > 0$.

For proofs to Theorems 2 to 5 we only give reductions. All arithmetics to prove feasibility and optimality are analogous to (2), (3), (4), and (5). Regarding Theorem 1 we restrict ourselves to show that $1|\beta|\gamma$ and $1|\beta, B^m|\gamma$ are equivalent to prove Theorem 2.

Proof. First, we reduce $1|\beta, B^m|\gamma$ to $1|\beta|\gamma$. Let P^m be an arbitrary instance of $1|\beta, B^m|\gamma$. We construct an instance P of $1|\beta|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. We set $p'_i = p_i + \underline{B}^m$, $d'_i = d_i + \underline{B}^m$, and $f'_i(C'_i, d'_i) = f_i(C'_i - \underline{B}^m, d_i)$. We derive a solution σ^m to P^m from an optimal solution σ to P by setting $C_i = C'_i - \underline{B}^m$.
Now, we reduce $1|\beta|\gamma$ to $1|\beta, B^m|\gamma$. Let P be an arbitrary instance of $1|\beta|\gamma$. We construct an instance P^m of $1|\beta, B^m|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. We set $\underline{B}^m = (\min_i p_i)/2$, $p'_i = p_i - \underline{B}^m$, $d'_i = d_i - \underline{B}^m$, and $f'_i(C'_i, d'_i) = f_i(C'_i + \underline{B}^m, d_i)$. We derive a solution σ to P from an optimal solution σ^m to P^m by setting $C_i = C'_i + \underline{B}^m$. □

Theorem 3. *If* $p_i = 1$, *then problems* $1|\beta, B^p|\gamma$, $1|\beta, B^m|\gamma$, *and* $1|\beta'|\gamma$ *are equivalent when* β' *restricts all processing times to be equal (but not necessarily unit).*

Proof. The reduction of $1|\beta, B^m|\gamma$ to $1|\beta'|\gamma$ is analogous to the one in the proof of Theorem 2. Note that $p'_j = p$ holds for all processing times. The reduction of $1|\beta'|\gamma$ to $1|\beta, B^m|\gamma$ is done as follows. If $p < 1$, then we can multiply all processing times and – if considered – due dates and release dates by a constant k such that $kp \geq 1$. Then, we set $\underline{B}^m = p - 1$, $p'_i = p - \underline{B}^m = 1$, $d'_i = d_i - \underline{B}^m$, and $f'_i(C'_i, d'_i) = f_i(C'_i + \underline{B}^m, d_i)$. □

Theorem 4. *If processing times are arbitrary, then problems* $1|\beta|\gamma$, $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, *and* $1|\beta, B^w|\gamma$ *are equivalent even if* $\underline{B}^m > 0$ *and* $\underline{B}^p > 0$.

Regarding Theorem 2 we restrict ourselves to show that $1|\beta|\gamma$ and $1|\beta, B^w|\gamma$ are equivalent.

Proof. First, we reduce $1|\beta, B^w|\gamma$ to $1|\beta|\gamma$. Let P^w be an arbitrary instance of $1|\beta, B^w|\gamma$. We construct an instance P of $1|\beta|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. We set $p'_i = p_i(1 + w_i^b)$, $d'_i = d_i + p_i w_i^b$, and $f'_i(C'_i, d'_i) = f_i(C'_i - p_i w_i^b, d_i)$. We derive a solution σ^w to P^w from an optimal solution σ to P by setting $C_i = C'_i - p_i w_i^b$.
Note that $1|\beta, B^w|\gamma$ is a generalization of $1|\beta, B^p|\gamma$. Due to Theorem 2 $1|\beta|\gamma$ can be reduced to $1|\beta, B^w|\gamma$. □

Theorem 5. *If* $p_i = p$, *then problems* $1|\beta, B^w|\gamma$ *and* $1|\beta'|\gamma$ *are equivalent even if* $\underline{B}^w > 0$ *when* β' *implies arbitrary processing times.*

Proof. First, we reduce $1|\beta, B^w|\gamma$ to $1|\beta'|\gamma$ in analogy to the reduction in the proof to Theorem 4.
Now, we reduce $1|\beta'|\gamma$ to $1|\beta, B^w|\gamma$. Let P be an arbitrary instance of $1|\beta'|\gamma$. We construct an instance P^w of $1|\beta, B^w|\gamma$ as follows. We preserve all problem's characteristics but p_i, d_i, and f_i, $i \in \{1, \ldots, n\}$. We set $p'_i = (\min_i\{p_i\})/2$, $w_i^b = (p_i - p'_i)/p'_i$, $\underline{B}^m = 1$, $d'_i = d_i - p_i + p'_i$, and $f'_i(C'_i, d'_i) = f_i(C'_i + p_i - p'_i, d_i)$. Note that we have no restriction for p_i. We derive a solution σ to P from an optimal solution σ^m to P^m by setting $C_i = C'_i - p'_i + p_i$. □

Remark 1. *Using the techniques employed above it is easy to see that* $1|\beta, B^w|\gamma$ *and* $1|\beta', B^w|\gamma$, *where* β *and* β' *imply* $p_j = 1$ *and* $p_j = p$, *respectively, are equivalent.*

Summarizing Theorems 1 to 5, we can provide strong results for the computational complexity of the robust counterparts of the single machine scheduling problem $1|\beta|\gamma$ when the robustness is given.

- $1|\beta, B^m|\gamma$ and $1|\beta, B^p|\gamma$ are equivalent
- $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, and $1|\beta|\gamma$ are equivalent unless $p_j = 1$
- $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, and $1|\beta'|\gamma$ are equivalent for β implying unit processing times and β' implying equal (but not necessarily unit) processing times
- $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, $1|\beta, B^w|\gamma$, and $1|\beta|\gamma$ are equivalent unless $p_j = p$
- $1|\beta, B^m|\gamma$, $1|\beta, B^p|\gamma$, $1|\beta, B^p|\gamma$, and $1|\beta'|\gamma$ are equivalent for β implying equal processing times and β' implying arbitrary processing times

Figure 3 illustrates the reducability of the classes of problems considered in this section. Each node represents the class of problems specified by the structure of processing times corresponding to the column and the robustness measure corresponding to the row. Note that $1|\beta|\gamma$ represents the underlying problem. The label at each arrow outlines the Theorem providing the relationship represented by the arrow. Here, "Gen." and "Rem. 1" refer to a trivial generalization and Remark 1, respectively. The dashed lines partition the problem classes into three equivalence classes.
Hence, in most cases if the underlying problem is known to be solvable in polynomial time, we can solve the robust counterpart where robustness is given using the algorithms for the

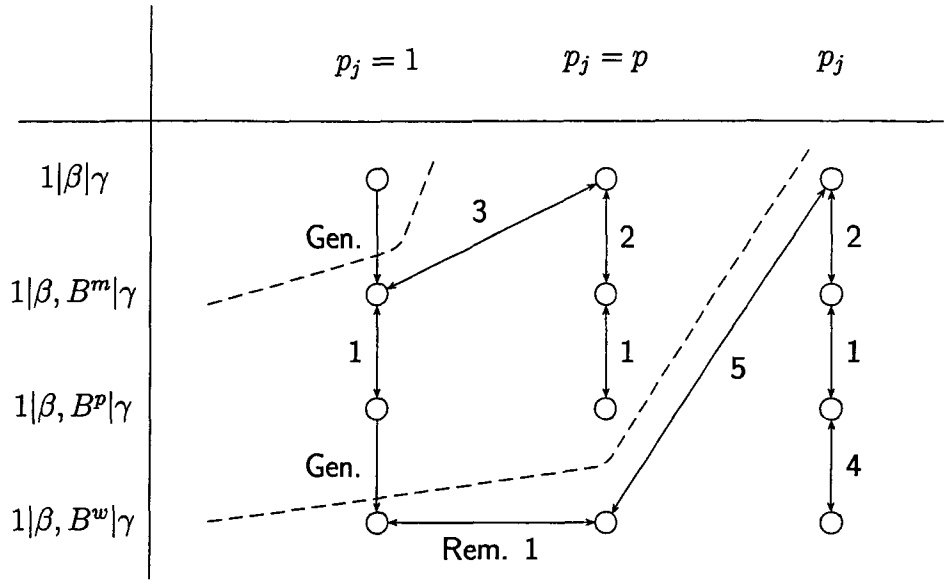

Figure 3: Equivalence Graph

P	P^w	P'
$1\|prec; p_i = p; r_i\|L_{max}$	$1\|prec; p_i = p; r_i; B^w\|L_{max}$	$1\|prec; r_i\|L_{max}$
$1\|prec; p_i = p; r_i\|\sum C_j$	$1\|prec; p_i = p; r_i; B^w\|\sum C_j$	$1\|prec; r_i\|\sum C_j$
$1\|p_i = p; r_i\|\sum w_j C_j$	$1\|p_i = p; r_i; B^w\|\sum w_j C_j$	$1\|r_i\|\sum w_j C_j$
$1\|p_i = p; r_i\|\sum w_j U_j$	$1\|p_i = p; r_i; B^w\|\sum w_j U_j$	$1\|r_i\|\sum w_j U_j$
$1\|p_i = p; r_i\|\sum T_j$	$1\|p_i = p; r_i; B^w\|\sum T_j$	$1\|r_i\|\sum T_j$

Table 1: Polynomial Problems with NP-hard Robust Counterparts

underlying problem. However, this does not hold in general. For underlying problems where processing times are restricted the processing time structure may get more general by the reduction mechanism, that is change from unit processing times to equal processing times or change from equal processing times to arbitrary processing times. Table 1 provides underlying problems in the left column that are maximum polynomially, see Brucker [7]. In the second column we have the strongly NP-hard robust counterpart P^w corresponding to the underlying problem P. P^w is (in each case) strongly NP-hard since it is equivalent to corresponding problem P' in the right column.
Although there may be some we could not find an underlying problem being polynomially solvable or binary NP-hard and having a binary NP-hard or unary NP-hard robust counterpart, respectively.

4 Robustness Optimization for a required Performance Level

In this section we consider the three robust counterparts $1|\beta, \gamma|B^m$, $1|\beta, \gamma|B^p$, and $1|\beta, \gamma|B^w$. The main issue is here that the order of jobs is not fixed but depends on the degree of robustness as we can illustrate with a rather simple example.
Figure 4 shows three optimal solutions σ_0, $\sigma_{0.5}$, and σ_1 corresponding to the same underlying

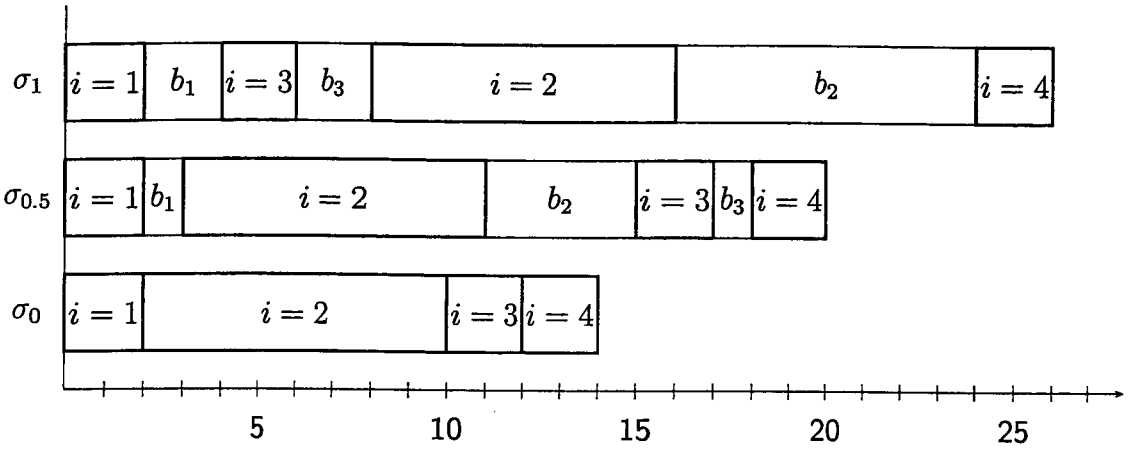

Figure 4: Changing sequence of jobs for $1||L_{max}$

problem instance of $1||L_{max}$ for $\underline{B}^p = 0$, $\underline{B}^p = 0.5$, and $\underline{B}^p = 1$, respectively. The problem instance itself is defined by $n = 4$, $p_1 = p_2 = p_4 = 2$, $p_3 = 8$, $d_1 = 2$, $d_2 = 11$, $d_3 = 15$, and $d_4 = 24$. Note that we can find an optimal solution by arranging a sequence of jobs in non-decreasing order of due dates. According to Theorem 2 this approach can be applied by sorting modified due dates $d'_i = d_i + p_i\underline{B}^p$. For $\underline{B}^p = 0.5$ this gives us $d'_1 = 3$, $d'_2 = 15$, $d'_3 = 16$, and $d'_4 = 25$ and, hence, we obtain a optimal sequence of $\sigma_{0.5} = (1, 2, 3, 4)$. Note that this sequence is the only optimal one. However, for $\underline{B}^p = 1$ this gives us $d'_1 = 4$, $d'_2 = 19$, $d'_3 = 17$, and $d'_4 = 26$ and, hence, the only optimal sequence is $\sigma_1 = (1, 3, 2, 4)$.

In what follows we analyze the robust counterparts of several basic scheduling problems that seek for the maximum robustness if a certain performance must be guaranteed. It is easy to see that the robust counterpart can not be easier than the underlying problem. That is why we restrict ourselves to underlying problems that are known to be solvable in polynomial time, namely $1|sp-graph|\sum w_iC_i$, $1||\sum U_i$, and $1|prec|f_{max}$.

Note that for all these cases the robust counterparts from Section 3 are equivalent to the underlying problem. Hence, as long as an upper bound on the robustness is given we can find a solution differing from the optimal one only by a constant ϵ in polynomial time using binary search on the robustness domain. Therefore, we focus on finding the exact solution or exploring the tradeoff curve between performance and robustness.

The strategy in all cases can be sketched as follows. We determine a set of values B^s for $\underline{B}^m$, $\underline{B}^p$, and $\underline{B}^w$ that mark sequence changing robustnesses for the problem, that is if $b \in B^s$, then the sequence of jobs optimizing performance for given $\underline{B}^w = b^- < b$ is different from the optimal sequence for given $\underline{B}^w = b^+ > b$. For $\underline{B}^w = b$ both sequences are optimal. Searching these intervals, we can determine the sequence of jobs for the maximum robustness. Given the sequence, finding the optimal robustness bolds down to easy arithmetics.

4.1 $1|sp-graph, \sum w_iC_i|B^w$

First, we consider underlying problem $1||\sum w_iC_i$ that is well studied and that serves well to gain insights that can be applied to more general problems. Afterwards, we tackle $1|sp-graph|\sum w_iC_i$.

4.1.1 $1|\sum w_iC_i|B^w$

It is well known that an optimal solution to this problem can be found by sorting jobs in non-increasing order of p_i/w_i. Regarding Theorem 4, $1|B^w|\sum w_iC_i$ can be solved by reducing it to $1||\sum w_iC_i$ where processing times are defined as $p'_i = p_i + w_i^b\underline{B}^w$. Thus, we derive the optimal order of jobs for given robusteness by sorting $(p_i + w_i^b\underline{B}^w)/w_i$. Furthermore, we can see the optimal performance as a non-decreasing function of the required robustness.
We determine

$$B^s = \left\{ b \mid b = \frac{p_iw_j - p_jw_i}{w_iw_j^b - w_jw_i^b} > 0, i < j \right\}$$

and sort this set according to non-decreasing values. Note that $|B_s| \in O(n^2)$. Let b_k, $k \in 1, \ldots, |B^s|$ be the kth value in B^s. Applying binary search on B^s, we can find the smallest value b_{k^*} in B^s such that the optimal solution value to $1|B^w|\sum w_iC_i$ with $\underline{B}^w = b_{k^*}$ exceeds $\overline{\gamma}$. Then, the sequence of jobs corresponding to the interval $[b_{k^*-1}, b_{k^*}]$ is the sequence of jobs for maximum robustness. For a given sequence σ of jobs we can determine the maximum robustness as a function of $\overline{\gamma}$:

$$B^w(\overline{\gamma}, \sigma) = \frac{\overline{\gamma} - \sum_i w_i \sum_{j \le i} p_j}{\sum_i w_i \sum_{j<i} w_j^b}$$

(we assume that jobs are numbered according to σ). The computational complexity of this procedure is $O(n^2 \log n)$.
Instead of finding the optimal robustness for a specific given $\overline{\gamma}$ we may be interested in finding the trade off curve between robustness and total weighted completion time. In the following, we give an algorithm to find the tradeoff curve represented by function $B^w(\overline{\gamma})$. The procedure resembles the one described above except for B^s being searched sequentially.

Algorithm 1

1. find and sort B^s
2. find the optimal sequence σ for $1||\sum w_iC_i$ and corresponding optimal solution value v_0^*
3. in $[0, v_0^*[$ the tradeoff curve is not defined
4. for $k = 1, \ldots, |B^s|$
   (a) find the optimal solution value v_k^* for $1|B^w|\sum w_iC_i$ where $\underline{B}^w = b_k$
   (b) in $[v_{k-1}^*, v_k^*[$ the tradeoff curve is given by linear function $B^w(\overline{\gamma}) = B^w(\overline{\gamma}, \sigma)$
   (c) modify σ by switching the jobs b_k corresponds to
   (d) $k \leftarrow k + 1$
5. in $[b_{|B^s|}, \infty[$ the tradeoff curve is given by linear function $B^w(\overline{\gamma}) = B^w(\overline{\gamma}, \sigma)$

Obviously, the algorithm finds all optimal sequences and, therefore, the whole tradeoff curve in $O(n^2 \log n)$. The complexity is not higher than the one for solving $1|B^w|\sum w_iC_i$ because in both cases sorting B^s takes the most effort.
In order to illustrate the connection between we give the following example. Let us suppose we have a set of jobs $I = \{1, 2, 3\}$. We have $p_i = i$ and $w_i = 1$ for each $1 \le i \le 3$, and

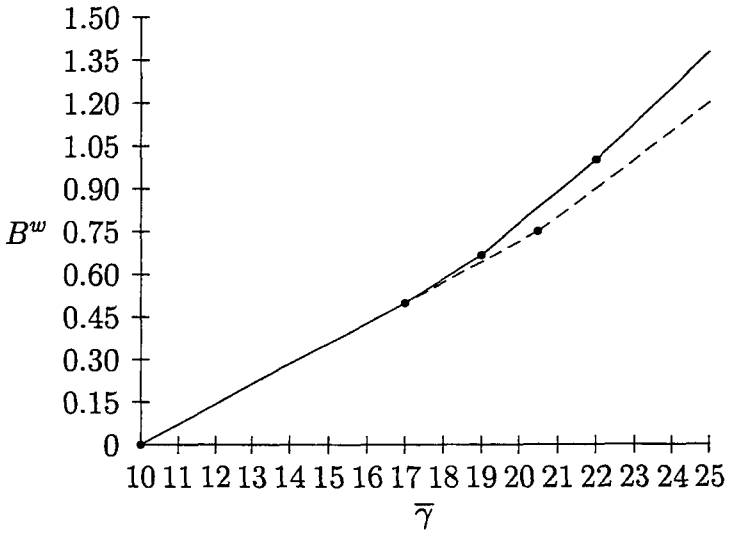

Figure 5: Trade Off Curve for $1||(\sum C_i, B^w)$

$w_1^b = 3.5$, $w_2^b = 1.5$ and $w_3^b = 0.5$. We then have $B^s = \{0.5, 2/3, 1\}$. Figure 5 illustrates the the graph corresponding to the trade off curve as the solid line. Break points are dotted and correspond to jobs 1 and 2, jobs 1 and 3, and jobs 2 and 3 switching in the optimal sequence. Note that the trade off curve is piecewise linear and convex which can also be observed from the formal expression of $B^w(\overline{\gamma})$. Due to the definition of B^s jobs i' and j', $p_{i'}/w_{i'} < p_{j'}/w_{j'}$, can switch positions in the optimal sequence only if $w_{i'}w_{j'}^b - w_{j'}w_{i'}^b < 0$. Let σ and σ' be the sequences before and after the switch. Then,

$$\sum_{i=1}^{n} w_{\sigma'(i)} \sum_{j=1}^{i-1} w_{\sigma'(j)}^b - \sum_{i=1}^{n} w_{\sigma(i)} \sum_{j=1}^{i-1} w_{\sigma(j)}^b = w_{i'}w_{j'}^b - w_{j'}w_{i'}^b < 0.$$

Note that for special case $1|B^p|\sum w_iC_i$ we have $B^s = \emptyset$ which means that each optimal sequence of jobs for $\underline{B^p} = 0$ is optimal for each $\underline{B^p} > 0$. Hence, in this case finding the tardeoff curve can be done in $O(n \log n)$.

4.1.2 $1|sp - graph, \sum w_iC_i|B^w$

First, we give a brief overview of the algorithm for $1|sp-\text{graph}|\sum w_iC_i$ by Lawler [18]. For a given problem the algorithm does a series of comparisons of values p_i/w_i. However, the exact values are of no importance as far as sequencing is concerned since it only decides which of two values is larger. In each iteration the algorithm may form a composite job (i, j) from two jobs i and j that is defined by $p_{(i,j)} = p_i + p_j$ and $w_{(i,j)} = w_i + w_j$. This is why, in contrast to Section 4.1.1, it is not sufficient to consider

$$B^s = \left\{ b \mid b = \frac{p_i w_j - p_j w_i}{w_i w_j^b - w_j w_i^b} > 0, i < j \right\}$$

to find all values of $\underline{B^w}$ leading to changing optimal sequences.

We pick up the example of Section 4.1.1 and add a single precedence constraint requiring that 2 cannot precede 1. Considering the decomposition tree we can see that 2 follows 1 immediately in each optimal sequence. Therefore, only optimal sequences are $\sigma_1 = (1,2,3)$ and $\sigma_2 = (3,1,2)$. It turns out that $\sigma_1 = (1,2,3)$ provides a better schedule for $\underline{B}^w < 0.75$ and that $\sigma_2 = (1,2,3)$ provides a better schedule for $\underline{B}^w > 0.75$. However, $0.75 \notin B^s$. In Figure 5 we represent the corresponding trade off curve as dashed line. Note that in the first section both trade off curves are identical due to identical optimal sequences. However, in contrast to Section 4.1.1 jobs 1 and 2 can not switch and from this point both trade off curves are different.

The reason for this effect is that composite job (i,j) replaces simple jobs i and j whenever it is decided that j follows i immediately. In our case, $(1,2)$ replaces 1 and 2. Then, comparing

$$\frac{p_1 + w_1^b \underline{B}^w + p_2 + w_2^b \underline{B}^w}{w_1 + w_2} \text{ to } \frac{p_3 + w_3^b \underline{B}^w}{w_3}$$

gives the result that 3 should precede $(1,2)$ for $\underline{B}^w > 0.5$. An extension of B^s covering all sequence changing robustnesses could be defined as

$$B'^s = \left\{ b \mid b = \frac{p_{I'} w_{I''} - p_{I''} w_{I'}}{w_{I'} w_{I''}^b - w_{I''} w_{I'}^b} > 0, I', I'' \subset I \right\}$$

where $p_{I'} = \sum_{i \in I'} p_i$, $w_{I'} = \sum_{i \in I'} w_i$, and $w_{I'}^b = \sum_{i \in I'} w_i^b$. However, since $|B'^s| \in O(2^{2n})$ we can not sort it in polynomial time.

Taking this into account, our algorithm works in a stepwise manner mimicking the algorithm by Lawler [18]. We determine $B^{s,1}$ which is identical to B^s in Section 4.1.1 and sort this set according to non-decreasing values. After having determined $b_{k^*}^1$ (analogue to b_{k^*}) we obtain a unique ordering of values

$$\frac{p_i + w_i^b b_{k^*-1}^1}{w_i}, i \in I^1 = I$$

which allows us to execute at least one step of the algorithm by Lawler [18]. As soon as a composite job (i,j) is created, we have to consider the modified set of jobs $I^2 = I^1 \setminus \{i,j\} \cup \{(i,j)\}$ starting the procedure all over again. More specifically, the algorithm can be given as follows.

Algorithm 2

1. $I^1 \leftarrow I$
2. for $k \leftarrow 1$
3. repeat
   (a) find and sort $B^{s,k}$ according to I^k
   (b) find $b_{k^*}^k$ using binary search on $B^{s,k}$ and the algorithm by Lawler [18]
   (c) having a unique ordering of jobs in I^k execute one step of the algorithm by Lawler [18]
   (d) if no composite job is created go to Step 3c
   (e) if a composite job is created obtain set of jobs I^{k+1} by adding the composite job and dropping all jobs being contained in the composite job from I^k

(f) if only one node is left in the decomposition tree go to Step 4

(g) $k \leftarrow k+1$

4. extract optimal sequence σ from the remaining job

5. obtain maximum robustness as $B^w(\overline{\gamma}, \sigma)$

Note that determining $B^{s,l}$, $l > 1$, takes less effort than determining $B^{s,1}$. Since we restrict ourselves to robustness values within $[b^{l-1}_{k^*-1}, b^{l-1}_{k^*}[$ we consider

$$B^{s,2} = \left\{ b \mid b^{l-1}_{k^*-1} \leq b = \frac{p_{(i,j)}w'_i - p'_i w_{(i,j)}}{w_{(i,j)}w'^b_i - w'_i w^b_{(i,j)}} < b^{l-1}_{k^*}, i' \in I^1 \setminus \{i,j\} \right\}$$

and obtain $|B^{s,l}| \in O(n)$.
Correctness of our algorithm follows from correctness of the algorithm for $1|sp-\text{graph}|\sum w_i C_i$ and the fact that sequencing is based only on elementary steps of deciding which of two values p_i/w_i and p_i/w_i is larger.
Computational complexity of our algorithm can be determined as follows. Finding and sorting $B^{s,1}$ takes $O(n^2 \log n)$. Employing binary search and the algorithm by Lawler [18] to find $b^1_{k^*}$ costs $O(n \log^2 n)$. Since $|B^{s,l}| \in O(n)$, $l > 1$, computational effort to determine and sort $B^{s,l}$ in following iterations is $O(n \log n)$. Furthermore, finding $b^l_{k^*}$ costs $O(n \log^2 n)$ again. Since we have no more than n iterations overall complexity is $O(n^2 \log^2 n)$.
Note, that analogue to Section 4.1.1 the optimal sequence of jobs does not change depending on $\underline{B^p}$. To see this, consider two disjoint subsets of jobs I^1 and I^2. Corresponding composite jobs have

$$\frac{\sum_{i \in I^1} p'_i}{\sum_{i \in I^1} w_i} \text{ and } \frac{\sum_{i \in I^2} p'_i}{\sum_{i \in I^2} w_i}$$

which translates to

$$\frac{\sum_{i \in I^1} p_i(1+\underline{B^p})}{\sum_{i \in I^1} w_i} \text{ and } \frac{\sum_{i \in I^2} p_i(1+\underline{B^p})}{\sum_{i \in I^2} w_i}$$

regarding the reduction of the robust problem to the underlying problem. Hence,

$$\frac{\sum_{i \in I^1} p_i(1+\underline{B^p})}{\sum_{i \in I^1} w_i} < \frac{\sum_{i \in I^2} p_i(1+\underline{B^p})}{\sum_{i \in I^2} w_i}$$

if and only if

$$\frac{\sum_{i \in I^1} p_i}{\sum_{i \in I^1} w_i} < \frac{\sum_{i \in I^2} p_i}{\sum_{i \in I^2} w_i}.$$

Therefore, we can determine the tradeoff curve trivially by finding the only optimal sequence in $O(n \log n)$.

4.2 $1|\sum U_i|B^w$

In this section we first develop properties of optimal solution. Then, algorithms for $1|\sum U_i|B^m$ and $1|\sum U_i|B^w$ are proposed.

Lemma 2. *In an optimal solution to* $1|\sum U_i|B^w$ *the number of late jobs is exactly* $\overline{\gamma}$.

Proof. First, we show that in an optimal schedule there is at least one tight job. Let a schedule be given such that all early jobs are scheduled before the first late job and let jobs be numbered according to this schedule. Suppose no job in the set of early jobs I^e is tight in schedule σ. Then, we can increase B^w_σ by

$$\min_{i \in I^e} \frac{d_i - C_i}{\sum_{j<i} w^b_j} > 0$$

which means that σ is not optimal.
Now, let assume we have a schedule σ with less than $\overline{\gamma}$ late jobs and i is the first tight job. Moving i to the end of schedule we can start each following job p_i time units earlier. Considering the above, we increase B^w_σ and, therefore, σ has not been optimal. □

4.2.1 $1|\sum U_i|B^m$

Note that the problem is not bounded for $\overline{\gamma} = n$ or $\overline{\gamma} = n-1$ and there is a job i having $p_i \leq d_i$. For $\overline{\gamma} \leq n-2$, Lemma 2 enables us to develop the following algorithm for $1|\sum U_i|B^m$. Since there is at least one tight job among exactly $n - \overline{\gamma}$ early jobs in an optimal schedule σ there are up to $n - \overline{\gamma} - 1$ buffers before the first tight job that determines B^m. Hence, the maximum robustness is

$$B^m \in \left\{ \frac{d_i}{k} \mid i \in I, k \in \{1, \ldots, n - \overline{\gamma} - 1\} \right\}.$$

Assuming that processing times as well as due dates are integer w. l. o. g., we obtain integer total buffer before the first tight job. Multiplying processing times and due dates by $\Pi_{k=2}^{n-\overline{\gamma}-1} k$ we obtain an integer maximum robustness as optimal solution to $1|\sum U_i|B^m$. Hence, we can apply binary search on $\left\{1, \ldots, \max_i d_i \Pi_{k=2}^{n-\overline{\gamma}-1} k\right\}$ to find maximum robustness. Note that

$$\log(\max_i d_i \Pi_{k=2}^{n-\overline{\gamma}-1} k) = O(n \log n).$$

Considering, that we have to solve the underlying problem $1||\sum U_i$ in each step which takes $O(n \log n)$ we obtain overall complexity of $O(n^2 \log^2 n)$.

4.2.2 $1|\sum U_i|B^w$

As for $1|\sum U_i|B^m$, the problem is not bounded for $\overline{\gamma} = n$ or $\overline{\gamma} = n-1$ and there is a job i having $p_i \leq d_i$. For all other cases we propose an algorithm using the known solution algorithm for the underlying problem $1||\sum U_i$ in an iterative procedure. The basic idea of the following algorithm is to focus on a set B^s such that we can find an interval $[b_{k^*-1}, b_{k^*}]$, $b_{k^*-1}, b_{k^*} \in B^s$, of robustness values containing the optimal solution value and providing a unique ordering of modified processing times and due dates for all $\underline{B}^w \in [b_{k^*-1}, b_{k^*}]$.
We consider the set

$$B^s = \left\{ b \mid b = \frac{p_i w_j - p_j w_i}{w_i w^b_j - w_j w^b_i} > 0, i, j \in I \right\} \cup \left\{ b \mid b = \frac{d_i w_j - d_j w_i}{w_i w^b_j - w_j w^b_i} > 0, i, j \in I \right\}.$$

We can find the smallest value b_{k^*} in B^s such that the optimal solution value to $1|B^w|\sum w_i C_i$ with $\underline{B}^w = b_{k^*}$ exceeds $\overline{\gamma}$ using binary search. This can be done in $O(n^2 \log n)$ time since $|B^s| \in O(n^2)$. The order of non-decreasing modified processing times $p'_i = p_i + b w^b_i$ and

non-decreasing modified due dates $d'_i = d_i + bw_i^b$ (breaking ties according to non-decreasing w_i^b) gives two total orders of the set of jobs for the optimal solution. Note that the algorithm by Moore [23] is based on steps dependent on the comparison of processing times or due dates only.

The basic idea of our algorithm is as follows. Based on the current solution (and, hence, given sequence of early jobs) for a given $\underline{B}^w$ we enlarge robustness as much as possible without one of the early jobs violating its due date. Let jobs be numbered according to non-decreasing due dates and let I^e be the set of early jobs. Then, the maximum amount b^+ by which robustness can be increased without the current sequence of early jobs getting infeasible is

$$b^+ = \min_{i \in I^e} \frac{d'_i - C_i}{\sum_{j \in I^e, j<i} w_j^b}.$$

If we set $\underline{B}^w = \underline{B}^w + b^+$ at least one job $i \in I^e$ will be tight. If we further increase robustness by ϵ, according to the procedure by Moore [23] a job $j = \arg\max_{j \in I^e, j \leq i}$ will be chosen to be late. Note that this does not mean that necessarily the number of late jobs goes up.

In the following we first specify the algorithm and afterwards provide a proof of correctness.

Algorithm 3

1. find b using binary search on B^s
2. solve problem $1|B^w|\sum U_i,\ \underline{B}^w = b$
3. repeat until the optimal solution has more than $\overline{\gamma}$ tardy jobs
   (a) find b^+
   (b) $b \leftarrow b + b^+ + \epsilon$
   (c) solve problem $1|B^w|\sum U_i,\ \underline{B}^w = b$

Let σ and σ' be the sequences of jobs before Step 3b and after Step 3c. Let j be the job having largest processing time p'_j among those jobs being on time and scheduled before tight job i. In the following we neglect the subschedule of late jobs and assume that late jobs are scheduled after the last early job. Jobs in I^e be numbered according to non-decreasing due dates.

Lemma 3. *If k is the position of j in σ, then $\sigma(k) < \sigma'(k)$ and $\sigma(k') = \sigma'(k')$ for each $k' < k$.*

Proof. Since the order of modified processing times and modified due dates is identical, the algorithm by Moore [23] processes identically for the first $\sigma(k) - 1$ jobs. To see that, note that if for a subset I' the largest due date (corresponding to job $i' \in I'$) was violated before Step 3b, then the same subset violates the largest due date again after Step 3c.

$$\begin{aligned} \sum_{i \in I'} (p'_i + w_i^b b^+) &> \sum_{i \in I'} p'_i + w_{i'}^b b^+ \\ &> d'_{i'} + w_{i'}^b b^+ \end{aligned}$$

Furthermore, let $I^{t,l} \subset \{1, \ldots, \sigma(k)\}$ be the subset of jobs chosen to be tardy in iteration l of the algorithm until $\sigma(k)$ is scheduled. Since the order of modified processing times does not change we obtain $I^{t,l} \subseteq I^{t,l+1}$. The Lemma follows. □

Theorem 6. *The algorithm terminates after no more than $O(n^3)$ iterations.*

Proof. Let jobs be numbered according to non-decreasing due dates. Consider a string of binary values indicating that job i is early in iteration l if and only if the corresponding bit equals 1. Note that the number of ones can never be increased during our algorithm. Regarding Lemma 3, the fact that in the solution given by the algorithm by Moore [23] early jobs are sorted according to non-decreasing due dates, and the unique order of due dates, we can sketch the behaviour of the bstring like this: For each number of tardy jobs the number of zeroes is fixed. The zeroes may go from left to right in the string which cannot take more than n^2 steps. So, the overall number of steps is n^3. □

Regarding Theorem 6 and the fact that we apply the algorithm of Moore [23] in each iteration we obtain a computational complexity of $O(n^4 \log n)$ to find the optimal solution to $1|\sum U_i|B^w$. To find the trade off curve we have to search B^s sequentially. This cummulates in run time complexity of

$$O(n^2 \cdot n^4 \log n) = O(n^6 \log n).$$

This curve is defined only for $\overline{\gamma} \in \{0, \ldots, n-2\}$ and gives the highest b that allowes for a certain number $n - \overline{\gamma}$ of early jobs.

4.3 $1|prec, f_{\max}|B^w$

In this section we consider robust counterparts of problem $1|prec|f_{\max}$ that is known to be solvable in $O(n^2)$, see Lawler [17]. First, we focus on the special cases $1|prec|C_{\max}$ and $1|prec|L_{\max}$. Afterwards, we consider a more general case where f_i is an arbitrary non-decreasing function in C_i. The basic idea is to employ algorithms known for the underlying problem in a procedure searching the domain of $\underline{B}$.

4.3.1 $1|prec, C_{\max}|B^w$ and $1|prec, L_{\max}|B^w$

Obviously, an arbitrary sequence of jobs (as long as precedence constraints are not violated) gives an optimal solution for $1|prec|C_{\max}$. The reduction of $1|prec, B^w|C_{\max}$ to $1|prec, B^w|f_{\max}$ as proposed in Section 3 leads to $p_i' = p_i + w_i^b \underline{B}^w$ and $f_i'(C_i') = f_i(C_i' - w_i^b \underline{B}^w) = C_i' - w_i^b \underline{B}^w$. The makespan (according to modified processing times) in the reduced problem is $\sum_i p_i'$ but due to the modification of f_i jobs may be differently suitable to be chosen as the last job. An intuitive explanation is that the buffer corresponding to the last job does not contribute to the makespan and should be, therefore, chosen as large as possible.
It is easy to see that chosing job

$$i^* = \arg\max_{i \in I} \left\{ w_i^b \mid (i,j) \notin E \, \forall j \neq i \right\}$$

as the last job provides an optimal solution for $1|prec, C_{\max}|B^w$. This implies that the choice is arbitrary for $1|prec, C_{\max}|B^m$. Note in both cases the optimal sequence of jobs does not depend on $\underline{B}^w$. Furthermore, we do not even need to find the optimal sequence of jobs to find the trade off curve. The curve is given as a function

$$B^w(\overline{\gamma}) = \frac{\overline{\gamma} - \sum_{i \in I} p_i}{\sum_{i \in I, i \neq i^*} w_i^b}$$

which means the trade off curve (that is linear in this case) can be found $O(n)$.

For $1|prec|L_{\max}$ we again refer to the reduction of $1|prec, B^w|f_{\max}$ to $1|prec|f_{\max}$ as proposed in Section 3 leading to a modification. We obtain $p'_i = p_i + w_i^b \underline{B}^w$ and $f'_i(C'_i, d'_i) = f_i(C'_i - w_i^b \underline{B}^w, d'_i - w_i^b \underline{B}^w) = C'_i - d'_i$. Therefore, we can apply a modification of the well known earliest due date rule: We choose the job having largest modified due date among those having no successor to be the last job. In order to solve $1|prec, L_{\max}|B^w$ we consider set of robustness values

$$B^s = \left\{ b \mid b = \frac{d_i - d_j}{w_j^b - w_i^b} > 0, i, j \in I \right\}.$$

Applying binary search to find the $b^* \in B^s$ that is the smallest value such that $1|prec, B^w|L_{\max}$ with $\underline{B}^w = b^*$ leads to an objective value exceeding $\overline{\gamma}$. This gives us the order of modified due dates for the optimal value of B^w and, hence, enables us to apply the modified due date rule. This takes takes $O(n^2 \log n)$ time since $|B^s| \in O(n^2)$. After finding the optimal sequence σ we can compute the maximum robustness as

$$\min_{1 \le i \le n} \left\{ \frac{\overline{\gamma} + d_{\sigma(i)} - \sum_{j \le i} p_{\sigma(j)}}{\sum_{j<i} w^b_{\sigma(j)}} \right\}$$

in linear time. Hence, overall computational complexity is $O(n^2 \log n)$.
Of course, by sequential search of B^s we can find all optimal sequences in $O(n^3 \log n)$. Note that there may be several break points of the trade off curve for a given sequence σ since the tight job

$$i^* = \arg\min_{1 \le i \le n} \left\{ \frac{\overline{\gamma} + d_{\sigma(i)} - \sum_{j \le i} p_{\sigma(j)}}{\sum_{j<i} w^b_{\sigma(j)}} \right\}$$

may change. It is easy to see that if i^* is the tight job for $\overline{\gamma}$ and j^* is the tight job for $\overline{\gamma}' > \overline{\gamma}$ and both optimal sequences are identical, then $j \ge i$. Hence, the tight job for a given sequence of jobs cannot change more than $n-1$ times. Since finding the next tight job is in $O(n)$ finding the whole trade off curve is in $O(n^5 \log n)$.
The trade off curve $B^w(\overline{\gamma}, \sigma, i^*)$ for given sequence of jobs σ and tight job i^* is linear and specified by

$$B^w(\overline{\gamma}, \sigma, i^*) = \frac{\overline{\gamma} + d_{\sigma(i^*)} - \sum_{j \le i^*} p_{\sigma(j)}}{\sum_{j<i^*} w^b_{\sigma(j)}}.$$

Since for given sequence σ each tight job for larger $\overline{\gamma}$ cannot be a predecessor of i^* in σ, we can see that the trade off curve for σ is concave.
However, as we illustrate with an example the trade off curve may not be concave in general. Consider 3 jobs specified by $p_1 = p_2 = p_3 = 1$, $w_1^b = w_3^b = 1$, $w_2^b = 2$, $d_1 = 1$, $d_2 = 2$, and $d_3 = 5$. We observe that job 1 can be scheduled first in each schedule since $d'_1 \le \min\{d'_2, d'_3\}$. Job 2 precedes and follows job 3 if $\underline{B} < 3$ and if $\underline{B} > 3$, respectively. Note that $\underline{B} = 3$ corresponds to $\overline{\gamma} = 7$. Since $L_{\max}$ cannot be negative (due to job 1) $B^w(\overline{\gamma})$ is defined for $\overline{\gamma} \ge 0$.
We observe that job 2 is tight for $\overline{\gamma} \in [0, 1]$ while job 3 is tight for $\overline{\gamma} \in [1, 7]$. Note that jobs 2 and 3 switch at $\underline{B} = 3$ and $\overline{\gamma} = 7$, respectively. For $\underline{B} \ge 3$ and $\overline{\gamma} \ge 7$ job 2 is tight resulting into a break point at the switch point, see Figure 6. Clearly, $B^w(\overline{\gamma})$ is neither concave nor convex. However, $B^w(\overline{\gamma})$ is concave for both optimal sequences corrseponding to intervals $[0, 7]$ and $[7, \infty[$.

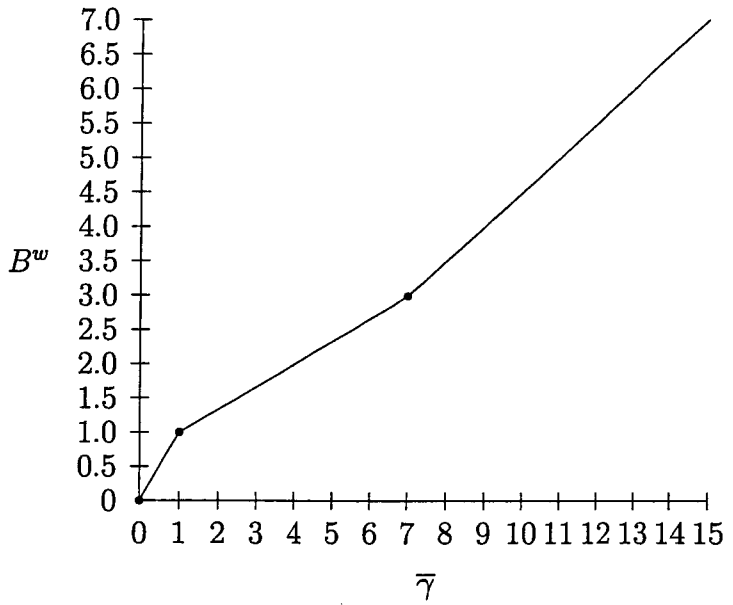

Figure 6: Trade Off Curve for $1||(L_{\max}, B^w)$

4.3.2 $1|prec, f_{\max}|B^w$

Here we consider the more general case where the only restriction on f_i is that it is non-decreasing function in completion time. Here, we distinguish two cases regarding f_i.

Theorem 7. *If finding* $\max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}$ *is NP-hard, then* $1|prec, f_{\max}|B^m$, $1|prec, f_{\max}|B^p$, *and* $1|prec, f_{\max}|B^w$ *are NP-hard also.*

Proof. We can reduce the problem to find $\max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}$ to $1|prec, f_{\max}|B^m$ as follows. Let the number of jobs be 2. Consider both jobs having unit processing time and functions $f_i(C) = f_j(C) = f(C-2)$. The maximum buffer size b between both jobs will obviously lead to the second job being finished exactly at $\max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}+2$. Subtracting overall processing time of 2, we obtain $B^m_\sigma = \max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}$ as value of optimal schedule σ. The proof can be done analogously for $1|prec, f_{\max}|B^p$ and $1|prec, f_{\max}|B^w$. □

Theorem 8. *If finding* $\max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}$ *can be done in polynomial time, then* $1|prec, f_{\max}|B^m$, $1|prec, f_{\max}|B^p$, *and* $1|prec, f_{\max}|B^w$ *can be solved in polynomial time.*

Proof. We reduce $1|prec, f_{\max}|B^m$, $1|prec, f_{\max}|B^p$, and $1|prec, f_{\max}|B^w$ to $1|prec, L_{\max}|B^m$, $1|prec, L_{\max}|B^p$, and $1|prec, L_{\max}|B^w$, respectively. Obviously $d_i' = \max\{C_i \mid f_i^{-1}(C_i) \leq \overline{\gamma}\}$ gives a due date for job i. Given an instance P of $1|prec, f_{\max}|B^w$ we create an instance P' of $1|prec, L_{\max}|B^w$ as follows:

- $n' = n$
- $p_i' = p_i$
- $d_i' = d_i'$
- $\overline{\gamma} = 0$

It is easy to see, that the optimal solution to P' provides an optimal solution to P. □

Consequently, the computational complexity of $1|prec, f_{\max}|B^m$ and $1|prec, f_{\max}|B^w$ is $O(C(d_i^\gamma) + n \log n)$ and $O(C(d_i^\gamma) + n^2 \log n)$, respectively, where $C(d_i^\gamma)$ gives the comutational complexity to find $\max\{C \mid f^{-1}(C) \leq \overline{\gamma}\}$. Corresponding trade off curves can be found in $O(C(d_i^\gamma) + n \log n)$ and $O(C(d_i^\gamma) + n^3 \log n)$.

5 Conclusions and Outlook

In this paper we propose three surrogate measures for robustness of a schedule on a single machine based on time buffers between jobs. We introduce a generic robustness counterpart for classic single machine scheduling problems and obtain a robust decision problem and three robust optimization problems corresponding to each classic single machine scheduling problem. While being reasonable our robustness concept has the advantage over other concepts in the literature that we do not loose polynomiality of classic problems when considering the robust counterpart in general. In particular, for the problem to minimize the objective function while providing a certain robustness level we show exactly in which cases we loose polynomialty. For the problem to maximize robustness while not exceeding a certain objective value we show for three problems exemplarily that we can obtain polynomial algorithms. Also trade off curves for minimizing the objective value and maximizing the robustness are found in polynomial time. For future research we can think of several variations and extensions of the concept provided in the paper at hand. First, incontrast to jobs being related with the following buffer each job may be related to the preceding buffer. This can be motivated by requiring a certain protection level for each job which is implemented by the preceding job. Furthermore, combinations of both concepts are possible. Second, the size or importance, respectively, of a buffer may depend on its position in the sequence. Of course the probability of interruptions before a specific position is higher for positions being located towards the end of the schedule and, hence, buffers at the end should be larger than at the beginning tendentially. Third, probabilistic analysis can be employed to derive the size of a specific buffer. Fourth, the concept proposed here can be extended to the case with more than one machine.

References

[1] M. Al-Fawzan and M. Haouari. Executing production schedules in the face of uncertainties: A review and some future directions. *International Journal of Production Economics*, 96:175–187, 2005.

[2] H. Aytug, M. Lawely, K. McKay, S. Mohan, and R. Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110, 2005.

[3] A. Ben-Tal and A. Nemirovski. Robust optimization methodology and applications. *Mathematical Programming, Series B*, 92:453–480, 2002.

[4] A. Ben-Tal and A. Nemirovski. Selected topics in robust convex optimization. *Mathematical Programming, Series B*, 112:125–158, 2008.

[5] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):3553, 2004.

[6] C. Briand, H. T. La, and J. Erschler. A Robust Approach for the Single Machine Problem. *Journal of Scheduling*, 10:209–221, 2007.

[7] P. Brucker. *Scheduling Algorithms*. Springer, 2004.

[8] R. L. Daniels and J. E. Carrillo. β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29:977–985, 1997.

[9] R. L. Daniels and P. Kouvelis. Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-stage Production. *Management Science*, 41(2), 1995.

[10] A. J. Davenport and J. C. Beck. A Survey of Techniques for Scheduling with Uncertainty. *Working Paper*, 2000.

[11] S. Goren and I. Sabuncuoglu. Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions*, 40:66 83, 2008.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:236–287, 1979.

[13] W. Herroelen and R. Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004.

[14] P. Kobylanski and D. Kuchta. A note on the paper by M. A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, 107:496–501, 2007.

[15] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, 1997.

[16] P. Kouvelis, R. L. Daniels, and G. Vairaktarakis. Robust Scheduling of a Two-Machine Flow Shop with Uncertain Processing Times. *IIE Transactions*, 2000.

[17] E. Lawler. Optimal Sequencing of a Single Machine subject to Precedence Constraints. *Management Science*, 19:544–546, 1973.

[18] E. Lawler. Sequencing Jobs to Minimize Total Weighted Completion Time Subject To Precedence Constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.

[19] V. J. Leon, S. D. Wu, and R. H. Storer. Robustness Measures and Robust Scheduling for Job Jobs. *IIE Transactions*, 26(5):32–43, 1994.

[20] R. Leus and W. Herroelen. The Complexity of Machine Scheduling for Stability with a Single Disrupted Job. *OR Letters*, 33:151–156, 2005.

[21] R. Leus and W. Herroelen. Scheduling for stability in single-machine production systems. *Journal of Scheduling*, 10(3):223–235, 2007.

[22] S. V. Mehta and R. Uzsoy. Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer-Integrated Manufacturing*, 12:15–38, 1999.

[23] J. M. Moore. An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. *Management Science*, 15(1):102–109, 1968.

[24] Z. Shi, E. Jeannot, and J. J. Dongarra. Robust task scheduling in non-deterministic heterogeneous computing systems. In *IEEE International Conference on Cluster Computing*, 2006.

[25] S. Van de Vonder, E. Demeulemeester, W. Herroelen, and R. Leus. The Use of Buffers in Project Management: The Trade-Off between Stability and Makespan. *International Journal of Production Economics*, 2005.

[26] S. D. Wu, E. Byeon, and R. H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47:113–124, 1999.