Technische Universität Berlin
Fachbereich Mathematik

Masterarbeit

Algorithmic Study of Bilevel Machine Scheduling Problems

Angefertigt von:	Felix Paul Simon
Matrikel-Nr.:	320383
E-Mail:	felix@simonberlin.de
Betreuerin &	
Erstgutachtering:	Dr. Nicole Megow
Zweitgutachter:	Prof. Dr. Rolf H. Möhring
Datum:	13. Juli 2014

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den ....

....
Unterschrift

Zusammenfassung

Die vorliegenden Arbeit beschäftigt sich mit dem *Bilevel Weighted Completion Time Problem*, einem Bilevel Machine Scheduling Problem auf parallelen identischen Maschinen. In der Bilevel-Optimierung gibt es zwei Spieler auf verschiedenen Entscheidungsebenen. Es besteht die Aufgabe, das Problem des höhergestellten Spielers zu lösen, welcher die auf seine Entscheidung folgende Reaktion des anderen Spielers mit berücksichtigen muss. Daher kann die Bilevel-Optimierung auf eine Vielzahl von realen Planungsproblemen mit hierarchischen Entscheidungsstrukturen angewendet werden.

In dieser Arbeit werden einerseits strukturelle Ergebnisse für das Problem und andererseits verschiedene Ansätze zur Approximierung des Problems vorgestellt. Diese Ansätze beinhalten an erster Stelle ein voll-polynomielles Approximationsschema (FPTAS) für den Spezialfall einer fixen Anzahl an gegebenen Maschinen. Weiterhin werden verschiedene Algorithmen vorgestellt, die die Jobs anhand einer gegebenen Liste den Maschinen zuweisen oder gültige Lösungen aus der Optimallösung des zugrundeliegenden Problems $P||\sum_j w_j C_j$ mit einer Entscheidungsebene generieren. Außerdem wird sowohl ein ganzzahliges Modell (IP) als auch ein ganzzahliges quadratisches Modell (IQP) des Problems präsentiert. Den Abschluss der Arbeit bildet die Auswertung von einigen behandelten Algorithmen anhand einer detailierten experimentellen Analyse auf einer Menge von zufällig erzeugten Instanzen.

Abstract

In this thesis we consider the *Bilevel Weighted Completion Time Problem*, a bilevel machine scheduling problem on parallel identical machines. The task in bilevel optimization is to optimize the problem of a top level decision maker, while taking the subsequent reaction of a bottom level decision maker into account. For this reason it can be applied to many real world planning problems with hierarchical structures.

We will give some structural results for the problem and present several approaches for achieving approximation algorithms. These include a fully polynomial-time approximation scheme (FPTAS) for the case of a fixed number of machines and different algorithms processing the job in the order of a given list. Furthermore we generate solutions based on solving the one-level problem $P||\sum_j w_j C_j$. Additionally an integer programming (IP) and an integer quadratic programming (IQP) formulation for the problem are given. The thesis is concluded by a detailed experimental evaluation of some of the presented algorithms on a set of randomly generated instances.

Contents

Introduction

In mathematical optimization one faces the problem of finding a solution, that optimizes some objective function under a given set of constraints. Sometimes these constraints even contain a second mathematical optimization problem. This is the case for *bilevel optimization problems*. Here the decisions, that have to be made, are partitioned into two subsets. The first set of decisions is made by the so-called "*leader*", which determines with his choice the conditions for the second decision maker named the "*follower*". Each decision maker solves his own optimization problem, that could depend in terms of the objective function or the given constraints on both decision sets. The goal of a bilevel optimization problem is to optimize the problem of the leader, taking into account the subsequent decision of the follower. Therefore it is highly connected to game theory, in which, however, one mainly searches for equilibria. The reason for this is, that the players have the possibility to correct their decision based on the made decisions of the other player or both players have to decide simultaneously.

One often distinguishes in bilevel optimization between the *optimistic* and the *pessimistic* case based on the behavior of the follower facing multiple optimal solutions. While in the optimistic case the follower chooses the optimal solution optimizing the leader's objective, he chooses the worst optimal solution from the leader's point of view in the pessimistic case.

Bilevel optimization can be applied to many real world problems with hierarchical decision structures. The most natural case is the one of a manager or project planner delegating tasks to his employees or subcontractors, which have their own objectives and definitions of a feasible solution. But also the planning of public infrastructure is a bilevel optimization problem, since the planer have to take the selfish decisions of the individual users into account.

There are several approaches for bilevel optimization problems in the literature, like studies on bilevel assignment problems in [GasKli2009] or bilevel knapsack problems in [CapCarLodWoe2014]. While the problems presented in the first paper are proved to be $\mathcal{NP}$-hard, the bilevel knapsack problems of [CapCarLodWoe2014] are Σ_2^p-complete. In the polynomial hierarchy this complexity class is located one level above the class $\mathcal{NP}$. Therefore are problems that are Σ_2^p-complete at least as hard as those being $\mathcal{NP}$-complete.

For an overview of the polynomial hierarchy and a broad collection of Σ_2^p-complete problems we refer to [Joh2011]. Here it was also shown that bilevel 0-1 integer programming is Σ_2^p-complete as well.

It this thesis we will study a bilevel machine scheduling problem, the Bilevel Weighted Completion Time Problem (BWCTP), which is based on the well studied one-level problem $P||\sum_j w_jC_j$. The problem was initially presented and studied in [KisKov2012], where it was also proved to be $\mathcal{NP}$-hard. Apart from this paper, there are only few results on bilevel machine scheduling in the literature, like for example [KarWan1996], [KisKov2011] and [LukRosSo2008], who present heuristics to solve bilevel programming approaches for scheduling problems.

In contrast to this, the following thesis will mainly deal with approximation approaches and is organized as follows. In Chapter 1 we will given some basic definitions of the used terms and concepts. Afterwards we will precisely define the studied BWCTP and summarize known approximation and complexity results for it as well as for the one-level version $P||\sum_j w_jC_j$ of the problem. The next Chapter 2 will firstly give some new general structural results for the BWCTP. Since the problem is known to be $\mathcal{NP}$-hard in general, we will present various natural approximation approaches in the following sections. In addition we will provide an integer programming formulation and an integer quadratic programming formulation for the BWCTP in Chapter 3. Furthermore some approximation approaches based on the non-integer relaxations of the models will be presented here. The thesis is concluded in Chapter 4 by a detailed empirical evaluation of some of the algorithmic approaches, presented in the previous chapters.

1. Definitions and known results

In this initial chapter we will prepare the study following in the subsequent chapters, by giving precise definitions of the used terms and concepts. Furthermore we will define the analyzed problems and present some known structural and algorithmic results from the literature.

1.1. Basic concepts

The following section contains basic definitions from the field of mathematical optimization, especially discrete mathematical optimization, which will be used during the thesis. It is based on the notes of the lectures given by Möhring [Moe2010] and Grötschel [Gro2010] as well as the book chapter of Schuurman and Woeginger [SchWoe2001].

First the general definition of a mathematical optimization problem is given following the definition in [Gro2010].

Definition 1.1 (Optimization problem)**.** *An* optimization problem P *consists of a set* $\mathcal{S}$ *of solutions, an objective function* $c : \mathcal{S} \to \mathbb{R}$ *and a set* $\mathcal{C}$ *of constraints. The set* $\mathcal{S}' \subseteq \mathcal{S}$ *of all solutions fulfilling all constraints in* $\mathcal{C}$ *is called the set of* feasible *solutions. The aim of an optimization problem is now to find some feasible* $S^* \in \mathcal{S}'$ *such that either* $c(S^*) \leq c(S) \quad \forall S \in \mathcal{S}'$ *(minimization problem) or* $c(S^*) \geq c(S) \quad \forall S \in \mathcal{S}'$ *(maximization problem) holds. The solution* S^* *is called* optimal *solution.*

Since most of the problems discussed in this thesis are minimization problems, the objective function c will sometimes also be referred to as *cost function*.

Given an optimization problem, a procedure which finds preferably good and feasible solutions is called an *algorithm*. To classify algorithms which solve optimization problems according to their efficiency, a measure for the size of an optimization problem is needed. The definition of the input size of an optimization problem is given based on [Moe2010].

Definition 1.2 (Input size)**.** *Given an optimization problem* P *and some encoding* E*. The* input size *of the optimization problem is the number of characters needed to encode* P *using* E*.*

The relevant encoding types for this thesis are the *binary* and the *unary* encoding, using two respectively only one character.

Given these definitions and the concept of the *runtime* or *execution time* of an algorithm stated in [Moe2010], it is now possible to define an algorithm with polynomial execution time.

Definition 1.3 (Polynomial time algorithm). *Given an optimization problem P, whose input is binary encoded. Then, an algorithm with polynomial execution time or shortly a* polynomial time algorithm *is an algorithm, which finds an optimal solution of P and has an execution time, that is bounded from above by a polynomial in the size of the input.*

These kind of algorithms are in general considered to solve the given optimization problem efficiently. Often also the term "*solvable in polynomial time*" is used to declare the existence of a polynomial algorithm for the optimization problem. By using another input encoding, a second efficiency class of algorithms can be defined, the pseudo-polynomial algorithms.

Definition 1.4 (Pseudo-polynomial time algorithm). *Given an optimization problem P, whose input is unary encoded. An algorithm with pseudo-polynomial execution time or shortly a* pseudo-polynomial time algorithm *is an algorithm, which finds an optimal solution of P and has an execution time, that is bounded from above by a polynomial in the size of the unary encoded input.*

Therefore a pseudo-polynomial algorithm's execution time can depend on the number of input parameters as well as on the size of their absolute values.

There are many optimization problems for which no polynomial time algorithm is known, for example the problems lying in $\mathcal{NP}$. These problems presently cannot be solved efficiently to optimality. One possibility to deal with this situation is to look for algorithms, which find solutions in reasonable time, that can be proved to be near to the optimal value. These algorithms are called approximation algorithms. For a general survey of approximation algorithms see for example Shmoys and Williamson [ShmWil2010] or Schuurman and Woeginger [SchWoe2001]. Based on the second one, the following definition is given.

Definition 1.5 (α-approximation algorithm). *Given an algorithm $\mathcal{A}$ for a minimization respectively maximization problem P and let α be defined as $(1+\epsilon)$ respectively $(1-\epsilon)$ for $\epsilon > 0$. Then $\mathcal{A}$ is called a α-*approximation algorithm*, if for the optimal solution S^* of P and the solution A generated by the algorithm it holds:*

$$|c(A) - c(S^*)| \leq \epsilon \cdot c(S^*)$$

In this case, α is called a performance ratio, approximation ratio *or* worst case ratio *of the approximation algorithm $\mathcal{A}$.*

For some problems, a complete family of approximation algorithms is known and can be used to generate a solution of the desired approximation ratio. The definition is again adopted from the one of Schuurman and Woeginger [SchWoe2001].

Definition 1.6 (Approximation schemes)**.** *An* approximation scheme *for an optimization problem P is a family of algorithms, such that each algorithm $\mathcal{A}_\epsilon$ is a $(1+\epsilon)$- respectively $(1-\epsilon)$-approximation algorithm for the given problem.*

A polynomial time approximation scheme *(PTAS) is an approximation scheme which has an execution time, that is bounded from above by a polynomial in the size of the input.*

A fully polynomial time approximation scheme *(FPTAS) is a PTAS, whose execution time is also bounded from above by a polynomial in $\frac{1}{\epsilon}$.*

Although an approximation scheme provides an approximation algorithm for each arbitrarily small ϵ, it cannot be used to solve problems with arbitrary precision in reasonable time since in general the execution time depends on $\frac{1}{\epsilon}$. This motivates the definition of an FPTAS, whose runtime dependency on $\frac{1}{\epsilon}$ can be described and bounded by a polynomial. Nevertheless, approximation schemes are a very powerful tool to solve problems, for which no polynomial time algorithm is known.

Since we will frequently deal with different job orders, we will need the following last definitions.

Definition 1.7. *Let $>_A$ and $>_B$ be two order on the jobset J. Then the job order $>_B$* respects *the job order $>_A$ if and only if it holds:*

$$j <_A k \Rightarrow j <_B k$$

Furthermore we say that the jobset J is indexed according to *a job order $>_A$ if it holds:*

$$j <_A k \Rightarrow j < k$$

1.2. Problem definition and known results

The problem which will be studied in this thesis is the Bilevel Weighted Completion Time Problem or BWCTP in short, which was first defined and investigated by Kis and Kovács in [KisKov2012]. It is a parallel machine scheduling problem based on the Weighted Completion Time Problem, which will be defined below. For a general overview of scheduling problems and machine scheduling problems see for example the book of Brucker [Bru2007].

The definition below follows the description in [KisKov2012].

Problem 1.8 (Bilevel Weighted Completion Time Problem). *Given two decision makers named "leader" and "follower" and a set of jobs J as well as a set $\mathcal{M}$ of parallel identical machines with $n = |J|$ and $m = |\mathcal{M}|$. For each job $j \in J$ two positive integer weights are given by w_j^1 and w_j^2 as well as the processing time p_j, which is also positive and integer.*

The goal is to find a schedule by assigning the jobs to machines and choose a job order on each machine. Thus, the jobs have to be scheduled non-preemptively on the machines, which means that each job has to run on exactly one machine without interruption.

The decision on how to schedule the jobs is partitioned into the decision of the leader decision, who assigns the jobs to the machines, and the decision of the follower, who determines the order the jobs on the machines. Given the decision of the leader, the follower schedules the jobs on each machine such that his objective value $c_F(S) = \sum_{j \in J} w_j^2 C_j$ for a given schedule S is minimized, with C_j denoting the completion time of job j in S.

The problems objective is to minimize the leader's objective value $c(S) = \sum_{j \in J} w_j^1 C_j$ by deciding which job has to be scheduled on which machine and taking the subsequent decision of the follower into account.

Note that we generally assume that the number n of jobs is greater or equal to the number m of machines, since otherwise the optimal solution could trivially be achieved by scheduling each job on a single machine.

As already mentioned above, the BWCTP is based on is the Weighted Completion Time Problem, which will be referred to as WCTP during the thesis. It is well studied in the literature and can be defined in the following way:

Problem 1.9 (Weighted Completion Time Problem). *Given a set of jobs J and a set $\mathcal{M}$ of identical machines with $n = |J|$ and $m = |\mathcal{M}|$. For each job $j \in J$ a positive integer weight w_j and a positive integer processing time p_j is given.*

The problem's objective is to schedule the jobs non-preemptively on the machines, such that the weighted completion time $c(S) = \sum_{j \in J} w_j C_j$ for a given schedule S is minimized.

Thus the WCTP can be obtained from the BWCTP by combining the two decision makers to a single decision maker or decision level. This problem will therefore also be referred to as the *one-level problem.* As in the bilevel case, we assume the number n of jobs to be at least as big as the number m of machines. In the standard three-field notation for machine scheduling problems by Graham et al.[GraLawLenK1979] the WCTP can be written as $P||\sum_j w_j C_j$.

As a further description of the defined problems the following assumption is made:

Assumption 1.10. *Each feasible schedule for the BWCTP or the WCTP is assumed to use all available machines, so no machine remains empty. Furthermore, the jobs are scheduled on the machines without any idle times.*

Note that in an optimal solution for one of the problems both assumption are fulfilled.

The remaining part of the chapter summarizes known results for both, the WCTP and the BWCTP. We will first repeat some known results for the WCTP.

1.2.1. One-level problem results

It is well known that the problem WCTP is $\mathcal{NP}$-hard in the strong sense (cf. [SkuWoe2000] referring to [GarJoh1979]) and $\mathcal{NP}$-hard in the ordinary sense for a constant number of machines $m \geq 2$ [BruLenK1977]. Note that in the proof for constant machine numbers $w \equiv p$ holds. Therefore the WCTP is $\mathcal{NP}$-hard in this special case, too. Hence, there is in general no polynomial time algorithm to solve the problem under the assumption of $\mathcal{P} \neq \mathcal{NP}$. However, in the case of $w \equiv 1$ WCTP can be solved in polynomial time [BruLenK1977] by using for example a minimal cost flow approach.

If there is only one machine to schedule the jobs on, an optimal solution can be found by sorting the jobs on this machine according to "Smith's Rule". It sorts the jobs descending to the ratio $\frac{w}{p}$, so the weight per processing time of a job [Smi1956]. The optimality of this solution can be proved by a job exchange argument.

By the same argument it follows that there is an optimal solution for the general WCTP, scheduling the jobs on each machine in "Smith's Rule" order without idle times, see for example [LawLenKanShm1993]. This result justifies the made Assumption 1.10.

Although there cannot be a polynomial time algorithm solving the general version of the WCTP exactly, there are several approximation approaches to generate good solutions in reasonable time. One of them is the PTAS provided by Skutella and Woeginger [SkuWoe2000]. In the approach they first present a PTAS for the special case of the WCTP with a fixed range for the job ratios $\frac{w}{p}$. This result is afterwards applied to a partitioning of the jobs into subsets with similar weight to processing time ratios to conclude a PTAS for the general case.

If the number of machines m is not considered to be part of the input but being fixed, Schuurman and Woeginger showed even the existence of an FPTAS for WCTP in the case of $m = 2$, which can be extended to an FPTAS for general fixed m [SchWoe2001]. It is based on a pseudo-polynomial time algorithm combined with clever trimming of the solution space and will be reviewed more detailed in Section 2.2.

There are also some other approximation results apart from the approximation schemes stated above. One of them is based on a list scheduling algorithm, which uses the “Smith's Rule” order, that was already mentioned earlier in this section. Here, the jobs are scheduled in the given order to that machine with the current minimal load, that is, the minimal sum of the processing times of all the jobs scheduled so far to that machine. It is easy to prove that this gives a 2-approximation by using an upper bound on C_j, which is based on the structure of the resulting solution, and the following general lower bounds to an optimal schedule S^* of the WCTP. For this lower bounds we assume the jobs to be indexed according to the “Smith's Rule” order.

$$\begin{aligned} c(S^*) &\geq \sum_{j \in J} w_j p_j \\ c(S^*) &\overset{(*)}{\geq} \sum_{j \in J} w_j \sum_{k \leq j} \frac{p_k}{m} + \frac{m-1}{2m} \sum_{j \in J} w_j p_j \\ &\overset{(**)}{\geq} \frac{1}{m} \sum_{j \in J} \sum_{k < j} w_j p_k \end{aligned}$$

The first lower bound is obtained by simply adding the caused objective contribution of each job j when running non-preemptively for a time of p_j on an arbitrary machine. For the second bound, inequality $(*)$ goes back to Eastman et al. [EasEveIsa1964] and inequality $(**)$ just removes a non-negative term. For the same algorithm Kawaguchi and Kyan proved an approximation ratio of $\frac{1}{2}(1+\sqrt{2})$ and gave also an instance of the problem, for which the bound is attained [KawKya1986].

Another approximation approach is the randomized machine choosing, in which the jobs are scheduled on each machine with probability $\frac{1}{m}$. On each single machine “Smith's Rule” is used to sort the jobs afterwards. The expected objective value can be proved to be at most 2 times as high as the optimal objective value by using the same bounds as for the list scheduling approximation ratio of 2 before. From this a 2-approximation algorithm can be build via derandomization techniques.

This bound has been improved by Skutella using randomized rounding based on the solution of a convex quadratic program [Sku2001]. His approach, which proves an approximation ratio of 2, will be further described in Section 3.2.

1.2.2. Bilevel problem results

The BWCTP was formulated for the first time in the paper of Kis and Kovács [KisKov2012]. Here also some first findings were given, which we will introduce in the following.

The result with the biggest impact regards the subproblem the follower has to solve. Given the leader's decision about the distribution of the jobs to the machines, he has to order the jobs on the machines with the objective of minimizing $\sum_{j\in J} w_j^2 C_j$. Since each machine can be considered independently of the others, he is facing m single machine problems, which can be solved by ordering the jobs on the machine according to "Smith's Rule" using the weights w^2, see the previous subsection. Hence the follower's optimality constraint can be translated into a given order of the jobs on the machines.

In bilevel optimization one often distinguishes the optimistic and pessimistic case, in which the follower chooses that optimal solution to his subproblem, which is best respectively worst for the leader. Kis and Kovács [KisKov2012] prove that for each instance I of the problem there is an instance $\bar{I}$ with unique "Smith's Rule" order for the follower, such that all optimal solutions of $\bar{I}$ are optimal for I, too. This result holds for the optimistic case as well as for the pessimistic case. Therefore the BWCTP can be analyzed under the general assumption, that the follower's order is a total order. This total order of the follower will from now on be referred to as $>_F$ or as the *follower's order*. Since it will also be used frequently during the following explanations and in the rest of this thesis, the partial order generated by "Smith's Rule" using leader weights w^1 will be referred to as $>_L$ or as the *leader's order*. Formally these two job order can be defined for two jobs j and k in J as:

$$j <_F k :\Leftrightarrow \frac{w_j^2}{p_j} > \frac{w_k^2}{p_k}$$
$$j <_L k :\Leftrightarrow \frac{w_j^1}{p_j} > \frac{w_k^1}{p_k}$$

As another result, the complexity of the BWCTP is determined in the given paper. Since it holds for $w^1 \equiv w^2$ that on the one hand the follower's order $>_F$ is equal to the order $>_L$ using the leader's weights and on the other hand there is an optimal solution of the WCTP, in which the jobs on the machine are ordered by "Smith's Rule", the BWCTP is equivalent to the WCTP with weights w^1 in this case. Therefore the WCTP is a special case of the BWCTP and inherits the membership to the class of strongly $\mathcal{NP}$-hard problems in the general case and the one to the class of $\mathcal{NP}$-hard problems for m being constant to the BWCTP. In contrast, the complexity of the $w^1 \equiv 1$ and $p_j \equiv 1$ special cases of the BWCTP are yet unknown.

As a consequence of the argument above, Kis and Kovács [KisKov2012] conclude that the BWCTP is equivalent to the one-level problem with weights $w := w^1$ if the two "Smith's Rule" orders using w^1 and w^2 are identical.

It is also shown that the BWCTP can be solved using a special case of the MAX k-CUT problem. In this problem, given a complete graph with non-negative edge weights, a set of edges with maximal sum of weights is searched, such that removing these edges results in a graph with exactly k connected components. To solve the BWCTP, the nodes of the graph are identified with the set of jobs in the BWCTP and the weight of an edge (j, k) is chosen to be $w(j, k) := p_j w^1_k$ if and only if $j <_F k$. An optimal solution of the BWCTP is then obtained by solving the MAX k-Cut problem for $k = m$ and scheduling all jobs on the same machine, whose corresponding nodes lie in one connected component.

In the following part of the paper, the special case of $w^1 \equiv 1$ is analyzed. First Kis and Kovács [KisKov2012] show that the BWCTP is solvable in polynomial time if in addition the order of the follower $>_F$ respects the leader's "Smith's Rule" order $>_L$. The proof is based on the above argument, that the BWCTP and the WCTP are equivalent in this case. There is also a polynomial time algorithm for the unit weight special case of the BWCTP, if the *reversed* follower's order $\bar{>}_F$ respects the partial order of the leader. Here an optimal solution S^* of the following structure exists, supposing the jobs being indexed according to $>_F$:

> For the jobset J_i of jobs being scheduled on machine i, it holds
> $J_i = \{\pi_{i-1} + 1, \pi_{i-1} + 2, \dots, \pi_i\}$ with $\pi_0 := 0$ and $1 \leq \pi_1 \leq \pi_2 \leq \dots \leq \pi_m = n$.

In other words the solution can be found by building first a list of jobs sorted by index and then choosing $m - 1$ points to cut the list into m pieces. The machines then correspond to the m pieces, preserving the order of the list also on the machines, since the list was sorted according to $>_F$.

To find the best solution of that kind Kis and Kovács [KisKov2012] suggest a shortest path approach but there is also a simple dynamic program to solve the problem. The proof for the existence of such an optimal solution is made using the connection to the MAX k-CUT problem. Note that this special structure of an optimal solution only holds for the special case of $w^1 \equiv 1$ and not for the general case of the BWCTP.

For the general case with weights $w^1 \equiv 1$, two integer programming (IP) formulations are provided as well as a heuristic using an optimal solution of the corresponding linear programming (LP) relaxations to generate a feasible integer solution. Both IP models describe the BWCTP exactly. Since a one-to-one correspondence between the optimal solutions of the two LP relaxations is shown, only the second IP formulation using less variables will be presented and used in this thesis. This IP model will from now on be called $(\text{IP})_{\text{KK}}$.

Preliminary the job indexing is assumed to be *reversed* to the $>_F$ order, hence the job with the smallest index on a machine has to be scheduled behind all other jobs. Additionally the position indices are also *reversed*. Thus the last job on a machine has position 1, the penultimate job has position 2 and so on. Therefore for two jobs j and k in J with $j < k$ holds, that if j and k are scheduled on the same machine, job j has to have a smaller position than k.

Now $(\mathrm{IP})_{\mathrm{KK}}$ can be formulated in the following way using variables $x_{\rho,j}$ to indicate that job j is scheduled on position ρ:

(IP)$_{\mathbf{KK}}$

$$\text{Objective:} \qquad \min_x c(x) = \sum_{j=1}^{n} \sum_{\rho=1}^{m_j} \rho x_{j,\rho} p_j$$

$$\sum_{\rho=1}^{m_j} x_{j,\rho} = 1 \quad \forall\, j = 1, \dots, n \tag{1.1}$$

$$\sum_{j=\rho}^{n} x_{j,\rho} \le m \quad \forall\, \rho = 1, \dots, n-m+1 \tag{1.2}$$

$$\sum_{j=\rho}^{l} x_{j,\rho} \ge \sum_{j=\rho+1}^{l+1} x_{j,\rho+1} \quad \forall\, l = 1, \dots, n-m \text{ and } \rho \le l \tag{1.3}$$

$$\text{position variables:} \qquad x_{j,\rho} \in \{0,1\} \quad \forall\, j = 1, \dots, n, \rho = 1, \dots, m_j \tag{1.4}$$

In the model, m_j denotes the largest possible position of job j in an solution of the BWCTP without idle machines, which coincides with Assumption 1.10. It is shown that m_j is equal to $\min\{j, n-m+1\}$. Furthermore note, that for the position ρ_j of job j in a feasible schedule $\rho_j \le j$ holds. Otherwise there would be a job $k > j$ on a smaller position of the same machine as j. Hence, only variables satisfying this condition are considered.

The constraints (1.1) and (1.2) ensure that each job is scheduled on exactly one machine respectively that there are at most m jobs on one position. The third constraint (1.3) is stated to guarantee for each feasible solution of $(\mathrm{IP})_{\mathrm{KK}}$ the existence of a feasible solution for the BWCTP with corresponding positions. The corresponding proof will be given in Section 3.1 of this thesis by showing, how such a solution can be constructed. On the other hand it is easy to see, that the variable assignment resulting from a feasible machine solution satisfies (1.3). Let J_l be the set of the l jobs with smallest index. The constraint requires the sum of jobs from J_l on position ρ to be at least as big as the sum of jobs from J_{l+1} on position $\rho + 1$. In a feasible machine solution, each job j on position $\rho + 1$ is followed by some job $k < j$ because there are no idle times due to Assumption 1.10. Hence for each contribution of some job in J_{l+1} to the sum on position $\rho + 1$ there is a unique contribution of the same size by some job in J_l to the sum on position ρ, since the positions and indices are counted backwards. That is why each variable assignment from a feasible machine solution satisfies (1.3). Since also the constraints (1.1) and (1.2) are fulfilled by such a variable assignment, each feasible machine solution S admits a feasible solution x_S for $(\text{IP})_{\text{KK}}$. Furthermore the objective values coincide, since for the special case of $w^1 \equiv 1$, the contribution of a job j in S to the objective function value is equal to the processing time p_j of the job multiplied with the number of jobs, whose completion time includes p_j. This number is equal to the number of jobs following j on its machine plus the job j itself. Thus it is equal to the position ρ of the job as defined for $(\text{IP})_{\text{KK}}$. Therefore the objective values of S and x_S coincide.

For the LP relaxation of the model named $(\text{LP})_{\text{KK}}$, the constraint (1.4) is exchanged with the term $0 \leq x_{j,\rho} \quad \forall j = 1, \dots, n, \rho = 1, \dots, m_j$. The missing upper bound of 1 is already covered by constraint (1.1).

After showing the equivalence of the LP relaxations of the two IP formulations, Kis and Kovács [KisKov2012] present a theorem, which analyzes $(\text{LP})_{\text{KK}}$ in a case proved to be polynomial solvable in the beginning of the paper.

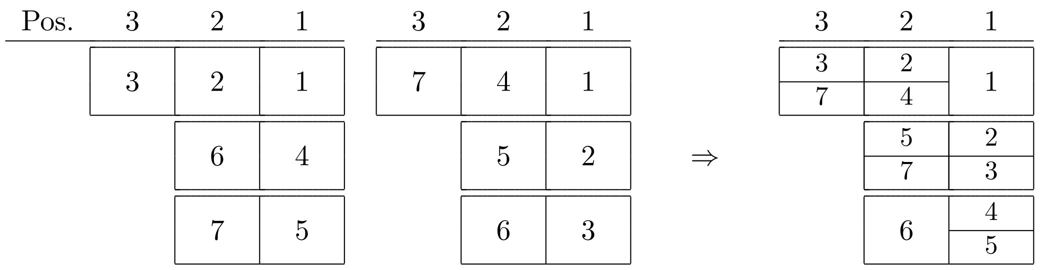
Theorem 1.11. *If $>_F$ is respecting $>_L$, then $(LP)_{KK}$ admits an optimal integer solution. If additionally $>_L$ is a total order then all optimal solutions of $(LP)_{KK}$ are integer.*

Proof sketch. To prove the theorem an exchange argument is used, which works as follows. Since $w^1 \equiv 1$, it holds that job $j \leq_L k$ if $p_j \leq p_k$. Furthermore $>_F$ respects $>_L$ and the jobs are indexed in reverse to $>_F$. Hence it holds that $p_1 \geq p_2 \geq \cdots \geq p_n$. Suppose x is a fractional optimal solution of $(\text{LP})_{\text{KK}}$. Let j be the smallest job index with fractional values of x and let ρ be the smallest position such that $x_{j,\rho} > 0$. If there would be no other job $k \neq j$ such that $0 < x_{k,\rho} < 1$ holds, then it would be possible to define a new feasible solution $\bar{x}$ with $\bar{x}_{j,\rho} = 1$, $\bar{x}_{j,\rho'} = 0 \quad \forall \rho' \neq \rho$ and $\bar{x}_{a,b} = x_{a,b} \quad \forall a \neq j$. Since $c(\bar{x}) < c(x)$ would hold, there must be a job k with $0 < x_{k,\rho} < 1$.
Furthermore there must be a position $\rho' > \rho$ such that $0 < x_{j,\rho'} < 1$, since constraint (1.1) holds for x. Now the job j on position ρ' is changed with job k on position ρ with the maximal possible fraction, such that the new solution $\bar{x}$ stays in the bound of $0 \leq \bar{x} \leq 1$. More formally a new solution $\bar{x}$ is defined by $\bar{x}_{j,\rho} := x_{j,\rho} + \epsilon$, $\bar{x}_{k,\rho} := x_{k,\rho} - \epsilon$, $\bar{x}_{j,\rho'} := x_{j,\rho'} - \epsilon$ and $\bar{x}_{k,\rho'} := x_{k,\rho'} + \epsilon$ as well as $\bar{x}_{a,b} := x_{a,b} \quad \forall (a,b) \notin \{(j,\rho),(k,\rho),(j,\rho'),(k,\rho')\}$. The value ϵ is chosen maximal such that $0 \leq \bar{x} \leq 1$ holds in any component of $\bar{x}$.
Now the proof states $\bar{x}$ to be a feasible solution for $(\text{LP})_{\text{KK}}$ with objective value at least as low as x.

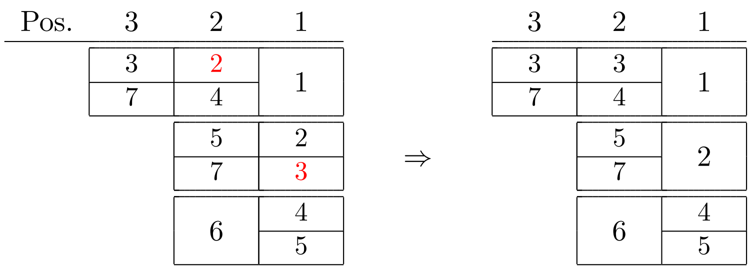
While the second part of the statement is true, $\bar{x}$ is not necessarily feasible as the following example shows:

Example 1.12 (Counter example for proof of Kis and Kovács)**.** Suppose $p \equiv 1$. In this case all feasible solutions are optimal, that schedule the jobs on the smallest possible positions.

This holds also for the two feasible integer solutions given below and the feasible fractional solution resulting from building the convex combination of both, while each solution has a value of $\frac{1}{2}$.



When applying the exchange argument of the wrong proof, the resulting coefficients are $j = 2$, $\rho = 1$, $k = 3$, $\rho' = 2$. The marked fraction of job 2 moves from position 2 to 1 and the marked fraction of job 3 moves the opposite way:



The resulting schedule is *not* feasible for $(\mathrm{LP})_{\mathrm{KK}}$, since constraint (1.3) is violated for indices $l = \rho = 2$:

$$0 = x_{2,2} \overset{!}{\geq} x_{3,3} = \frac{1}{2} \quad \Rightarrow \quad \lightning$$

The same contradiction appears, if the job exchange is applied on one of the other possible jobs to be chosen for k, namely 4 or 5.

In part 3.1 of this thesis it will be analyzed, if the proof of Theorem 1.11 can be repaired or if the statement is not correct.

Based on the LP relaxation of $(\mathrm{IP})_{\mathrm{KK}}$, Kis and Kovács [KisKov2012] provide a heuristic for the BWCTP. In this, a set of jobs on consecutive positions starting at position 1 is chosen, which is ordered ascending by index for ascending positions and whose corresponding variables have a positive and maybe fractional value in the LP solution. This jobset is then scheduled on one machine and the jobs as well as the machine are removed from $(\mathrm{LP})_{\mathrm{KK}}$. Now the next machines are filled by iteratively applying this procedure, until only one machine and a final set of unscheduled jobs is left. With scheduling all the remaining jobs to the last machine, the solution is finally build. After proving the correctness of the algorithm, a computational evaluation of the heuristic is given in the paper.

The results section for the BWCTP is concluded by a remark on the existence of an FPTAS for the BWCTP with fixed number of machines m based on the FPTAS given by Schuurman and Woeginger [SchWoe2001] for the one-level case, see Section 1.2.1. This remark will be checked in part 2.2 of this thesis.

2. Natural approximation approaches

It has been shown in the paper of Kis and Kovács [KisKov2012], that the general BWCTP is $\mathcal{NP}$-hard. Even for the special case of $p \equiv 1$, no polynomial time algorithm is known to solve it. Therefore this chapter presents some first approaches to obtain approximation algorithms for the BWCTP.

In the beginning of the chapter, some preliminary findings for the BWCTP are given. The following first approximation approach results in an FPTAS for the BWCTP with fixed number of machines. After this, we will present different list based algorithms. The chapter is concluded by a section, in which the generated solutions for the BWCTP will be based on solutions of the one-level WCTP.

2.1. Preliminary results

In this first section of the chapter, some general results for the BWCTP are given, that form the basis for some theorems being presented later in this thesis. The first of them regards the relationship between the follower's weights w^2 and the follower's order $>_F$, defined in Section 1.2.2.

Lemma 2.1. *Given an instance of the BWCTP and some arbitrary total order $>_A$ on the jobs. Then there exist weights $w^2(A)$ for the follower such that for the BWCTP with $w^2 := w^2(A)$ it holds, that $>_F$ is equivalent to $>_A$.*

Proof. Let the jobs be indexed according to $>_A$. We define the follower's weights as $w_j^2(A) := (n+1-j)p_j$. Hence it holds that $r_j = \frac{w_j^2(A)}{p_j} = n+1-j$. Since $>_F$ is ordered descending by r, the result follows. $\square$

The weight function $w^2(A)$ was already used by Kis and Kovács [KisKov2012], while justifying the assumption of $>_F$ being a total order.

Now we can prove the following result.

Theorem 2.2. *An instance I of the BWCTP with the processing times $p \equiv 1$, leader weights w^1 and follower order $>_F$ is equivalent to an instance $\bar{I}$ of the BWCTP having* *the leader weights $\bar{w}^1 \equiv 1$, processing times $\bar{p} \equiv w^1$ and follower order $\bar{>}_F$ reversed to the follower order $>_F$ of instance I. Hence the special case of the BWCTP with $p \equiv 1$ is equivalent to the special case of the BWCTP with $w^1 \equiv 1$.*

Proof. Let I be an instance of the BWCTP as specified and let A be a feasible assignment of the jobset J to the set of machines $\mathcal{M}$, such that each job is scheduled on exactly one machine. We denote by J_i the set of jobs being scheduled on machine $i \in \mathcal{M}$ in A. Furthermore we denote by S the schedule for I, resulting from the assignment A and the given job order of the follower $>_F$ on the machines, with completion times C_j for each job $j \in J$. Then the objective function value $c_I(S)$ of instance I and schedule S can be stated as:

$$\begin{aligned} c_I(S) = \sum_{j \in J} w_j^1 C_j &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} w_j^1 \sum_{k \in J_i : k \leq_F j} p_k \\ &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} w_j^1 \cdot |\{k \in J_i : k \leq_F j\}| \end{aligned}$$

Now suppose $\bar{I}$ to be an instances as specified. Such an instance with reversed follower order exists because of Lemma 2.1. Let $\bar{S}$ be the schedule with completion times $\bar{C}$ resulting from applying assignment A to instance $\bar{I}$. Then the corresponding objective function value $c_{\bar{I}}(\bar{S})$ of instance $\bar{I}$ for schedule $\bar{S}$ is:

$$\begin{aligned} c_{\bar{I}}(\bar{S}) = \sum_{j \in J} \bar{w}_j^1 \bar{C}_j &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} \bar{C}_j \\ &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} \sum_{k \in J_i : k \bar{\leq}_F j} \bar{p}_k \\ &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} \bar{p}_j \cdot |\{k \in J_i : k \bar{\geq}_F j\}| \\ &= \sum_{i \in \mathcal{M}} \sum_{j \in J_i} w_j^1 \cdot |\{k \in J_i : k \leq_F j\}| = c_I(S) \end{aligned}$$

Thus we can conclude that each feasible assignment A of jobs to machines has the same objective function value in I and $\bar{I}$. Therefore these instances are equivalent. As we already stated above we can find for each instance I for the BWCTP with $p \equiv 1$ an equivalent instance $\bar{I}$ of the BWCTP with $w^1 \equiv 1$. Since the same holds also for the opposite direction, these special cases of the BWCTP are equivalent. □

From the theorem we can directly conclude the following statements:

Corollary 2.3. *For the special case of $p \equiv 1$ for the BWCTP the following holds:*

(1) If $>_F$ respects $>_L$, then the BWCTP is polynomial solvable.

(2) If the reversed follower order $\bar{>}_F$ respects $>_L$, then the BWCTP is polynomial solvable. Furthermore there is an optimal solution S with the following structure, if we assume the jobs to be index according to $>_F$:

For the jobset J_i of jobs being scheduled on machine i in S, it holds $J_i = \{\pi_{i-1}+1, \pi_{i-1}+2, \dots, \pi_i\}$ with $\pi_0 := 0$ and $1 \leq \pi_1 \leq \pi_2 \leq \dots \leq \pi_m = n$.

(3) By replacing p by w^1 in the objective function of (IP)$_{KK}$ and counting the positions starting from the first job on each machine, the resulting IP model (IP)$_{\mathrm{p}}$ is an exact description of the BWCTP with $p \equiv 1$. All results for (IP)$_{KK}$ are also valid for (IP)$_{\mathrm{p}}$.

Proof. For proving (1) and (2), we observe that both statements hold for the BWCTP with $w^1 \equiv 1$ [KisKov2012]. Now look at an instance I of the BWCTP, in which $>_F$ respects $>_L$. For the corresponding equivalent instance $\bar{I}$ of the BWCTP it holds according to the above Theorem 2.2, that the job order of the follower $\bar{>}_F$ is reversed to $>_F$. Furthermore it holds, that $\bar{p} \equiv w^1$ and $\bar{w}^1 \equiv p$. Thus also $\bar{>}_L$ is reversed to $>_L$ and $\bar{>}_F$ respects $\bar{>}_L$. Therefore $\bar{I}$ can be solved in polynomial time as stated above, which solves I and proves (1).

The same argument could be applied for the proof of the polynomial solvability of (2). To construct an optimal solution of the given structure, we could simply use the optimal solution S of instance $\bar{I}$, since the assignment of jobs to machines is preserved. We only have to re-index the machines reversed to the indexing in S, since the job indexing is defined on the bases of the follower's order.

To prove the third statement of the corollary we consider an instance $\bar{I}$ of the BWCTP with $\bar{w}^1 \equiv 1$. Now we claim that (IP)$_{\mathrm{p}}$ is an exact description of the equivalent instance I of the BWCTP with $p \equiv 1$. Therefore we recall, that the job positions in (IP)$_{\mathrm{KK}}$ are counted backwards on the machines, that means that the last job on a machine has position 1, the penultimate job has position 2 and so on. Furthermore the jobs are indexed according to the order reversed to the order $\bar{>}_F$ of the follower in $\bar{I}$, that is, according to $>_F$.

If we now count the positions starting from the first position, as supposed in (3), we can conclude for instance I, that if two jobs j and k with $j < k$ are scheduled on one machine, it holds that the position of j has to be smaller than the one of k. That is exactly the property guaranteed by the constraints of (IP)$_{\mathrm{KK}}$ and therefore this is also ensured by the constraints of (IP)$_{\mathrm{p}}$.

To finally prove the correctness of $(\mathrm{IP})_{\mathrm{p}}$, we have to show that the objective function is correctly chosen. Therefore note that the following was shown in the proof of Theorem 2.2:

$$\sum_{i\in\mathcal{M}}\sum_{j\in J_i}\bar{p}_j\cdot|\{k\in J_i : k \mathrel{\bar{\geq}_F} j\}| = \sum_{i\in\mathcal{M}}\sum_{j\in J_i} w_j^1\cdot|\{k\in J_i : k\leq_F j\}|$$

The term $|\{k\in J_i : k \mathrel{\bar{\geq}_F} j\}|$ on the left hand side coincides exactly with the position definition for some job j in $(\mathrm{IP})_{\mathrm{KK}}$, while the position definition in $(\mathrm{IP})_{\mathrm{p}}$ can be expressed by the term $|\{k\in J_i : k\leq_F j\}|$ for job j. Therefore the objective function values of the IP models applied on the instance I respectively $\bar{I}$ coincide. Since this also holds also for feasible solutions S and $\bar{S}$ of the instances and since $(\mathrm{IP})_{\mathrm{KK}}$ was an exact description of instance $\bar{I}$, $(\mathrm{IP})_{\mathrm{p}}$ is an exact description of I.

Note, that the models for both special cases have the exact same structure. Hence the corresponding results for $(\mathrm{IP})_{\mathrm{KK}}$ can be adopted to $(\mathrm{IP})_{\mathrm{p}}$. $\square$

Because of the above proved equivalence of both special cases and the less complicated position numbering in the IP model, only the BWCTP with $p\equiv 1$ will be analyzed in this thesis.

As a last result of the section, the impact of different orders of the follower on the leader's objective function is investigated.

Theorem 2.4. *Given an instance of the BWCTP with follower's order $>_F$ and optimal solution S^*. Let S_L^* be the optimal solution of the BWCTP using the order of the leader as order for the follower, that is, $>_F:=>_L$, and let $S_{\bar{L}}^*$ be the optimal solution of the BWCTP using the* reversed *order of the leader $\bar{>}_L$ as order for the follower, that is, $>_F:=\bar{>}_L$. Then the following holds:*

$$c(S_L^*)\leq c(S^*)\leq c(S_{\bar{L}}^*)$$

To prove the theorem the following lemma is used.

Lemma 2.5. *Given an instance of the BWCTP with follower order $>_F$ and optimal solution S^*. Let j and k with $j<_F k$ be two neighboring jobs in the follower's order $>_F$, thus there is no job j' such that $j<_F j'<_F k$. Denote by $\hat{>}_F$ the follower's order resulting from flipping j and k in $>_F$. Hence it holds $k \mathrel{\hat{<}_F} j$. Let $\hat{S}^*$ be the corresponding optimal solution of the BWCTP with $>_F:=\hat{>}_F$. Then the following holds:*

(1) If $k\leq_L j$ then $c(\hat{S}^*)\leq c(S^*)$.

(2) If $k\geq_L j$ then $c(\hat{S}^*)\geq c(S^*)$.

Proof. In the proof of (1) the following cases are distinguished:

Case 1: The jobs j and k are scheduled on the same machine in S^*.
Since j and k are neighboring in $>_F$, they are also scheduled consecutive on the machine, that is, on neighboring positions. Then it is possible to build a feasible solution S for $\hat{>}_F$ by flipping j and k on the machine. This flip does not increase the objective function because $k \leq_L j$ is true. Since furthermore $\hat{S}^*$ is the optimal solution for the BWCTP with $>_F := \hat{>}_F$ it holds:

$$c(S^*) \geq c(S) \geq c(\hat{S}^*)$$

Case 2: The jobs j and k are scheduled on different machines in S^*.
Then S^* is also feasible for the new follower's order $\hat{>}_F$. Since again $\hat{S}^*$ is the optimal solution for the BWCTP with $>_F := \hat{>}_F$ it holds:

$$c(S^*) \geq c(\hat{S}^*)$$

To prove (2) assume the BWCTP with $\hat{>}_F$ is given. Then the flip of j and k in the follower's order can be applied in the other direction to obtain $>_F$ again. The result follows from using (1) on this case. □

Now it is possible to prove the theorem above.

Proof of Theorem 2.4. It is possible to obtain $>_L$ from $>_F$ by using only flips of the type (1) defined in Lemma 2.5 and also to obtain $\bar{>}_L$ by using only flips of type (2) defined in the Lemma by for example using bubble sort. Therefore we can conclude, that:

$$c(S_L^*) \leq c(S^*) \leq c(S_{\bar{L}}^*)$$

□

From the theorem it follows on the one hand, that if the follower's order $>_F$ is reversed to the job order $>_L$ of the leader, the corresponding optimal solution value is the biggest comparing the different job orders of $>_F$. Therefore, this case is worst possible for the leader. On the other hand, the best possible follower's order for the leader is his own "Smith's Rule" order of jobs. Note that in this case the bilevel problem is equivalent to the one-level case, see part 1.2.2. Hence the following holds:

Corollary 2.6. *Given an instance of the BWCTP and a corresponding optimal solution S^*. Let $\bar{S}^*$ be the optimal solution of the one-level WCTP using the leader's weights, that is, $w := w^1$. Then the following holds:*

$$c(\bar{S}^*) \leq c(S^*)$$

2.2. FPTAS for the BWCTP with fixed number of machines

As a first approach to solve the BWCTP we will show in the following section, that the FPTAS for the WCTP with a number $m = 2$ of machines described by Schuurman and Woeginger in [SchWoe2001] can be adopted to our case. After presenting the corresponding construction we will furthermore give an argument for the possible expansion of this approach to the case of an arbitrary fixed number m of machines. Note that Kis and Kovács already remarked this result in their paper [KisKov2012] without proving it.

To prove the polynomial execution time of the algorithm, the following observation is needed:

Observation 2.7. *Given an instance I of the BWCTP and let $p_{sum} := \sum_{j \in J} p_j$ and $w^1_{sum} := \sum_{j \in J} w^1_j$. Then for the input size $|I|$ of instance I it holds*

$$|I| > \log_2(p_{sum}) + \log_2(w^1_{sum}) = c(\ln(p_{sum}) + \ln(w^1_{sum}))$$

for some constant c.

The approximation scheme is based on a pseudo-polynomial time algorithm for the BWCTP with $m = 2$. The idea is to use this algorithm and restrict the number of examined solutions, to achieve a polynomial execution time. Afterwards we will show, that the objective value of the best solution found is within a factor of $(1 + \epsilon)$ of the possibly skipped optimal solution's value. Firstly the pseudo-polynomial algorithm is presented.

Pseudo-polynomial algorithm. For the pseudo-polynomial algorithm, a special encoding of a feasible solution is used. Each schedule is represented by a triple $[\mathcal{L}_1, \mathcal{L}_2, Z]$, where the last value Z denotes the objective value of the schedule and the $\mathcal{L}_i$ represent the load on machine i, that is, the sum of processing times of the jobs on this machine. The load on a machine can be bounded from above by the sum of processing times of all jobs p_{sum}. On the other hand the objective value of a schedule is derived from $\sum_{j \in J} w^1_j C_j$. Therefore it is bounded from above by $p_{sum} w^1_{sum}$, because each completion time is bounded by p_{sum} and hence it holds $\sum_{j \in J} w^1_j C_j \leq \sum_{j \in J} w^1_j p_{sum} = p_{sum} w^1_{sum}$. Because p and w^1 are positive integers, it follows that each triple $[\mathcal{L}_1, \mathcal{L}_2, Z]$ representing a solution is a vectors with integer coefficients in the three dimensional space $[0, p_{sum}] \times [0, p_{sum}] \times [0, p_{sum} w^1_{sum}]$, the solution space of the problem.

We assume the jobs to be indexed according to $>_F$ and denote by V_k the set of feasible subsolutions for the first k jobs. For each point $[x, y, z]$ in V_k, we save a list $S([x, y, z])$ of length at most n and with elements from $\{1, 2\}$. The i-th list element S_i denotes the machine on which job i is processed in the schedule represented by the corresponding point. In addition we denote by $S + a$ the list resulting from appending an element a to S. Then it is possible to define the pseudo-polynomial algorithm $\mathcal{A}$ as stated in Algorithm 1.

Algorithm 1: Pseudo-polynomial algorithm $\mathcal{A}$

```
1  point $P_1 \longleftarrow [p_1, 0, w_1^1 p_1]$;
2  point $P_2 \longleftarrow [0, p_1, w_1^1 p_1]$;
3  $V_1 \longleftarrow \{P_1, P_2\}$;
4  $S(P_1) \longleftarrow [1]$;
5  $S(P_2) \longleftarrow [2]$;
6  foreach job $k \in \{2, \ldots, n\}$ do
7      $V_k \longleftarrow \emptyset$;
8      foreach point $[x, y, z] \in V_{k-1}$ do
9          point $P_1 \longleftarrow [x + p_k, y, z + w_k^1(x + p_k)]$;
10         point $P_2 \longleftarrow [x, y + p_k, z + w_k^1(y + p_k)]$;
11         $S(P_1) \longleftarrow S([x, y, z]) + 1$;
12         $S(P_2) \longleftarrow S([x, y, z]) + 2$;
13         $V_k \longleftarrow V_k \cup \{P_1, P_2\}$;
14 return $[x^*, y^*, z^*]$ with $z^* = \min_{[x,y,z] \in V_n} z$;
```

To show the claimed properties of $\mathcal{A}$ we will prove the following theorem.

Theorem 2.8. *Algorithm $\mathcal{A}$ is an pseudo-polynomial algorithm for the BWCTP with $m = 2$ machines.*

Proof. We first prove that algorithm $\mathcal{A}$ finds the optimal solution.

For this propose we observe, that the algorithm constructs iteratively for each value $k \in \{1, \ldots, n\}$ the point set V_k. At the end of its execution, that point in V_n with the minimal objective value is returned, which is by the definition above the optimal feasible solution. The construction of the sets V_k is started with V_1, which contains the points corresponding to the possible schedules of job 1 on the two machines. Based on this each V_k is build from V_{k-1} by adding job k to each schedule in V_{k-1} on each of the two machines. Since $k >_F j$ holds for all $j < k$, the job k is added to the last position on the corresponding machines in each schedule. Therefore two new points in V_k can be generated from some point in V_{k-1} by adding p_k to the load L of the corresponding machine the job is scheduled on and increasing the objective function value by $w_k^1(L+p_k)$, where the value $(L+p_k)$ is equal to the completion time C_k of job k the machine. Thus, $\mathcal{A}$

considers all possibilities to schedule the n jobs on $m = 2$ machines and returns a point corresponding to an optimal schedule.

To evaluate the runtime of the algorithm, we observe that each iteration k of the algorithm runs in $O(|V_k|)$. Since $|V_k|$ is bounded from above by $O((p_{sum})^3 w^1_{sum})$ the total execution time of the algorithm is in $O(n \cdot ((p_{sum})^3 w^1_{sum}))$. Hence $\mathcal{A}$ is a pseudo-polynomial algorithm for the given problem. $\square$

Trimming of the solution space. The above presented pseudo-polynomial algorithm is now used to construct the FPTAS for the BWCTP with $m = 2$. The idea is to skip the adding of those solution points to V_k, for which there is already a "similar" solution in V_k. Hereby the execution time of the algorithm is reduced to be polynomial in the size of the input.

Therefore the solution space is divided into boxes produced by cuts in each dimension. The values, at which the cuts were applied, are Δ^i for $i = -1, 0, 1, 2, \ldots, L_d$ for each dimension d, where Δ is defined as $\Delta = 1 + \frac{\epsilon}{2n}$ and L_d will be calculated below. A point lying directly on a cut is considered to be part of both adjacent boxes. The values L_d for the biggest needed cuts can be bounded in the following way:

$$\begin{aligned} L_3 = \left\lceil \log_\Delta (p_{sum} w^1_{sum}) \right\rceil &= \left\lceil \frac{\ln(p_{sum} w^1_{sum})}{\ln(\Delta)} \right\rceil \\ &\leq \left\lceil (1 + \frac{2n}{\epsilon})(\ln(p_{sum}) + \ln(w^1_{sum})) \right\rceil \end{aligned}$$

By using the same arguments, it follows that $L_{1,2} \leq \left\lceil (1 + \frac{2n}{\epsilon}) \ln(p_{sum}) \right\rceil$. For the last inequality we use the following fact, which is based on the taylor series of $\ln(x)$:

$$\forall z > 1: \qquad \ln(z) = \sum_{n=1}^{\infty} \frac{1}{n} \left(\frac{z-1}{z} \right)^n > \frac{z-1}{z}$$

Then the argument below yields the used inequality:

$$\begin{aligned} \ln(z) \geq \frac{z-1}{z} \qquad \forall z > 1 \quad &\Rightarrow \qquad \ln(\Delta) \geq \frac{\frac{\epsilon}{2n}}{1 + \frac{\epsilon}{2n}} \\ &\Rightarrow \qquad 1/\ln(\Delta) \leq \frac{2n}{\epsilon}(1 + \frac{\epsilon}{2n}) = \frac{2n}{\epsilon} + 1 \end{aligned}$$

From this definition of the partition of the solution space the following statement can be concluded.

Observation 2.9. *Suppose the two integer points $x = [x_1, x_2, x_3]$ and $y = [y_1, y_2, y_3]$ are* *lying in the same box. Then for each dimension $i \in \{1, 2, 3\}$ it holds:*

$$x_i/\Delta \leq y_i \leq \Delta x_i$$

Note that if $x_i = 0$ or $y_i = 0$ holds, the other equality follows. This is true, because if $x_i = 0$ there cannot be another integer point y in the same box having $y_i > 0$, since the cut at Δ^{-1} separates the 0-level and the 1-level in each dimension.

Now the algorithm $\mathcal{A}^\#$ of the FPTAS can be defined. It works in the manner of the pseudo-polynomial algorithm $\mathcal{A}$ except for line 13. In contrast to the set of solutions V_k for iteration k of $\mathcal{A}$, the set $V_k^\#$ for $\mathcal{A}^\#$ only accepts one solution per box of the solution space.

Regarding the runtime of $\mathcal{A}^\#$, the execution time of each iteration k is polynomially bounded by $|V_k^\#|$, which is again bounded by the number of boxes. The number of boxes lies in $O(L_1 L_2 L_3)$ and so the total execution time of algorithm $\mathcal{A}^\#$ is an element of $O(nL_1 L_2 L_3)$. From the above given bounds of $L_{1,2,3}$ and Observation 2.7 finally follows that the runtime of $\mathcal{A}^\#$ is polynomially bounded in the size of the input as well as in $\frac{1}{\epsilon}$. Therefore the runtime satisfies the requirements of an FPTAS algorithm.

Proving a near optimal solution. As a last condition, that has to be checked for algorithm $\mathcal{A}^\#$ to be an FPTAS, it has to be shown that given the optimal solution value z of algorithm $\mathcal{A}$, the solution generated by algorithm $\mathcal{A}^\#$ has a solution value of $z^\# \leq (1+\epsilon)z$, where ϵ is bigger than 0 due to the Definition 1.5 of an approximation algorithm. To prove this, the following lemma is used.

Lemma 2.10. *For all solutions $[x, y, z] \in V_k$ in iteration k of algorithm $\mathcal{A}$ there is some solution $[x', y', z'] \in V_k^\#$ in iteration k of algorithm $\mathcal{A}^\#$ such that the following holds:*

$$x' \leq \Delta^k x \text{ , } y' \leq \Delta^k y \text{ and } z' \leq \Delta^k z$$

From the lemma follows, that there is a solution $[x', y', z'] \in V_n^\#$ such that for the solution $[x, y, z] \in V_n$ with a minimal value in z it holds $z' \leq \Delta^n z$. Now we use the inequality $f(z) := (1 + \frac{z}{n})^n \leq 1 + 2z$, which is true for $0 \leq z \leq 1$. To show this, we first observe that the second derivative of f is $f''(z) = \frac{n-1}{n}(1 + \frac{z}{n})^{n-2}$, which is non-negative for $0 \leq z \leq 1$ and thus f is convex in this interval. We define g to be the line through the points $f(0)$ and $f(1)$ and note that $(1 + \frac{1}{n})^n$ converges from below against $e < 3$ for positive $n \to \infty$. Then we follow for $0 \leq z \leq 1$:

$$\left(1+\frac{z}{n}\right)^n = f(z) \overset{\text{convexity}}{\leq} g(z) = \frac{f(1)-f(0)}{1-0}z + f(0)$$
$$= \left(\left(1+\frac{1}{n}\right)^n - 1\right)z + 1 < (3-1)z+1 = 2z+1$$

By choosing $z = \frac{\epsilon}{2}$, it can be concluded that:

$$\Delta^n = \left(1+\frac{\epsilon}{2n}\right)^n = \left(1+\frac{z}{n}\right)^n \leq 1+2z = 1+2\frac{\epsilon}{2} = 1+\epsilon$$

Hence for the optimal value $z^{\#}$ of $\mathcal{A}^{\#}$ it holds $z^{\#} \leq z' \leq (1+\epsilon)z$.

To finally verify, that $\mathcal{A}^{\#}$ is an FPTAS, the Lemma 2.10 has to be proved.

Proof of Lemma 2.10. To prove the lemma an induction on k is used. For the initial case of $k = 1$ let $[x, y, z]$ be a solution of V_1. If $[x, y, z] \in V_1^{\#}$ it is possible to choose $[x', y', z'] = [x, y, z]$.

If $[x, y, z] \notin V_1^{\#}$ there is some $[x', y', z'] \in V_1^{\#}$ lying in the same box as $[x, y, z]$. From Observation 2.9 it follows that $x' \leq \Delta x$, $y' \leq \Delta y$ and $z' \leq \Delta z$ holds, which are exactly the needed statements.

Now suppose as an induction hypothesis, that the statement is correct for $k-1$. Let $[x, y, z]$ be a solution in V_k. Without loss of generality it can be assumed, that in this solution job k is scheduled on machine 1. Therefore a precedent solution $[a, b, c]$ in V_{k-1} exists with:

$$[x, y, z] = [a + p_k, b, c + w_k(a + p_k)] \tag{2.1}$$

By using the induction hypothesis on $[a, b, c]$, a solution $[a', b', c'] \in V_{k-1}^{\#}$ can be found, such that it holds:

$$a' \leq \Delta^{k-1}a \text{ , } b' \leq \Delta^{k-1}b \text{ and } c' \leq \Delta^{k-1}c \tag{2.2}$$

Hence the solution $P = [a' + p_k, b', c' + w_k(a' + p_k)]$ is generated in iteration k of the algorithm $\mathcal{A}^{\#}$ and there exists a possibly different solution $[\alpha, \beta, \gamma] \in V_k^{\#}$ lying in the same box as P. Therefore it is possible to conclude the following inequalities:

$$\alpha \overset{\text{Obs. 2.9}}{\leq} \Delta(a' + p_k) \overset{(2.2)}{\leq} \Delta^k a + \Delta p_k \leq \Delta^k(a + p_k) \overset{(2.1)}{=} \Delta^k x$$
$$\beta \overset{\text{Obs. 2.9}}{\leq} \Delta b' \overset{(2.2)}{\leq} \Delta^k b \overset{(2.1)}{=} \Delta^k y$$
$$\gamma \overset{\text{Obs. 2.9}}{\leq} \Delta(c' + w_k(a' + p_k)) \overset{(2.2)}{\leq} \Delta^k c + w_k(\Delta(a' + p_k)) \overset{(2.2)}{\leq} \Delta^k(c + w_k(a + p_k)) \overset{(2.1)}{=} \Delta^k z$$

This proves the induction and therefore the lemma. □

The above given construction of an FPTAS is valid for the BWCTP with $m = 2$. This approach can be extended to an arbitrarily fixed value for m.

Theorem 2.11. *There is an FPTAS for the BWCTP with a fixed number of machines m.*

Proof. To adopt the construction given above to the case of a fixed number of machines m, we first have to extend the solution space to $[\mathcal{L}_1, \dots, \mathcal{L}_m, Z]$. Hence in the pseudo-polynomial algorithm $\mathcal{A}$ in iteration $k \in \{1, \dots, n\}$ the points $P_1, \dots, P_m$ have to be constructed and the k-th element of $S(P_i)$ is set to i. The trimming of the solution space works as described above, such that for $i \in \{1, \dots, m\}$ it holds $L_i \le \lceil (1 + \frac{2n}{\epsilon}) \ln(p_{sum}) \rceil$ and additionally $L_{m+1} \le \lceil (1 + \frac{2n}{\epsilon})(\ln(p_{sum}) + \ln(w^1_{sum})) \rceil$. For this reason the runtime of algorithm $\mathcal{A}^\#$ lies in $O(nL_1^m L_{m+1})$ and is therefore slightly different from the runtime stated by Kis and Kovács in [KisKov2012], which gave an upper bound of $O(nL^m)$, with $L := L_1$. For concluding the proof of the approximation ratio of $(1 + \epsilon)$ we notice, that Observation 2.9 and Lemma 2.10 could be extended to the case of m machines as well and that subsequent argumentation is regardless of the number of machines made for the objective function value. □

Adopt the PTAS for the general WCTP to the bilevel case. We have seen in Subsection 1.2.1, that even the general WCTP admits a polynomial time approximation scheme. The corresponding construction was presented by Skutella and Woeginger [SkuWoe2000]. They first provided an approximation scheme for instances of the WCTP with a constant range of job ratios $\frac{w_j}{p_j}$ for jobs $j \in J$. In other words they assumed an arbitrary $R > 0$ and a real constant $0 < \rho \le 1$ independent of the input, such that for all $j \in J$ it holds: $\frac{w_j}{p_j} \in [\rho R, R]$. By using this PTAS on a job partitioning into subsets of similar job ratios, they could build a PTAS for the general WCTP.

We will prove in the following theorem, that this approach cannot be adopted to the BWCTP.

Theorem 2.12. *The PTAS of Skutella and Woeginger [SkuWoe2000] for the one-level WCTP cannot be adopted to the BWCTP.*

Proof. In the proof of the PTAS for the WCTP, Skutella and Woeginger frequently use the fact, that the job order on the machines corresponds for an optimal solution of the problem to the "Smith's Rule" order based on the weights given in the objective function. This is not true in the bilevel case, since we have two different weight functions w^1 and w^2 given here: One to use in the objective function and one to specify the "Smith's Rule" job order on the machines. Therefore even the proved PTAS for a constant range of job ratios $\frac{w_j}{p_j}$ for jobs $j \in J$ cannot be adopted to the bilevel case using $w := w^1$.

To prove this we observe that Skutella and Woeginger use the modified but equivalent objective function $\sum_{j\in J} w_j\Gamma_j$ for the problem, with is defined as:

$$\sum_{j\in J} w_j\Gamma_j = \sum_{j\in J} w_jC_j - \frac{1}{2}\sum_{j\in J} w_jp_j$$

Since the negative term is constant for each schedule, the objective function is equivalent to $\sum_{j\in J} w_jC_j$.

Now they defined $\rho_1 > \rho_2 > \cdots > \rho_q$ to be the different job ratios $\frac{w_j}{p_j}$ for $j \in J$. For each $1 \leq h \leq q$ they denoted by $J_h := \{j \in J \,|\, \frac{w_j}{p_j} = \rho_h\}$ the set of all jobs with job ratio ρ_h. Furthermore for a given schedule S the jobset $J_{h,i}$ consists of those jobs of J_h, that are scheduled on machine $i \in \mathcal{M}$ in schedule S. Then it was followed that the modified objective value of the schedule S can be stated as:

$$\sum_{j\in J} w_j\Gamma_j = \frac{1}{2}\sum_{h=1}^{q}\rho_h\sum_{i=1}^{m}\left(\sum_{j\in J_{h,i}} p_j\right)^2 + \sum_{1\leq h<l\leq q}\rho_l\sum_{i=1}^{m}\left(\sum_{j\in J_{h,i}} p_j\right)\left(\sum_{j\in J_{l,i}} p_j\right)$$

The proof of the inequality in based on the assumption, that jobs with equivalent job ratios are scheduled consecutively on the machines, which is not true for the BWCTP. Therefore the inequality does not hold here. Since this description of the objective function value of a schedule S is used to prove the $(1+\epsilon)$-approximation ratio of the PTAS, this does not work for the bilevel case.

Furthermore the polynomial runtime of the PTAS is based on the fact, that in the WCTP each job j, whis is scheduled on the last position of a machine in an optimal schedule, starts at or before the completion time of each other last job on the other machines. Otherwise we could shift the job j to the last position of another machine and reduce its completion time. However, this is again not valid in the bilevel case, since here the job order on the machines is given by the follower. Therefore it might be optimal to distribute the job unequally to the machines, even if the processing times are equivalent for all jobs. We will see this behavior in many of the following examples in the thesis. For this two reasons we can conclude, that the PTAS of Skutella and Woeginger [SkuWoe2000] cannot be adopted to the BWCTP. $\square$

2.3. List based algorithms

In the previous section an FPTAS for the BWCTP with fixed number of machines m has been presented. Therefore we will look from now on for algorithms solving the BWCTP in polynomial time, when m is considered to be part of the input.

In a first approach of this kind we will study list based algorithms. Here a feasible schedule is build by first determining an order of the jobs and then constructing the solution by assigning the jobs in the given order to the machines. For the WCTP we have seen in Section 1.2.1, that a list based algorithm, the list scheduling algorithm, combined with the "Smith's Rule" order admits a constant approximation ratio. We will show in the first following subsection, that this result cannot be adopted to the case of the BWCTP. Furthermore we will prove that the list scheduling algorithm cannot have a constant approximation ratio, independent of the chosen job order. Therefore a second list based algorithm, the min-increase algorithm, will be presented in the second part of this section.

2.3.1. List scheduling

The first examined list based algorithm is the list scheduling algorithm. Here the jobs of the list will be subsequently scheduled on a machine with the current minimal load, that is, the machine with the smallest sum of processing times of jobs being scheduled on the corresponding machine so far. In the case of the one-level WCTP it was proved, that the list scheduling algorithm is a constant factor approximation algorithm, given a specific ordered job list. The used list was ordered according to "Smith's Rule" in this case, see Subsection 1.2.1. Since in an optimal solution of the WCTP the jobs on the machines have to be ordered according to that list, too, each new job is scheduled on the last position of a machine in the list scheduling algorithm. To achieve the same behavior also for the bilevel case, we have to order the job list used for the list scheduling algorithm according to the follower's order $>_F$. However, it can be proved, that the list scheduling algorithm using $>_F$ does not admit a constant approximation ratio for the BWCTP.

Theorem 2.13. *There cannot be a constant value $\alpha > 0$ for each instance I of the BWCTP, such that for the corresponding solution S of the list scheduling algorithm using $>_F$ and the optimal solution S^* of this instance I of the BWCTP holds:*

$$c(S) \leq \alpha c(S^*)$$

Proof. We prove the theorem by giving a corresponding example instance.

Example 2.14. Let I be an instance with 2 machines and $2n$ jobs indexed according to $>_F$ with unit length processing times $p \equiv 1$ and the following weights of the leader:

$$\begin{aligned} w_i^1 &= \epsilon \quad \forall i = 1, \dots, 2n-1 \\ w_{2n}^1 &= M \text{ with } M \gg \epsilon \end{aligned}$$

Since $M \gg \epsilon$ holds, it is optimal to schedule job $2n$ as early as possible and therefore on the first position of a machine. However, job $2n$ has the biggest index and the jobs are indexed according to $>_F$. Hence each job in $\{1, \dots, 2n-1\}$ would be scheduled in front of job $2n$ if being assigned to the same machine. Thus the below presented optimal schedule S^* for the BWCTP would assign job $2n$ to one machine and the other jobs to the other one.

Pos.	1	2	...	$2n-1$
	1	2	...	$2n-1$
	$2n$			

Therefore the optimal objective function value of the instance would be:

$$c(S^*) = M + \epsilon \cdot \frac{4n^2 - 2n}{2}$$

Now look at the list scheduling solution S. It processes the jobs with increasing indices and schedules each new job to the end of a machine. Since the processing times are all equal to 1, S has the following structure, up to permutations of the jobs on the same positions:

Pos.	1	2	...	$n-1$	n
	1	3	...	$2n-3$	$2n-1$
	2	4	...	$2n-2$	$2n$

The corresponding objective function value of S can be denoted as:

$$\begin{aligned} c(S) &= \overbrace{\epsilon \cdot 2(\frac{n \cdot (n-1)}{2})}^{\text{first } n-1 \text{ positions}} + \overbrace{M \cdot n + \epsilon \cdot n}^{\text{position } n} \\ &= M \cdot n + \epsilon \cdot n^2 \end{aligned}$$

Since $M \gg \epsilon$, there is no constant α such that $c(S) \leq \alpha c(S^*)$ holds for all values of n.

$\square$

From the theorem follows, that the proof of the 2-approximation ratio from the one-level case cannot holds for the BWCTP. Although the in the proof used upper bound $u = \sum_j w_j p_j + \frac{1}{m} \sum_j \sum_{k<j} w_j p_k$ for the objective function value $c(S)$ of the list scheduling solution S is still valid in the bilevel case when defining w as w^1, it cannot be bounded from above by $2 \cdot c(S^*)$. The reason for this is, that the second summand of u is no longer a lower bound to the optimal objective function value of the problem, since we adopted the job indices according to $>_F$.

The above proved result of Theorem 2.13 can be further extended to show that even list scheduling with an arbitrary given list of jobs does not admit a constant approximation ratio.

Theorem 2.15. *There cannot be a constant value $\alpha > 0$ for each instance I of the BWCTP, such that for the corresponding solution S of a list scheduling algorithm using an arbitrarily ordered list and the optimal solution S^* of this instance I of the BWCTP it holds:*

$$c(S) \leq \alpha c(S^*)$$

Proof. As in the proof of the previous theorem, we can use Example 2.14 to show the statement. Thus, the used instance and the corresponding optimal solution S^* of the BWCTP coincide with the specifications of the example.

To determine the list scheduling solution S, we observe, that the processing times of the jobs are all equal to 1. Since the list scheduling algorithm assigns the jobs to one of the machines with minimal load, the $2n$ jobs will be equally distributed to the two machines for each possible order of the job list. Furthermore the job $2n$ has the highest index and will therefore be scheduled behind all other jobs on its machine. Hence S is for every possible job list equivalent to the list scheduling result given in Example 2.14, apart from permutations of the jobs $1, \dots, 2n-1$. Thus S and S^* have the same objective value as in Example 2.14, which proves the theorem. □

2.3.2. Min-increase scheduling

We have seen in the previous subsection, that list scheduling does not seem to be very promising for achieving good approximation results for the BWCTP, which, at least for the in the proofs given counter Example 2.14, is based on the fact, that the algorithm only considers the current machine loads to decide where to put the next job. Therefore a more sophisticated way of assigning a new job to the machines might generate better results. This is exactly what the “min-increase” scheduling algorithm does, which is also a list based scheduling algorithm. Here, for each new job j and each machine i the increase of the objective function value is calculated, that scheduling job j on machine i would cause. Then the job is assigned to that machine with the minimal cost increase.

As a first result we observe, that the min-increase algorithm together with a job list ordered according to $>_L$ produces the optimal solution for Example 2.14. The reason for this is, that job $2n$ with a heavy weight of the leader would be scheduled first and therefore on a single machine. Each subsequent job would then be scheduled to the other machine, which is cheaper than shifting the heavy job one position behind.

In addition to the job order $>_L$ using "Smith's Rule" with the weights of the leader and $>_F$ using "Smith's Rule" with the weights of the follower, we define the following two job orders for the min-increase algorithm, which for each job considers both defined weights. For the *sum of weighted processing times order* $>_+$ it holds for jobs j and k, that:

$$j <_+ k :\Leftrightarrow \frac{w_j^1}{p_j} + \frac{w_j^2}{p_j} > \frac{w_k^1}{p_k} + \frac{w_k^2}{p_k}$$

Additionally the *product of weighted processing times order* $>_\times$ is defined for jobs j and k as:

$$j <_\times k :\Leftrightarrow \frac{w_j^1}{p_j} \cdot \frac{w_j^2}{p_j} > \frac{w_k^1}{p_k} \cdot \frac{w_k^2}{p_k}$$

However, we can prove the following theorem for the min-increase algorithm using job lists sorted according to $>_+$ and $>_\times$ as well as to the leader's job order $>_L$ and the follower's job order $>_F$.

Theorem 2.16. *Let $>_x \in \{>_L, >_F, >_+, >_\times\}$ be one of the defined job orders. Then it holds, that there cannot be a constant value $\alpha > 0$ for each instance I of the BWCTP, such that for the corresponding solution S_x of the min-increase algorithm using a job list ordered according to $>_x$ and the optimal solution S^* of this instance I of the BWCTP holds:*

$$c(S_x) \leq \alpha c(S^*)$$

Proof. We first note that in the case of sorting the job list according to $>_F$, the min-increase algorithm acts exactly like list scheduling for the same job list. Here, each new job j will be scheduled on the last position of each machine. For this reason the increase of the objective value can be stated as $w_j^1 C_j$. Since w_j^1 is equal for each of the machines, j would be scheduled on that machine minimizing the resulting completion time C_j. This is equal to choosing the machine with the minimal load, since the processing time of job j is the same on each machine. Hence the given statement follows for the follower's job order $>_F$ from Theorem 2.13.

For the other job orders we use the following example instance to prove the theorem.

Example 2.17. Let I be an instance with again 2 machines and $2n$ jobs having identical processing times of $p \equiv 1$. Then the following leader's and follower's weights are defined:

$$w_i^2 = 2n + 1 - i \quad \forall i = 1, \dots, 2n$$
$$w_i^1 = \epsilon \quad \forall i = 1, \dots, 2n - 2$$
$$w_{2n-1}^1 = w_{2n}^1 = M \text{ with } M \gg n \text{ and } M \gg \epsilon$$

By choosing the given processing times and follower's weights, we ensure, that the job indices are sorted according to $>_F$.

In the optimal schedule the heavy weighted jobs would be scheduled together on one machine, while the light weighted jobs would be assigned to the other one. This is due to the argument already used in Example 2.14, stating that each light weighted job would shift a heavy weighted job one position behind, if being scheduled on the same machine. Since the heavy weighted jobs have a much bigger impact on the objective function value, this will be avoided. Thus, the optimal schedule S^* looks as follows:

Pos.	1	2	...	$2n-2$
	1	2	...	$2n-2$
	$2n-1$	$2n$		

The optimal solution's objective value is then given by $c(S^*) = M \cdot 3 + \epsilon \cdot \frac{(2n-1)\cdot(2n-2)}{2}$.

Now look at the min-increase algorithm for the job order $>_x \in \{>_L, >_+, >_\times\}$. For all three job lists sorted by $>_x$ it holds that the jobs $2n-1$ and $2n$ precede the other jobs in the orders, because $M \gg n$ and $M \gg \epsilon$ holds. Therefore they will be assigned to the two machines first, followed by the other jobs with smaller weight. Hence the min-increase algorithm will schedule the jobs $2n$ and $2n-1$ on different machines.

Now each of the other jobs $j \in \{1, \dots, 2n-2\}$ will be scheduled in front of one of the heavy weighted jobs, causing an increase of the objective function value of M. Because of this, the objective function value increase caused by the shift of the jobs $2n$ or $2n-1$ can be neglected, when choosing the machine for job j. Since all the jobs $j \in \{1, \dots, 2n-2\}$ have a leader's weight of $w_j^1 = \epsilon$, they will be equally distributed over the two machines. Therefore each of the schedules S_x produced by the min-increase algorithm using a job list sorted according to $>_x \in \{>_L, >_+, >_\times\}$ respectively has the same objective function value as the following schedule S:

Pos.	1	2	...	$n-1$	n
	1	3	...	$2n-3$	$2n-1$
	2	4	...	$2n-2$	$2n$

Hence for the objective values of the schedules it holds:

$$\begin{aligned} c(S_x) = c(S) &= 2(\epsilon \cdot \frac{n \cdot (n-1)}{2} + M \cdot n) \\ &= M \cdot 2n + \epsilon \cdot (n^2 - n) \end{aligned}$$

Therefore it can again be concluded that there is no constant α such that for all n the following inequality holds $c(S) \leq \alpha c(S^*)$, since $M \gg \epsilon$.

□

Although it was shown above, that even for the $p \equiv 1$ case of the BWCTP min-increase scheduling can be arbitrarily bad for the job lists sorted according to $>_F$, $>_L$, $>_+$ and $>_\times$, it is not clear if there is an arbitrarily bad example for each existing job list. Another possible result, which would exclude the existence of an arbitrarily bad example for each job list, is, that there is an order of jobs for each instance of the BWCTP, such that the min-increase solution produced by using this list is only a constant factor away from the optimal solution or even optimal itself.

For each of the two examples given above exists such a job order, which produces the optimal schedule S^*. Given Example 2.14, sorting the jobs by $>_L$ would produce the optimal solution, as already argued above. For Example 2.17, suppose that job $2n-1$ is assigned to one machine first. If afterwards the jobs $1, \dots, 2n-2$ would be assigned, they would all be scheduled on the other machine as described above for the optimal solution of Example 2.14. At last, job $2n$ would be scheduled to the end of one of the machines and therefore would be assigned to the machine having the smallest current load. Therefore it would choose the same machine as job $2n-1$, resulting in S^*.

Apart from this theoretical study of the min-increase algorithm, it is also evaluated empirically on a set of random instances in Chapter 4 using the different job orders.

2.4. Approximation based on solving the WCTP

For the general proof of some worst case ratio α of an approximation algorithm, we have to show that the objective value of the algorithm's solution is at most a factor of α worse than the corresponding optimal solution, see Definition 1.5 of an α-approximation algorithm. In contrast to the specific instances of the Examples 2.14 and 2.17 in the previous section, the optimal solution of the BWCTP is generally not known. Therefore a lower bound L to the objective value of the optimal solution S^* is needed to compare the approximative solution value against it. Suppose we could prove a approximation ratio of α for the approximative solution S by comparing it against such a lower bound L. That means, we could prove, that the following holds:

$$c(S) \leq \alpha L \leq \alpha c(S^*)$$

If we now denote by β the gap between this lower bound L and the objective value of the optimal solution $c(S^*)$, it is possible to deduce:

$$\beta = \frac{c(S^*)}{L} \leq \frac{c(S)}{L} \leq \frac{\alpha L}{L} = \alpha$$

Therefore each approximation ratio α has to be at least as big as the gap between the lower bound and the optimal solution value. Hence it is essential for finding good approximation results with small values of α, to know lower bounds near to the optimal solution value.

In the following section we will presents some approaches for approximation algorithms, that uses the same lower bound, namely the optimal solution of the one-level WCTP using weights $w := w^1$. That the optimal solution for the one-level problem actually is a lower bound for the optimal solution of the BWCTP has been shown by Corollary 2.6 in Section 2.1.

The approaches of this section will all work in a similar way. Given an instance of the BWCTP, we will solve a possibly different instance of the one-level WCTP and construct a feasible solution of the BWCTP from it. Since it is already $\mathcal{NP}$-hard to solve the WCTP, the PTAS suggested by Skutella and Woeginger [SkuWoe2000] is used to generate a near-optimal solution having an approximation ratio of $(1+\epsilon)$, with $\epsilon > 0$. This factor of $(1+\epsilon)$ will then also be involved in the approximation ratios of the presented algorithms.

In addition to the algorithmic approaches, the section will also analyze, if the WCTP lower bound is near to the optimal value of the BWCTP in general.

Max ratios between w^1 and w^2. The first approach is targeted at those instances having similar weights w^1 of the leader and w^2 of the follower. Therefore we will prove an approximation ratio based on the following two values:

$$r_{12} := \max_{j \in J} \left\{ \frac{w_j^1}{w_j^2} \right\}$$
$$r_{21} := \max_{j \in J} \left\{ \frac{w_j^2}{w_j^1} \right\}$$

From these definitions it directly follows, that:

$$r_{12} \geq \frac{w_j^1}{w_j^2} \quad \forall j \in J \Leftrightarrow w_j^1 \leq r_{12} w_j^2 \quad \forall j \in J \tag{2.3}$$

$$r_{21} \geq \frac{w_j^2}{w_j^1} \quad \forall j \in J \Leftrightarrow w_j^1 \geq \frac{w_j^2}{r_{21}} \quad \forall j \in J \tag{2.4}$$

To show the approximation, we need to consider instances with different objective function values for both, the WCTP and the BWCTP. Hence the following notation is introduced for some schedule S with completion times C:

$$c^1(S) := \sum_{j \in J} w_j^1 C_j$$
$$c^2(S) := \sum_{j \in J} w_j^2 C_j$$

With this definition it is possible to prove the following lemma:

Lemma 2.18. *Let S^* be an optimal solution of the BWCTP with objective function c^1 and let $\bar{S}^*$ be an optimal solution of the one-level WCTP using the objective function c^2. Then the following holds:*

$$c^1(S^*) \geq \frac{1}{r_{21}} c^2(\bar{S}^*)$$

Proof. We denote by C^* the job completion times in S^* and note that S^* is also a feasible solution for the one-level WCTP using $w := w^2$. Then it is possible to conclude the followings:

$$\begin{aligned} c^1(S^*) &= \sum_{j \in J} w_j^1 C_j^* \overset{(2.4)}{\geq} \sum_{j \in J} \frac{w_j^2}{r_{21}} C_j^* \\ &= \frac{1}{r_{21}} \sum_{j \in J} w_j^2 C_j^* = \frac{1}{r_{21}} c^2(S^*) \geq \frac{1}{r_{21}} c^2(\bar{S}^*) \end{aligned}$$

□

Summing up the above obtained results it is now possible to finally prove the following theorem:

Theorem 2.19. *Consider the WCTP using the objective function c^2 and denote by $\bar{S}^*$ the optimal solution of this problem. Let the schedule S' be the solution generated by the PTAS for this version of the WCTP. Then we construct a feasible approximative schedule S^a with job completion times C^a for the BWCTP using the objective function c^1 by reordering the schedule S' on each machines, such that the job order corresponds to $>_F$.*

The schedule S^a is a $(1+\epsilon)r_{12}r_{21}$-approximation for the BWCTP, since for the optimal solution S^* of the BWCTP holds:

$$c^1(S^a) \leq (1+\epsilon)r_{12}r_{21}c^1(S^*)$$

Proof. To prove the theorem we first observe, that for the BWCTP with $m = 1$ and an objective function c^2 using the follower's weights w^2, it is optimal to sort the jobs according to $>_F$. Hence it holds that:

$$c^2(S^a) \leq c^2(S') \tag{2.5}$$

From this result we can conclude:

$$\begin{aligned} c^1(S^a) &= \sum_{j\in J} w_j^1 C_j^a \overset{(2.3)}{\leq} \sum_{j\in J} r_{12} w_j^2 C_j^a \\ &= r_{12}c^2(S^a) \overset{(2.5)}{\leq} r_{12}c^2(S') \\ &\leq r_{12}(1+\epsilon)c^2(\bar{S}^*) \overset{\text{Lemma 2.18}}{\leq} r_{12}(1+\epsilon)r_{21}c^1(S^*) \end{aligned}$$

□

Max ratios between w^1 and p. A very similar result to the one given above can be achieved, when comparing the weights w^1 of the leader and the processing times p. In this case, the worst case ratio of the later presented approximative solution will depend on the following values:

$$\begin{aligned} r_{1p} &:= \max_{j\in J} \left\{ \frac{w_j^1}{p_j} \right\} \\ r_{p1} &:= \max_{j\in J} \left\{ \frac{p_j}{w_j^1} \right\} \end{aligned}$$

As in the approach above, we can directly conclude the following statements:

$$r_{1p} \geq \frac{w_j^1}{p_j} \quad \forall j \in J \Leftrightarrow w_j^1 \leq r_{1p}p_j \quad \forall j \in J \tag{2.6}$$

$$r_{p1} \geq \frac{p_j}{w_j^1} \quad \forall j \in J \Leftrightarrow w_j^1 \geq \frac{p_j}{r_{p1}} \quad \forall j \in J \tag{2.7}$$

In addition to the already defined objective functions, we will need the following one for some schedule S with job completion times C.

$$c^p(S) := \sum_{j \in J} p_j C_j$$

Now it is possible to deduce the lemma below, which corresponds to Lemma 2.18 of the previous approach.

Lemma 2.20. *Let S^* refer again to an optimal solution of the BWCTP with objective function c^1 and let $\bar{S}^*$ be an optimal solution of the one-level WCTP using the objective function c^p. Then it holds that:*

$$c^1(S^*) \geq \frac{1}{r_{p1}} c^p(\bar{S}^*)$$

Proof. Using the inequality (2.7), the proof follows the exact same line of argumentation as in Lemma 2.18. □

With this preliminary results, it is now possible to prove the following theorem.

Theorem 2.21. *Let $\bar{S}^*$ be an optimal solution of the one-level WCTP with the objective function c^p as stated in Lemma 2.20. Since the WCTP is still $\mathcal{NP}$-hard in this case as mentioned in the presentation of the results for the WCTP in Section 1.2.1, we will again denote by S' the corresponding near-optimal PTAS solution for the same problem. As in Theorem 2.19 we obtain the approximative solution S^a from solution S' by reordering the jobs on the machines according to $>_F$.*

Then S^a is a $(1+\epsilon)r_{1p}r_{p1}$-approximation for the BWCTP, since for the optimal solution S^* of the BWCTP holds:

$$c^1(S^a) \leq (1+\epsilon) r_{1p} r_{p1} c^1(S^*) \tag{2.8}$$

Proof. We first note that the following equation holds:

$$c^p(S^a) = c^p(S') \tag{2.9}$$

It follows from the fact, that in the case of $w \equiv p$ the “Smith's Rule” ratio of each job is 1 and therefore the value of the objective function c^p is independent from the order of the jobs on the machines.

With this result, we can prove the theorem as follows:

$$\begin{aligned} c^1(S^a) &= \sum_{j\in J} w_j^1 C_j^a \overset{(2.6)}{\leq} \sum_{j\in J} r_{1p} p_j C_j^a \\ &= r_{1p} c^p(S^a) \overset{(2.9)}{=} r_{1p} c^p(S') \\ &\leq r_{1p}(1+\epsilon) c^p(\bar{S}^*) \overset{\text{Lemma 2.20}}{\leq} r_{1p}(1+\epsilon) r_{p1} c^1(S^*) \end{aligned}$$

□

Ratio between one-level and bilevel optimum. The last presented approximation approach, that is based on the one-level optimal solution lower bound, will depend directly on the above defined ratio β between the optimal solution value and the value of the corresponding lower bound. Given the optimal solution $\bar{S}^*$ of the one-level case and the optimal bilevel solution S^*, we could describe β in our case as:

$$\beta = \frac{c(S^*)}{c(\bar{S}^*)}$$

Therefore it follows that:

$$c(S^*) = \beta c(\bar{S}^*) \tag{2.10}$$

The desired result would be, that β can be proved to be bounded from above by a constant value for all instances of the BWCTP.

For the single machine case $m = 1$, it is easy to see that this is not true. Assume a situation with one heavy weighted job j of leader weight $w_j^1 = M$ and processing time $p_j = 1$ and many light weighted jobs with leader weight ϵ, which also have processing times of 1. If $M \gg \epsilon$, the optimal solution in the one-level case would have j scheduled in front of the light jobs, according to "Smith's Rule". Therefore the optimal solution value would roughly be M. In the bilevel case, the follower may value the light jobs much higher than job j. Thus j might be scheduled on the last position in this case. This solution would have an objective value bigger than $n \cdot M$. Hence there is no constant upper bound on β for every number $n-1$ of light jobs.

In the given example, it was not possible for the leader to react to the given job order of the follower, since there is only one machine to schedule the jobs on. Therefore a constant upper bound on β might be possible for instances with $m \geq 2$.

Before discussing the possibilities to prove this upper bound to be small, this goal is motivated by presenting the above announced approximation algorithm with worst case bound dependent on β. It is based on the PTAS of Skutella and Woeginger [SkuWoe2000] and on the FPTAS for the BWCTP with a fixed number m of machines presented in Section 2.2. The algorithm called $\mathcal{A}^\beta$ is stated in Algorithm 2 and will be proved to have the stated approximation ratio in the following theorem.

Algorithm 2: $\beta(1+\epsilon)^2$-approximation algorithm $\mathcal{A}^\beta$

```
1 $\bar{S}^\epsilon \longleftarrow$ PTAS solution for the WCTP with $w := w^1$;
2 Match the machines to pairs $M_i$ (and one triple if $m$ is odd);
3 foreach set of machines $M_i$ do
4     $\bar{S}_i^\epsilon \longleftarrow$ subschedule of $\bar{S}^\epsilon$ on $M_i$;
5     $J_i \longleftarrow$ set of jobs scheduled in $\bar{S}_i^\epsilon$;
6     $S_i^\epsilon \longleftarrow$ FPTAS solution for the BWCTP on $|M_i|$ machines for jobset $J_i$;
7 $S \longleftarrow$ solution of the BWCTP formed by combining the $S_i^\epsilon$;
8 return $S$;
```

In algorithm $\mathcal{A}^\beta$ we first construct a solution $\bar{S}^\epsilon$ for the corresponding WCTP with $w := w^1$ by using the mentioned PTAS. After this the machines are matched to pairs M_i. Note that we extend one pair to a triple in the case of an odd number of machines m. Furthermore we denote by $\bar{S}_i^\epsilon$ the subschedule of $\bar{S}^\epsilon$ on the machine set M_i. Now we define a new instances i for the BWCTP with a number $|M_i|$ of machines and the jobset J_i consisting of those jobs scheduled on the machines in M_i. The parameters w^1, w^2 and p and therefore also the follower's order $>_F$ remain the same for the corresponding jobs. For this instances i we create solutions S_i^ϵ for the BWCTP by using the mentioned FPTAS. The solution S for the original instance of the BWCTP is finally build by combining the subsolutions S_i^ϵ.

For algorithm $\mathcal{A}^\beta$ we can prove the following theorem.

Theorem 2.22. *Algorithm $\mathcal{A}^\beta$ is a $\beta(1+\epsilon)^2$-approximation algorithm.*

Proof. We first observe that S is a feasible solution for the BWCTP, since it schedules the jobset $\bigcup_i J_i = J$ on $\sum_i |M_i| = m$ machines. Furthermore the jobs on the machines are ordered according to $>_F$, since S is build from feasible subsolutions S_i^ϵ of instances of the BWCTP with equivalent follower's order. In addition $\mathcal{A}^\beta$ runs in polynomial time, since it mainly executes once the PTAS and less than m times the FPTAS. For the proof of the approximation ratio, we need to introduce the following notations:

- S^* is the optimal BWCTP solution of the instance.
- $\bar{S}^*$ is the optimal solution for the corresponding WCTP with $w := w^1$.
- S_i^* is the optimal BWCTP solution for the jobset J_i on $|M_i|$ machines.

- $\bar{S}_i^*$ is the optimal solution of the corresponding WCTP with $w := w^1$ for the jobset J_i on $|M_i|$ machines.

Then the following holds:

$$\begin{aligned} c(S) &= \sum_{M_i} c(S_i^\epsilon) \overset{\text{FPTAS}}{\leq} \sum_{M_i} (1+\epsilon) c(S_i^*) \\ &\overset{(2.10)}{=} (1+\epsilon) \sum_{M_i} \beta \cdot c(\bar{S}_i^*) \overset{(*)}{\leq} (1+\epsilon)\beta \sum_{M_i} c(\bar{S}_i^\epsilon) \\ &= (1+\epsilon)\beta \cdot c(\bar{S}^\epsilon) \overset{\text{PTAS}}{\leq} (1+\epsilon)^2 \beta \cdot c(\bar{S}^*) \\ &\leq (1+\epsilon)^2 \beta \cdot c(S^*) \end{aligned}$$

The inequality $(*)$ is valid, since $\bar{S}_i^*$ is the optimal WCTP schedule for jobset J_i on $|M_i|$ machines and $\bar{S}_i^\epsilon$ is a feasible schedule for this problem. Therefore $c(\bar{S}_i^*) \leq c(\bar{S}_i^\epsilon)$ holds, which finishes the proof. □

From the theorem follows that, if β could be proved to have have a constant upper bound on $m = 2$ and $m = 3$ machines, there would be a constant factor approximation algorithm for the BWCTP, since $|M_i| \in \{2, 3\}$. However, we can show that there is no constant upper bound on β for an arbitrary number m of machines, even in the case of the BWCTP with unit length processing times $p \equiv 1$. The proof of this statement will made by a counter example and needs some preliminary work.

The counter example will have processing times $p \equiv 1$ and jobs, whose indices are sorted according to $>_F$. In addition the *reversed* follower's order $\bar{>}_F$ will respect the "Smith's Rule" order of the leader $>_L$. For this scenario we proved in Corollary 2.3 the existence of an optimal solution for the BWCTP having the following structure:

For the jobset J_i of jobs being scheduled on machine i, it holds

$J_i = \{\pi_{i-1}+1, \pi_{i-1}+2, \ldots, \pi_i\}$ with $\pi_0 := 0$ and $1 \leq \pi_1 \leq \pi_2 \leq \cdots \leq \pi_m = n$.

A solution of this kind of structure will be stated in the following as *consecutive* solution. For an optimal consecutive solution the following can be observed:

Lemma 2.23. *Suppose S is an optimal solution of an instance of the BWCTP with $p \equiv 1$. Furthermore we assume S to be consecutive. Let $\mathcal{J}_h = \{j_1, \ldots, j_h\}$ be a set of h jobs on a machine $M \neq m$ in S, which are scheduled to the last h positions on machine M. If we denote by J_i the set of jobs on some machine i and by ρ_j the position of some job j in S, then the following statement holds:*

$$\sum_{j_i \in \mathcal{J}_h} \rho_{j_i} w_{j_i}^1 \leq \sum_{j_i \in \mathcal{J}_h} i \cdot w_{j_i}^1 + h \sum_{j \in J_{M+1}} w_j^1$$

Proof. Assume a schedule S_h, which is built from S by assigning the jobs $\mathcal{J}_h$ to machine $M+1$ instead of machine M. Note that, since S was consecutive, the jobs $\mathcal{J}_h$ will be scheduled on the first h positions of machine $M+1$ in S_h. Since the jobs $\mathcal{J}_h$ were scheduled on the last positions of machine M_j in S the contribution to the objective function of the jobs on machine M would decrease in S_h by a value:

$$\sum_{j_i \in \mathcal{J}_h} w^1_{j_i} C_{j_i} \overset{p\equiv 1}{=} \sum_{j_i \in \mathcal{J}_h} \rho_{j_i} w^1_{j_i}$$

Thus the decrease is equal to the left hand side of the stated inequality. On the other hand, the contribution to the objective function of the jobs on machine $M+1$ would increase in S_h by a value equal to the right hand side of the stated inequality. The first summand of the right hand side corresponds to the additional sum of weighted completion times for the jobs in $\mathcal{J}_h$, while the second summand describes the shift of the remaining jobs on machine $M+1$ by h positions. Therefore the inequality follows from the optimality of schedule S compared to the newly generated schedule S_h. □

As a second result we need the following statement on the structure of an optimal solution of the one-level WCTP, given unit length processing times:

Lemma 2.24. *Given an instance of the WCTP with processing times $p \equiv 1$ and let the jobs be indexed according to the reversed "Smith's Rule", that is, for two jobs j and k it holds that $j < k \Rightarrow w_j \leq w_k$. Then there is an optimal solution such that job j is scheduled on position $\bar{\rho}_j := \left\lceil \frac{n+1-j}{m} \right\rceil$ on some machine. In other words, the jobs with big indices are scheduled on small positions and vice versa.*

Proof. Note that for $p \equiv 1$ the completion time of a job does only depend on its position and not on the machine it is assigned to. In addition observe that, for each position ρ, the number of jobs j with $\bar{\rho}_j = \rho$ is equal to the number of possible jobs on that position, namely m. This does not necessarily hold for position $\bar{\rho}_1 = \left\lceil \frac{n+1-1}{m} \right\rceil = \left\lceil \frac{n}{m} \right\rceil$, where the number of jobs j with $\bar{\rho}_j = \bar{\rho}_1$ might be smaller.

Let S^* be an optimal solution of the WCTP. Then the lemma is proved by induction on the positions $\rho \leq \left\lceil \frac{n}{m} \right\rceil$ of the jobs in S^*. Suppose as an induction hypothesis that the statement holds for all jobs scheduled on positions $\rho' < \rho$ and that there are m jobs on each such position ρ'. These two statements trivially hold for position $\rho = 1$, since there are no smaller positions ρ'.

Now suppose for the induction step, that there is a job j on position ρ with $\bar{\rho}_j \neq \rho$. Because of the above observation and the induction hypothesis we can follow, that $\bar{\rho}_j > \rho$, since all jobs with $\bar{\rho}_j < \rho$ are scheduled on positions smaller than ρ. Furthermore there is some job k with $\bar{\rho}_k = \rho$ on position $\rho_k > \rho$, which again follows from the observed fact above. Now switch the positions of jobs j and k. From $\bar{\rho}_k = \rho < \bar{\rho}_j$ follows that $k > j$ and therefore $w_k \geq w_j$. Hence the optimal solution value of S^* is not increased by the switch. This switching procedure can be iterated until no job j with $\bar{\rho}_j > \rho_j = \rho$ is left.

If afterwards there is some empty slot on position ρ, there are either some jobs k with $\rho_k > \bar{\rho}_k = \rho$, which can be scheduled on position ρ or there are no jobs on bigger positions. In the last case $\rho = \bar{\rho}_1$. Therefore either there are no more positions to iterate and all jobs are set to positions $\leq \rho$, or the induction hypothesis holds for ρ. In both cases the statement of this lemma holds for all jobs on position ρ. $\square$

Now it is possible to prove the following theorem.

Theorem 2.25. *There cannot be a constant upper bound on the value β for instances of the BWCTP with $m = 2$ machines.*

Proof. The proof is made by giving a corresponding counter example instance.

Example 2.26. Given 2 machines as well as n jobs with $p \equiv 1$ and the following leader's and follower's weights:

$$w_i^1 = 10^i \quad \forall i = 1, \dots, n$$
$$w_i^2 = n + 1 - i \quad \forall i = 1, \dots, n$$

Therefore the jobs are indexed according to $>_F$ and the reversed follower's order $\bar{>}_F$ respects the leader's order $>_L$. For this case, we can assume the existence of an optimal consecutive solution S_n^* for the BWCTP due to Corollary 2.3. Thus the jobs 1 to j are assigned to machine 1 and the jobs $j + 1$ to n are assigned to machine 2 for some job $1 \leq j \leq n$.

From Lemma 2.24 follows, that an optimal solution $\bar{S}_n^*$ of the WCTP with weights $w := w^1$ can be stated as the following illustration shows, assuming n to be even:

Pos.	1	2	…	$\frac{n}{2}$
	n	$n-2$	…	2
	$n-1$	$n-3$	…	1

The objective function value of $\bar{S}_n^*$ is smaller, than the objective function value of a schedule S, which processes all jobs on one machine in decreasing index order. This holds for even values of n as well as for odd values. Hence we can conclude:

$$\begin{aligned} c(\bar{S}_n^*) = c(S) &< 10^n + 2 \cdot 10^{n-1} + \cdots + n \cdot 10 \\ &< 10^n \cdot (1.\bar{1}) + 10^{n-1} \cdot (1.\bar{1}) + \cdots + 10 \cdot (1.\bar{1}) \\ &< 10^n (1.\bar{1})^2 = 10^n \frac{10}{9}^2 = w_n^1 \frac{10}{9}^2 \end{aligned}$$

According to this argument the optimal objective value of the WCTP with $w := w^1$ can be bounded from above by the leader's weight w_n^1 of job n multiplied with some constant factor. Let ρ_n be the position of job n in the optimal consecutive schedule S_n^* of the BWCTP. Then it suffices to show the following equation for disproving the existence of a constant upper bound on β for all values of n.

$$\lim_{n \to \infty} \rho_n = \infty \tag{2.11}$$

Suppose there is a constant upper bound of B on the value β. Then it would hold that:

$$c(S_n^*) \overset{(2.10)}{=} \beta c(\bar{S}_n^*) \leq B c(\bar{S}_n^*) \tag{2.12}$$

From equation (2.11) we can deduce, that we find for each value of B a value for n, such that in the following line of inequalities the one marked by a $(*)$ is true:

$$c(S_n^*) \geq \rho_n w_n^1 \overset{(*)}{\geq} B \frac{10}{9}^2 w_n^1 > B c(\bar{S}_n^*)$$

This is a contradiction to inequality (2.12), which would prove the theorem.

To show that the equation (2.11) is correct, we assume the existence of a smallest possible upper bound $\hat{\rho}$ on the positions ρ_n of job n in the optimal consecutive schedules S_n^*. Since $\hat{\rho}$ is smallest possible, there is a value $\hat{n} = n$, such that n is scheduled on position $\hat{\rho}$ in $S_{\hat{n}}^*$. Let $n - k$ be the last job on the first machine in this setting. Therefore we have k jobs on machine 2. Now we consider $n = \hat{n} + 1$. In the optimal consecutive schedule $S_{\hat{n}+1}^*$ there are at most k jobs on machine 2, since $\hat{\rho}$ is an upper bound on the value of ρ_n. Suppose there are $k - h$ jobs on machine 2, thus the last h jobs on the first machines of $S_{\hat{n}+1}^*$ have been scheduled to machine 2 in $S_{\hat{n}}^*$. Since $S_{\hat{n}+1}^*$ is optimal, we can follow from Lemma 2.23 for the set $\mathcal{J}_h$ of the last h jobs on machine $M = 1$:

$$\sum_{j_i \in \mathcal{J}_h} \rho_{j_i} w_{j_i}^1 \leq \sum_{j_i \in \mathcal{J}_h} i \cdot w_{j_i}^1 + h \sum_{j \in J_2} w_j^1$$

Let $S_{\hat{n}}^h$ be the schedule constructed from $S_{\hat{n}}^*$ by assigning the jobs in $\mathcal{J}_h$ on machine 1 for $n = \hat{n}$. Then $S^h_{\hat{n}}$ schedules $k - h$ jobs on machine 2 equivalent to schedule $S^*_{\hat{n}+1}$. Because of the additional job on machine 1 it holds, that the positions ρ_{j_i} in $S^*_{\hat{n}+1}$ are one value bigger than the corresponding positions in $S^h_{\hat{n}}$. Therefore the above inequality would hold in the strict sense, if we compare the solutions $S^h_{\hat{n}}$ and $S^*_{\hat{n}}$. Hence we can follow $c(S^h_{\hat{n}}) < S^*_{\hat{n}}$, which is a contradiction to the optimality of $S^*_{\hat{n}}$.

Thus $S^*_{\hat{n}+1}$ schedules exactly k jobs on machine 2. By iterating this argument, we can show that there are exactly k jobs on machine 2 in S^*_n for all $n > \hat{n}$.

Now consider Lemma 2.23 for the set $\mathcal{J}_h = \{n-k\}$ containing the last job on machine 1 in any S^*_n for all $n > \hat{n}$. Then it follows:

$$\begin{aligned}(\rho_{n-k})w^1_{n-k} &\leq w^1_{n-k} + \sum_{j \in J_2} w^1_j = w^1_{n-k} + \sum_{j \in \{n-k+1,\dots,n\}} w^1_j \\ \Leftrightarrow n - k - 1 &\leq \frac{\sum_{j \in \{n-k+1,\dots,n\}} w^1_j}{w^1_{n-k}}\end{aligned}$$

If we increase n, then the left hand side of the last inequality increases, too. On the other hand, the right hand side stays constant, since for each increase of n by 1 the numerator and the denominator of the fraction increase by a factor of 10. Hence the whole fractional value stays constant. Therefore the stated inequality becomes invalid for some sufficient large n. This indicates, that the scheduling job $n - k$ to machine 2 would decrease the objective function value. Because of the above argumentation, the optimal consecutive solution would schedule job $n - k$ also to machine 2. Thus job n would have an position $\rho_n > \hat{\rho}$, which contradicts the assumption of an upper bound $\hat{\rho}$, proves equality (2.11) and finishes the proof of the theorem.

□

The above given theorem can be extended to every possible number of machines m.

Theorem 2.27. *There cannot be a constant upper bound on the value β for any instance of the BWCTP.*

Proof. Suppose an instance with jobs defined as in the above Example 2.26. Now we assume that there is a maximal position for job n on machine m in the optimal consecutive solutions of the BWCTP for each n. Therefore there is a job $n - k_{m-1}$ with a fixed k_{m-1} that cannot be scheduled on machine m, or the maximal position for n would be exceeded. Hence, job j_{m-1} has a maximum position on machine $m - 1$, or it would be scheduled to machine m according to the above argument based on Lemma 2.23.

This argumentation can be continued until reaching the first machine and some job $n - k_1$ with a fixed k_1 that has to be scheduled on machine 1 and whose position has to be smaller than some ρ. Since with increasing n the position of job $n-k_1$ also increases, if it stays on machine 1, the assumption on some maximal position for job n on machine m is false. □

Max ratio in w^1. The proof given above, which shows that β cannot be bounded from above by a constant value even in the case of unit length processing times, is based on the existence of very large differences in the weights of the leader. By restricting these differences, it might be possible to bound β to a small value as well. Before this idea is examined, the upper bound value on the differences in the leader's weights is defined as:

$$r_1 := \max_{j,k\in J}\left\{\frac{w_j^1}{w_k^1}\right\}$$

For the general BWCTP it is not possible to bound β from above by a constant value, even if r_1 is assumed to be equal to 1. To show this we could transform the given counter example of the proof of Theorem 2.27, for which $p\equiv 1$ holds, into a corresponding counter example with $w^1\equiv 1$ by using Theorem 2.2. Then r_1 would be equal to 1 and the argument of the proof could be applied to this case as well, since the structure of an optimal bilevel solution stays the same due to Subsection 1.2.2.

However, if we restrict the processing times to be equal to 1, it is easy to see, that a constant ratio r_1 leads to a constant upper bound value for β, by using the following observation:

Observation 2.28. *Given some jobs j and k in J with leader weights w_j^1 and w_k^1. Then it holds that:*

$$w_j^1 = w_k^1\cdot\frac{w_j^1}{w_k^1}\le w_k^1\cdot r_1$$

From this the following theorem can be deduced.

Theorem 2.29. *Given an instance I of the BWCTP with $p\equiv 1$ and r_1 as defined above. Then $\beta\le r_1$.*

Proof. Let S^* be the optimal BWCTP solution and $\bar{S}^*$ be the optimal solution for the corresponding WCTP with $w:=w^1$ on instance I. Additionally let S_R be a solution for the BWCTP resulting from reordering the jobs of $\bar{S}^*$ according to $>_F$ on each machine. Furthermore we denote by $j_{i,\rho}(S)$ the job, which is scheduled on machine i on position ρ in some schedule S. Then the following holds:

$$\beta c(\bar{S}^*) \overset{(2.10)}{=} c(S^*)\le c(S_R)=\sum_{i\in\mathcal{M}}\sum_{\rho}\rho\cdot w^1_{j_{i,\rho}(S_R)} \overset{\text{Obs. 2.28}}{\le}=\sum_{i\in\mathcal{M}}\sum_{\rho}\rho\cdot r_1 w^1_{j_{i,\rho}(\bar{S}^*)}=r_1 c(\bar{S}^*)$$

□

This result can be extended to the following one, by using a slightly more complex approach.

Theorem 2.30. *Given an instance I of the BWCTP with $p \equiv 1$ and r_1 as defined above. Then $\beta \leq m \sqrt[m]{r_1}$.*

Proof. Let $\underline{w}$ be the minimal leader's weight and $\bar{w}$ be the maximal leader's weight, that is, $\underline{w} := \min_{j \in J}\{w_j^1\}$ and $\bar{w} := \max_{j \in J}\{w_j^1\}$. Note that in this case holds $r_1 = \frac{\bar{w}}{\underline{w}}$. Now we partition the jobset J into subsets $J_1, \dots, J_m$ with:

$$J_i := \begin{cases} \{j \in J : \underline{w} \sqrt[m]{r_1^{i-1}} \leq w_j^1 < \underline{w} \sqrt[m]{r_1^i}\} & i \in \{1, \dots, m-1\} \\ \{j \in J : \underline{w} \sqrt[m]{r_1^{m-1}} \leq w_j^1 \leq \underline{w} \sqrt[m]{r_1^m} = \underline{w} r_1 = \bar{w}\} & i = m \end{cases}$$

Furthermore we define $r_1(i) := \max_{j,k \in J_i}\{\frac{w_j^1}{w_k^1}\}$. From the definition of the J_i it follows that $r_1(i) \leq \sqrt[m]{r_1}$. Therefore Observation 2.28 holds for $r_1(i)$, if it is applied to the jobset J_i.

Now let S^* be the optimal solution of the BWCTP on $p \equiv 1$ and let $\bar{S}^*$ be the corresponding optimal WCTP solution with $w := w^1$. In addition let S_C be that solution to the BWCTP resulting from scheduling each cluster of jobs J_i on one machine, respecting the order $>_F$ given by the follower. Furthermore we denote by $c_{J'}$ the contribution to the objective function value of a subset $J' \subseteq J$ of jobs. Thus for some schedule S with completion times C holds:

$$c_{J'}(S) = \sum_{j \in J'} w_j^1 C_j$$

Then the following statement is claimed to be true:

$$c_{J_i}(S_C) \leq m \sqrt[m]{r_1} c_{J_i}(\bar{S}^*) \tag{2.13}$$

With the claim, we could prove the theorem by concluding:

$$\beta c(\bar{S}^*) \overset{(2.10)}{=} c(S^*) \leq c(S_C) = \sum_{i=1}^{m} c_{J_i}(S_C) \overset{(2.13)}{\leq} \sum_{i=1}^{m} m \sqrt[m]{r_1} c_{J_i}(\bar{S}^*) = m \sqrt[m]{r_1} c(\bar{S}^*)$$

Hence just inequality (2.13) remains unproved. To prove it, we compare the jobset $\bar{\mathcal{J}}_1$, which consists of that m jobs of J_i having the smallest positions in $\bar{S}^*$ and the jobset $\mathcal{J}_1$, which consists of that m jobs of J_i having the smallest positions in S_C. Since in S_C all jobs of J_i are scheduled on one machine, the jobs in $\mathcal{J}_1$ have positions 1 to m. Additionally all jobs in $\bar{\mathcal{J}}_1$ have positions at least 1. Denoting by $\rho_j(S)$ the position of job j in schedule S, it holds:

$$c_{\mathcal{J}_1}(S_C) = \sum_{j \in \mathcal{J}_1} \rho_j(S_C) \cdot w_j^1 \overset{\text{Obs. 2.28}}{\leq} \sum_{k \in \bar{\mathcal{J}}_1} m \cdot \rho_k(\bar{S}^*) \cdot r_1(i) \cdot w_k^1 \leq m \sqrt[m]{r_1} \cdot c_{\bar{\mathcal{J}}_1}(\bar{S}^*)$$

For the central inequality note that for each job j and k as given it holds, that $\rho_j(S_C) \leq m \cdot \rho_k(\bar{S}^*)$. Therefore inequality (2.13) holds for these parts of jobs in J_i.

Now let the jobset $\bar{\mathcal{J}}_2$ consist of that m jobs of J_i having the next smallest positions in $\bar{S}^*$ apart from jobs in $\bar{\mathcal{J}}_1$ and let the corresponding jobset $\mathcal{J}_2$ consist of that m jobs of J_i having the next smallest positions in S_C apart from jobs in $\mathcal{J}_1$. The jobs in $\mathcal{J}_2$ have positions from $m+1$ to $2m$ and the jobs in $\bar{\mathcal{J}}_2$ have at least position 2, since otherwise there would be more than m jobs with positions smaller than 2 in $\bar{S}^*$, which is not possible. Therefore the same argument as for $\mathcal{J}_1$ and $\bar{\mathcal{J}}_1$ can be applied to show that inequality (2.13) also holds for the jobsets $\mathcal{J}_2$ and $\bar{\mathcal{J}}_2$.

This procedure can be iteratively continued for each h and the corresponding jobsets $\mathcal{J}_h$ and $\bar{\mathcal{J}}_h$ until all jobs in J_i are covered. Then the following final statement can be concluded, which proves inequality (2.13) and therefore the theorem:

$$c_{J_i}(S_C) = \sum_{\mathcal{J}_h} c_{\mathcal{J}_h}(S_C) \leq m \sqrt[m]{r_1} \sum_{\bar{\mathcal{J}}_h} c_{\bar{\mathcal{J}}_h}(\bar{S}^*) = m \sqrt[m]{r_1} \cdot c_{J_i}(\bar{S}^*)$$

□

Thus for the case of $p \equiv 1$ there is an upper bound of $m \sqrt[m]{r_1}$ for the value of β, where m denotes the number of machines. Given algorithm $\mathcal{A}^\beta$, in which the value of β is used for the BWCTP on 2 and 3 machines, it follows that:

Corollary 2.31. *Algorithm $\mathcal{A}^\beta$ is an $\beta(1+\epsilon)^2$-approximation algorithm for the BWCTP with $p \equiv 1$, where $\beta \leq \max\{2 \cdot \sqrt{r_1}, 3 \cdot \sqrt[3]{r_1}\}$ and $r_1 := \max_{j,k \in J} \left\{ \frac{w_j^1}{w_k^1} \right\}$.*

Note that it is possible to choose other sizes of the M_i in $\mathcal{A}^\beta$ and therefore change the value of m in the upper bound on β.

Partition that minimizes w^1 difference. In the proof of Theorem 2.30, a partitioning of the jobset J into parts J_i for $i \in \{1, \dots, m\}$ is used, that guarantees a maximal ratio $r_1(i)$ of $\sqrt[m]{r_1}$. However, there might be a better partition of the jobs to sets J_i, such that $r := \max_i\{r_1(i)\}$ is minimized for a given instance of the BWCTP. In this last part of the section an approach is presented, that finds a partition of the jobset J into subsets J_i that minimizes r, given a jobset with leader weights w^1 and a maximal partition size m.

Denote first, that there is an instance of the BWCTP with $p \equiv 1$, such that $\sqrt[m]{r_1}$ is the smallest possible value for r. Suppose there is a job j_k with $w^1_{j_k} = \underline{w} \cdot \sqrt[m]{(r_1)^k}$ for $k \in \{0, \ldots, m\}$. Hence there are $m+1$ jobs, whose pairwise compared leader weight values differ by at least a factor of $\sqrt[m]{r_1}$. Since there are only m different jobsets J_i in the partition, at least two of them have to be in the same jobset i. Therefore $\sqrt[m]{r_1} \leq r_1(i) \leq r$ holds. Thus the ratio for r in Theorem 2.30 is best possible.

Nevertheless, a smaller r might exist for a specific instance. Therefore imagine a complete graph G, in which each job of J is represented by a node in G. Each edge $e = \{j, k\}$ in the graph has an edge weight of $w_{\{j,k\}} = \frac{w^1_j}{w^1_k}$ if and only if $w^1_j \geq w^1_k$. Hence $w_{\{j,k\}} \geq 1$ holds. The task of finding a partition of J into m jobsets J_i with minimal r can now be translated into a node partitioning problem on G into m parts, such that the maximal weight of an edge connecting two nodes in the same part is minimized. Speaking in terms of colors, we are looking for a coloring of the nodes of G with m colors, such that the weight of a monochromatic edge, that is, an edge whose end nodes have the same color, is minimized.

Suppose we want to achieve a minimal monochromatic edge weight of value R with $1 \leq r \leq R$. Then the coloring of each edge having a weight smaller or equal to R is irrelevant. Therefore the edge could also be removed from G resulting in a non-complete graph $G_R = (V, E)$. For the rest of the edges $e \in E$ it holds, that $w_e > R$ and therefore their end nodes have to be in different color classes. This is the classic node coloring problem on G_R. Hence it is possible to achieve a minimal monochromatic edge weight of R and therefore find a partition of J into m different jobsets J_i such that $r \leq R$, if G_R can be node colored with at most m different colors.

In general, node coloring on an arbitrary graph is $\mathcal{NP}$-hard [Kar1972]. Therefore the result from above does not seem to provide an algorithm for finding a minimal R and hence a minimal r. However, G_R has a special structure, which makes it easy to find a proper coloring of the nodes.

To show this, we give each edge $\{i, j\}$ in the edge set E of G_R an orientation $i \to j$ if and only if $w^1_i < w^1_j$ and denote by F the set of oriented edges in G_R. An edge $\{i, j\}$ with orientation $i \to j$ will be referred to as (i, j) in the following. This orientation is transitive, since if $(i, j) \in F$ and $(j, k) \in F$ then follows that

$$\begin{aligned} w^1_i \leq w^1_j \leq w^1_k \quad &\text{and} \\ \frac{w^1_j}{w^1_i} > R \quad &\text{holds,} \\ \text{and therefore also } \frac{w^1_k}{w^1_i} > R\ . \end{aligned}$$

Therefore $\{i, k\}$ is in E and (i, k) is in F.

Since there is a transitive orientation of G_R, the graph is a comparability graph, compare the definition of Golumbic [Gol2004]. In the same book it is shown that comparability graphs are perfect graphs. Thus the minimal number of colors needed to color G_R is equal to the maximum size of a clique in G_R. Furthermore the height function of G_R defines a minimal coloring of the nodes in G_R, which can be seen from section 7 of chapter 5 in [Gol2004]. The height function on a transitive orientation can be defined for a node v in G_R as:

$$h(v) := \begin{cases} 0 & v \text{ is a source, that is, } \nexists (w, v) \in F \\ 1 + \max\{h(w) | (w, v) \in F\} & \text{else} \end{cases}$$

From this definition it follows, that in a height function with maximal height of $\bar{h}$ each value $h \in \{0, \dots, \bar{h}\}$ is assigned to some node. Therefore the number of different height values used is equal to $\bar{h} + 1$.

Since a height function can be recognized in linear time for a transitive orientation, we can determine the minimal number of colors needed to color the nodes of G_R given a value of R. However, the goal of this section was to give an algorithm which finds the minimal R for a given number m of maximal colors to be used. To achieve this, the following algorithm $\mathcal{A}_h$ is presented in Algorithm 3, which results in a step function $h_j : [1, \infty) \to \mathbb{N} \cup \{0\}$ for each job node j, which describes the height $h_j(R)$ of node j in G_R. The algorithm processes the job nodes in ascending leader's weights order. First

Algorithm 3: Algorithm $\mathcal{A}_h$

```
1  $L_w \leftarrow$ List of job nodes sorted ascending by $w^1$;
2  $\underline{w} \leftarrow$ minimal weight $w^1$;
3  foreach job $j \in L_w$ do
4      if $w^1_j = \underline{w}$ then
5          $h_j \leftarrow 0 \quad \forall R$;
6      else
7          $L_s \leftarrow \emptyset$;
8          foreach job node $i <_{L_w} j$ do
9              Define step function $f_{(i,j)}(R) := 0$ for $R \geq w_{(i,j)}$ and
               $f_{(i,j)}(R) := 1 + h_i(R)$ for $R < w_{(i,j)}$;
10             $L_s \leftarrow L_s \cup \{f_{(i,j)}\}$
11         $h_j(R) \leftarrow \max_{f_{(i,j)} \in L_s}\{f_{(i,j)}(R)\} \quad \forall R$;
```

the height of those jobs having the smallest leader's weight is set to zero for all values of R. For each job j with non-minimal weight of the leader the algorithms starts a second iteration over all jobs i with smaller leader weight. Here a step function $f_{(i,j)}$ is build, that is zero for $R \geq w_{(i,j)}$ and has the value of $1+h_i(R)$ for $R < w_{(i,j)}$. The step function $h_j(R)$ for job node j is finally defined as the maximum function over all step functions $f_{(i,j)}$.

For this algorithm $\mathcal{A}_h$ we can prove the following theorem.

Theorem 2.32. *Algorithm $\mathcal{A}_h$ determines the correct height $h_j(R)$ in the graph G_R for each job node j and each value of R in polynomial time.*

Proof. Note first that the height of jobs j with $w_j^1 = \underline{w}$ is correctly set to zero, since there is no ingoing edge for node j in any G_R. From this the correctness of h_j for all other job nodes j can be proved by induction of jobs in the list L_w. For each job j all edges (i, j) to preceding jobs i in L_w are considered, that is, all possible ingoing edges of job j. The step function $f_{(i,j)}$ is build for each such edge (i, j) and it holds that the height of job j has to be greater or equal than $f_{(i,j)}(R)$ in each G_R. This is true since $f_{(i,j)}$ is zero if (i, j) is not in G_R, that is, if $w_{(i,j)} \leq R$. If $w_{(i,j)} > R$, then the edge (i, j) is in the graph and the step function $f_{(i,j)}$ is set to $1 + h_i(R)$, where $h_i(R)$ is the correct height value for job i due to induction. Since in line 11, $h_j(R)$ is set to the maximum function over all step functions $f_{(i,j)}$ in the list of step functions L_s for preceding nodes i, the height is correctly chosen for j.

From this it follows, that each considered step function f has at most n steps, since the height of a node in some comparability graph can have at most n different values. Hence the runtime of line 11 is $O(n^2)$, because here the maximum function over at most n step functions with at most n steps is built. Since the line is executed at most n times and additionally the sorting of the list L_w can be done in $O(n \cdot \log n)$ by using merge sort, the overall runtime of algorithm $\mathcal{A}_h$ lies in $O(n^3)$. □

Corollary 2.33. *It is possible to determine a partition of the jobs J that minimizes the value of r in polynomial time.*

Proof. We first use algorithm $\mathcal{A}_h$ to determine the height of each job node j for all possible values R in polynomial time. To find that height function, which minimizes R, and therefore also r, while using at most m different height values, look at the job $\bar{j}$ with the maximal leader's weight w^1. It holds that if an edge (i, j) is in G_R, then also the edge $(i, \bar{j})$ is in G_R, since the following is true:

$$R < \frac{w_j^1}{w_i^1} \leq \frac{w_{\bar{j}}^1}{w_i^1}$$

Suppose node k is the one of G_R with the maximal height value $\bar{h}$, then there is some node i, such that edge (i, k) is in F and the height of i is $h - 1$. Then also edge $(i, \bar{j})$

exists in G_R and therefore $\bar{j}$ has the maximal height in each G_R.

Therefore the minimal possible R, for which the height has not more than m different values, is that minimal R, for which $\bar{j}$ has a height $h_{\bar{j}}(R) \leq m-1$. The partition of the jobset J into parts J_i is then done by assigning those jobs to J_i having height $i-1$ for the chosen value of R. $\square$

3. Approaches based on integer and quadratic programming

In the following chapter we will study two different models of the BWCTP: An integer programming model for the special case of the BWCTP having unit weight processing times and a quadratic integer programming model for the general case. Both formulations describe the BWCTP respectively the mentioned special case exactly and can therefore be used to find the corresponding optimal solution. However, it is well known that solving integer and quadratic integer programming models is $\mathcal{NP}$-hard in general. Therefore we will also consider the non-integer relaxations of the models and construct solutions for the corresponding case of the BWCTP from the optimal solutions of them. Note that both formulations will also be part of the experimental evaluation in Chapter 4 of this thesis.

3.1. An integer programming model for the BWCTP with $p \equiv 1$

The following section of the thesis will examine the possibility to solve the BWCTP with $p \equiv 1$ by using the IP formulation $(\mathrm{IP})_{\mathrm{p}}$. It was defined in Corollary 2.3 in Section 2.1 and is based on the IP $(\mathrm{IP})_{\mathrm{KK}}$ formulated by Kis and Kovács [KisKov2012] for the special case of $w^1 \equiv 1$. In Theorem 2.2 and Corollary 2.3 it was shown, that the special cases for the BWCTP are equivalent and that the results for $(\mathrm{IP})_{\mathrm{p}}$ and $(\mathrm{IP})_{\mathrm{KK}}$ coincide. Note that since $(\mathrm{IP})_{\mathrm{p}}$ is formulated for the special case of $p \equiv 1$ of the BWCTP, all results of this section are also only valid for this special case respectively the corresponding equivalent case of $w^1 \equiv 1$.

For the $p \equiv 1$ case the jobs are assumed to be indexed according to $>_F$ and the positions are conventionally numbered starting with 1 on the first job. Note that the completion time of a job does only depend on its position and not on the machine it is assigned to. Therefore, the model uses variables $x_{\rho,j}$ to indicate that job j is scheduled on position ρ, without specifying the actual machine. We denote by the value $m_j = \min\{j, n - m + 1\}$ the largest possible position of job j in a solution of the BWCTP without idle machines, which coincides with Assumption 1.10. Furthermore note, that similar to the model $(\mathrm{IP})_{\mathrm{KK}}$ for the position ρ_j of job j in a feasible schedule $\rho_j \leq j$ holds. Hence, only variables satisfying this condition are considered. Then $(\mathrm{IP})_{\mathrm{p}}$ can be formulated as follows:

(IP)$_{\mathbf{p}}$

$$\text{Objective:} \qquad \min_x c(x) = \sum_{j=1}^{n}\sum_{\rho=1}^{m_j} w_j^1 \rho x_{j,\rho}$$

$$\sum_{\rho=1}^{m_j} x_{j,\rho} = 1 \quad \forall\, j = 1,\dots,n \tag{3.1}$$

$$\sum_{j=\rho}^{n} x_{j,\rho} \leq m \quad \forall\, \rho = 1,\dots,n-m+1 \tag{3.2}$$

$$\sum_{j=\rho}^{l} x_{j,\rho} \geq \sum_{j=\rho+1}^{l+1} x_{j,\rho+1} \quad \forall\, l = 1,\dots,n-m \text{ and } \rho \leq l \tag{3.3}$$

$$\text{position variables:} \qquad x_{j,\rho} \in \{0,1\} \quad \forall\, j = 1,\dots,n, \rho = 1,\dots,m_j \tag{3.4}$$

The last constraint (3.4) is replaced by $0 \leq x_{j,\rho} \quad \forall\, j = 1,\dots,n, \rho = 1,\dots,m_j$ for the LP relaxation $(\mathrm{LP})_{\mathrm{p}}$ of the model.

Since solving an IP with variables in $\{0,1\}$ is known to be $\mathcal{NP}$-complete in general [Kar1972], the IP formulation above cannot be used to find the optimal solution X^* for the special case of the BWCTP in polynomial time unless $\mathcal{P} = \mathcal{NP}$. Therefore the idea is to solve $(\mathrm{LP})_{\mathrm{p}}$, which can be done in polynomial time, and create an integer solution out of the possibly fractional optimal solution $\bar{X}^*$ of the LP. Since each feasible solution of the IP is also a feasible solution of the LP, it holds that $c(\bar{X}^*) \leq c(X^*)$. Hence the optimal solution value of $(\mathrm{LP})_{\mathrm{p}}$ is a lower bound for the optimal solution of $(\mathrm{IP})_{\mathrm{p}}$.

General model results. In the following part some general results regarding the model will be presented, to prepare the subsequent analysis. The first one concerns the fact, that $(\mathrm{IP})_{\mathrm{p}}$ is an exact description of the BWCTP with $p \equiv 1$. This result was already stated in the paper of Kis and Kovács [KisKov2012] for the corresponding $w^1 \equiv 1$ case, but has not been proved there.

Theorem 3.1. *For each feasible schedule S of the BWCTP with $p \equiv 1$ there is an feasible solution x_S of (IP)$_p$ having the same objective value and assigning jobs to the same positions. On the other hand, for each feasible solution x of (IP)$_p$, there is a feasible schedule S_x using the positions given by x and having the same objective value.*

Proof. The first statement of the theorem has already been shown in the description of $(\mathrm{IP})_{\mathrm{KK}}$ in Subsection 1.2.2 and can be adopted to the $p \equiv 1$ case by using Corollary 2.3.

For the second statement, assume a feasible solution x of $(\mathrm{IP})_{\mathrm{p}}$. Then the schedule S_x is created by the following procedure.
For each position ρ, sort the jobs j with $x_{j,\rho} = 1$ ascending with respect to their index. Then schedule that job with the i-th smallest index on position ρ of machine i to obtain S_x.

Because of the constraints (3.1) and (3.2) each job is scheduled on exactly one position in S_x and at most m machines are used. So the only thing that has to be proved now to show that S_x is feasible, is the correct order $>_F$ on each machine, which corresponds to the order of the job indices. Suppose job j is scheduled on machine i on some position $\bar{\rho}$. If $\bar{\rho} = 1$ there is no conflict to previous jobs, because there are non. In the case of $\bar{\rho} > 1$ it is claimed that there is a job k on position $\bar{\rho} - 1 = \rho$ on machine i having a smaller index than j, which would prove the correct order on the machines.

Consider constraint (3.3) for the given ρ and $l = j - 1$:

$$\sum_{h=\rho}^{j-1} x_{h,\rho} \geq \sum_{h=\bar{\rho}}^{j} x_{h,\bar{\rho}}$$

The right hand side of the constraint sums up the x-values for position $\bar{\rho}$ up to job j. Therefore this sum is equal to the value i, denoting the machine number of the machine job j is scheduled on. Hence there are at least i jobs on the previous position ρ with smaller indices than j, since the left hand side only sums up jobs on position ρ up to an index of $j - 1$. Let k be the i-th smallest index having an x-value of 1 on position ρ, then k is scheduled in front of j on machine i and it holds that $j > k$, which proves the correct order on the machine in S_x. In addition, the jobs are scheduled to the machines without any idle times, since there is always a job k directly preceding a scheduled job j as described.

To prove that the objective values of x and S_x coincide, we observe that each position of a job corresponds to its completion time. Since the positions for the jobs in S_x are those specified by the values of x, the objective function values are equivalent. □

A similar theorem as the one stated above can also be proved for the LP relaxation $(\mathrm{LP})_{\mathrm{p}}$. Therefore we firstly define a *fractional machine schedule*.

Definition 3.2 (Fractional machine schedule)**.** *Given an instance of the BWCTP with jobset J and m machines. Then a feasible* fractional machine schedule $\bar{S}$ *has the following characteristics:*

- *Each used machine M has a machine value v_M.*

- *An arbitrary number $\bar{m}$ of machines can be used, but the machine values have to sum up to at most m, such that it holds:*

$$\sum_{M=1}^{\bar{m}} v_M \leq m \tag{3.5}$$

- *Each job j has to be scheduled on a set $\mathcal{M}_j$ of machines, such that it holds:*

$$\sum_{M \in \mathcal{M}_j} v_M = 1 \tag{3.6}$$

- *On each machine M there cannot be two jobs of the same index, the follower's order has to be respected and the processing time for each job j is equal to p_j resulting in completion times $C_j(M)$ for job j.*

- *The objective function value of $\bar{S}$ is defined as:*

$$c(\bar{S}) = \sum_{j \in J} \sum_{M \in \mathcal{M}_j} (v_M \cdot w_j^1) C_j(M)$$

From the definition the following observation directly results:

Observation 3.3. *Given a feasible fractional machine schedule $\bar{S}$ for the BWCTP with $v_M \equiv 1$ for all machines M. Then $\bar{S}$ is a "normal" feasible schedule for the BWCTP.*

With this definition it is now possible to state the theorem for $(\mathrm{LP})_\mathrm{p}$ corresponding to Theorem 3.1:

Theorem 3.4. *For each feasible fractional machine schedule $\bar{S}$ of the BWCTP with $p \equiv 1$ there is a feasible solution $\bar{x}_{\bar{S}}$ of $(LP)_p$ having the same objective value and assigning jobs to the same positions. On the other hand, for each feasible solution $\bar{x}$ of the LP, there is a feasible fractional machine schedule $\bar{S}_{\bar{x}}$ using the positions given by $\bar{x}$ and having the same objective value.*

Proof. To prove the theorem, we partition the set of machines $\mathcal{M}_j$, on which a job j in a fractional machine schedule $\bar{S}$ is scheduled, according to the particular position $\rho_j(M)$ of j on machine $M \in \mathcal{M}_j$. Thus we define $\mathcal{M}_j^\rho := \{M \in \mathcal{M}_j \,|\, \rho_j(M) = \rho\}$. Then the following terms are identified:

$$\bar{x}_{j,\rho} = \sum_{M \in M_j^\rho} v_M \tag{3.7}$$

From this it is possible to deduce for some $\bar{x}$ defined as above:

$$\sum_{j=\rho}^{n} \bar{x}_{j,\rho} = \sum_{j=\rho}^{n} \sum_{M \in M_j^\rho} v_M \overset{(*)}{\leq} \sum_{M=1}^{\bar{m}} v_M$$
$$\sum_{\rho=1}^{m_j} \bar{x}_{j,\rho} = \sum_{\rho=1}^{m_j} \sum_{M \in M_j^\rho} v_M = \sum_{M \in \mathcal{M}_j} v_M$$

Here the inequality $(*)$ follows from the argument, that there can only be one job on position ρ of some machine M. Therefore the sum of machine values of all jobs being scheduled on some position ρ is smaller than the sum of machine values over all machines.

From the two inequalities it follows, that having a feasible fractional machine schedule $\bar{S}$ which satisfies the fractional schedule constraints (3.5) and (3.6), the LP solution $\bar{x}_{\bar{S}}$ defined by (3.7) satisfies the LP constraints (3.1) and (3.2). Furthermore it holds, that $\bar{x}_{\bar{S}} \geq 0$. For the last LP constraint (3.3) it is possible to argue like in the IP case in Section 1.2.2. Due to Assumption 1.10 it holds also in a fractional machine schedule, that each job j on a position $\rho+1$ on some machine is preceded by some job $k < j$ on position ρ of that machine. Therefore we can find for each contribution to the sum of x-values of the first $l+1$ jobs on position $\rho+1$ an equivalent contribution to the sum of x-values of the first l jobs on position ρ, which proves the inequality.

For the proof of the objective function values being equal, note that also for fractional machine schedules the completion time is equal to the position since $p \equiv 1$ holds. Then the following statement is true:

$$\begin{aligned} c(\bar{S}) &= \sum_{j \in J} \sum_{M \in \mathcal{M}_j} (v_M \cdot w_j^1) C_j(M) \\ &= \sum_{j=1}^{n} \sum_{\rho=1}^{m_j} \sum_{M \in \mathcal{M}_j^\rho} v_M \cdot w_j^1 \rho \\ &\overset{(3.7)}{=} \sum_{j=1}^{n} \sum_{\rho=1}^{m_j} \bar{x}_{j,\rho} \cdot w_j^1 \rho = c(\bar{x}_{\bar{S}}) \end{aligned} \tag{3.8}$$

For the second part of the theorem we assume the existence of some feasible solution $\bar{x}$ of $(\mathrm{LP})_{\mathrm{p}}$. Then, similar to the proof of Theorem 3.1, we define the fractional machine schedule $\bar{S}_{\bar{x}}$ as follows:
For each position ρ, sort the jobs j with $\bar{x}_{j,\rho} > 0$ ascending with respect to their index. Then schedule the jobs in that order on the machines, which are also ordered by index, by using the following rules for job j and the current machine M:

- If $\bar{x}_{j,\rho} = v_M$ schedule j on position ρ of M and continue with the next job and machine.

- If $\bar{x}_{j,\rho} > v_M$ schedule j on position ρ of M and continue with the next machine and the remaining x-value of $\bar{x}_{j,\rho} - v_M$ for job j and position ρ.

- If $\bar{x}_{j,\rho} < v_M$ then split machine M into machines M_1 and M_2, where M_1 gets the index of M and $M_2 = M_1 + 1$. In addition shift the following machine indices also by a value of 1. Copy all jobs of M to both machines M_1 and M_2 and set the machine values $v_{M_1} = \bar{x}_{j,\rho}$ and $v_{M_2} = v_M - \bar{x}_{j,\rho}$. Now schedule j on position ρ of M_1 and continue with the next job and machine M_2.

From the definition of $\bar{S}_{\bar{x}}$ it follows that the equation (3.7) again holds, since the value $\bar{x}_{j,\rho}$ is equal to the sum of machine values of the machines, scheduling job j on position ρ. Therefore the constraint (3.6) is also valid for $\bar{S}_{\bar{x}}$, because the LP constraint (3.1) is valid for $\bar{x}$, as shown above. Furthermore the model constraint (3.2) is satisfied by $\bar{x}$. For this reason, the sum of x-values on each position ρ is not greater than m. Hence the sum of the machine values on that position is also bounded from above by m. We can conclude, that the overall sum of machine values is not greater than m, since on each position the order of machines to assign the jobs on is the same, which proves, that the fractional constraint (3.5) is satisfied.

Now it has to be proved, that the follower's order $>_F$ is respected on each machine and that there is no machine with two jobs of the same index. In addition there must not be any idle time between two jobs on the machines. From this the feasibility of $\bar{S}_{\bar{x}}$ would follow. The argument for the requirements works again similar to the proof of Theorem 3.1.

Suppose job j is scheduled on machine i on some position $\bar{\rho}$. If $\bar{\rho} = 1$ there is no conflict to previous jobs, because there are non. In the case of $\bar{\rho} > 1$ we claim the existence of some job k on position $\bar{\rho} - 1 = \rho$ on machine i, which has a smaller index than j. This would prove all the three requirements stated above.

To prove this, consider again constraint (3.3) for the given ρ and $l = j - 1$ as in the previous theorem.

$$\sum_{h=\rho}^{j-1} \bar{x}_{h,\rho} \geq \sum_{h=\bar{\rho}}^{j} \bar{x}_{h,\bar{\rho}}$$

The right hand side of the constraint sums up the x-values for position $\bar{\rho}$ up to job j, which is at least as big as the sum of x-values of jobs on the previous position ρ with smaller indices than j. Since the jobs are assigned to machines according to their indices, on each machine i, that has job j scheduled on it at position $\bar{\rho}$, there is some job $k < j$ scheduled in front of j. This proves the feasibility of $\bar{S}_{\bar{x}}$.

To finally prove that the objective values of $\bar{x}$ and $\bar{S}_{\bar{x}}$ are equal, note that equation (3.8) is also valid for these two solutions, since it has already been proved that equation (3.7) holds again. □

As a last general result concerning $(\mathrm{IP})_\mathrm{p}$, the Theorem 1.11 presented by Kis and Kovács [KisKov2012], which has not been proved correctly, will now be analyzed. After the theorem is again stated to have the given conditions in mind, a correct proof is presented for $(\mathrm{LP})_\mathrm{p}$ and $p \equiv 1$, which can be applied to $(\mathrm{IP})_\mathrm{KK}$ and the $w^1 \equiv 1$ case by using Theorem 2.2.

Theorem (Theorem 1.11)**.** *If the follower's order $>_F$ is respecting the leader's order $>_L$, then (LP)$_p$ admits an optimal integer solution. If additionally $>_L$ is a total order then all optimal solutions are integer.*

Correct proof of Theorem 1.11. To prove the theorem, first the structure of an optimal solution of $(\mathrm{IP})_\mathrm{p}$ is determined. It has been noted in Section 1.2.2, that if the follower's order $>_F$ is respecting the leader's order $>_L$ the BWCTP is equivalent to the corresponding WCTP using weights $w := w^1$. Furthermore an optimal solution of the WCTP for the $p \equiv 1$ case was described in Lemma 2.24. It was stated that indexing the jobs according to the *reversed* “Smith's Rule” order leads to an optimal schedule having the jobs with biggest indices scheduled on the first machines. Therefore the given job indexing according to the non-reversed follower's order $>_F$, respecting $>_L$ and hence also the “Smith's Rule” order of the WCTP, yields an optimal solution scheduling the jobs 1 to m to the first positions of the m machines, the jobs $m+1$ to $2m$ to the second positions of the machines and so on. By Theorem 3.1 we find an optimal IP solution x^* with corresponding x-values of $x^*_{j,\rho} = 1$ if and only if $\rho = \lceil \frac{j}{m} \rceil$.

Now assume an optimal solution $\bar{x}^*$ of $(\mathrm{LP})_\mathrm{p}$ with $\bar{x}^* \neq x^*$ is given. Then the following Algorithm 4 is used to transform $\bar{x}^*$ into some other optimal solution $\hat{x}^*$ of the LP. In the algorithm we iterate over all positions from 1 to $n-m+1$. For each position ρ we are looking for jobs j' and positions $\rho' \neq \rho$ with $\rho' \neq \rho = \left\lceil \frac{j'}{m} \right\rceil$, for which positive values $x_{j',\rho'}$ exist. These fractional x-values of job j' on position ρ' are then shifted to position ρ by exchanging them with x-values of some job j on position ρ with $\rho \neq \lceil \frac{j}{m} \rceil$, if such a job j exists. If there is no such job j, then we only shift the x-value of job j' to position ρ. This procedure is continued, until no more jobs j' as described exist.

The Algorithm 4 returns a solution of $\hat{x}^* = x^*$, since for each position ρ the condition $x_{j,\rho} = 1$ if and only if $\rho = \lceil \frac{j}{m} \rceil$ is established. Hence the returned solution is feasible for $(\mathrm{LP})_\mathrm{p}$, since it is feasible for $(\mathrm{IP})_\mathrm{p}$.

Algorithm 4: Transformation from $\bar{x}^*$ to $\hat{x}^*$

```
1  $x \leftarrow \bar{x}^*$ ; // for easier notation
2  foreach $\rho \in \{1, \dots, n-m+1\}$ do
3      while $\exists\, x_{j',\rho'} > 0 : \left\lceil \frac{j'}{m} \right\rceil = \rho \neq \rho'$ do
4          if $\exists\, x_{j,\rho} > 0 : \lceil \frac{j}{m} \rceil \neq \rho$ then
5              $\epsilon \leftarrow \min\{x_{j,\rho}, x_{j',\rho'}\}$;
6              $x_{j,\rho} \leftarrow x_{j,\rho} - \epsilon$;
7              $x_{j',\rho'} \leftarrow x_{j',\rho'} - \epsilon$;
8              $x_{j,\rho'} \leftarrow x_{j,\rho'} + \epsilon$;
9              $x_{j',\rho} \leftarrow x_{j',\rho} + \epsilon$;
10         else
11             $x_{j',\rho} \leftarrow x_{j',\rho} + x_{j',\rho'}$;
12             $x_{j',\rho'} \leftarrow 0$;
13 return $\hat{x}^* \leftarrow x$;
```

The only thing left to prove is that $c(\hat{x}) \leq c(\bar{x})$. Note that the positions are processed from 1 to $n-m+1$. Thus, also the statement $x_{j,\rho} = 1$ if and only if $\rho = \lceil \frac{j}{m} \rceil$ is established in the order. Therefore it holds that $\rho' > \rho$, since all x-values for jobs on positions less then ρ' already are positioned correctly. From the same argument follows that for a job j as define by the if-clause it holds that $\lceil \frac{j}{m} \rceil > \rho$. This yields that $j' < j$, since $\left\lceil \frac{j'}{m} \right\rceil = \rho < \lceil \frac{j}{m} \rceil$. Furthermore for two jobs j and j' the following statement is valid because of the chosen indexing and the conditions of the theorem: $j' < j \Leftrightarrow w^1_{j'} \geq w^1_j$.

Hence in the case specified by the if-condition the objective function value of x changes by a term of

$$\epsilon(w^1_j \rho' + w^1_{j'} \rho - w^1_j \rho - w^1_{j'} \rho') = \epsilon \cdot \underbrace{(w^1_{j'} - w^1_j)}_{\geq 0} \cdot \underbrace{(\rho - \rho')}_{<0} \leq 0$$

For the other case the objective function value's change can be stated as

$$\epsilon(w^1_{j'} \rho - w^1_{j'} \rho') = \epsilon \cdot w^1_{j'} \cdot \underbrace{(\rho - \rho')}_{<0} < 0$$

Therefore the objective function value of x never increases, which states the needed result.

In the case of a total order $>_L$, it holds that $j < k \Leftrightarrow w^1_j > w^1_k$. Thus, the objective function value of x always decreases. Since this is not possible for some optimal solution $\bar{x}^*$ of $(\mathrm{LP})_\mathrm{p}$, it can be concluded that $\bar{x}^* = x^*$, which proves the second statement. □

Approximation using the LP. As already announced above, the following part presents some approaches that try to use the optimal solution of $(\mathrm{LP})_\mathrm{p}$ to generate a good solution for the BWCTP with $p \equiv 1$. The idea is to prove, that the obtained integer solution has an objective value, which can be bounded from above by a constant multiple α of the objective value of the optimal LP solution. Since the optimal LP solution's objective value is a lower bound for the optimal solution's objective value of $(\mathrm{IP})_\mathrm{p}$ and therefore also for the optimal solution value of the BWCTP with $p \equiv 1$, the algorithm would be an approximation algorithm with constant approximation ratio α.

In the first part of Section 2.4 it has been explained, that it is important for the lower bound to be near to the corresponding optimal value. For the case of $(\mathrm{LP})_\mathrm{p}$, the corresponding gap to the optimal integer solution of $(\mathrm{IP})_\mathrm{p}$ seems to be quite small, like the empirical analysis in Section 4.1 of Chapter 4 shows. Furthermore there are special cases, in which the IP and its LP relaxation yield the same optimal solution value. As shown above in the proved Theorem 1.11, $(\mathrm{LP})_\mathrm{p}$ admits an integer optimal solution if $>_F$ is respecting $>_L$. In addition the results of Chapter 4 show, that the gap between $(\mathrm{IP})_\mathrm{p}$ and the relaxation is zero in every instance having $m = 2$ machines. Therefore the following conjecture is made:

Conjecture 3.5. *In this case of only two machines, $(LP)_p$ admits an optimal integer solution.*

One possible idea for a proof of the conjecture would be the observation, that $(\mathrm{LP})_\mathrm{p}$ benefits from the possibility to split the machines in comparison to the IP. In the case of only two machines, it is not possible to schedule a subset $J' \subset J$ exclusively on a set of fractional machines, such that their machine value sum is no integer value. If so, either the sum of machine values for jobs in J' or the sum of machine values for the remaining jobs would be smaller than 1, such that constraint (3.6) could not be satisfied. Therefore the possibility to create fractional schedules is highly restricted for $m = 2$. It remains unclear, if this idea can be extended to a correct proof of the conjecture.

Note that even if the conjecture could be proved, it does not hold for the case of $m \geq 3$, as the following theorem shows.

Theorem 3.6. *For a number of $m \geq 3$ machines there are instances of the BWCTP with $p \equiv 1$, such that each optimal solution of $(LP)_p$ is fractional.*

Proof. The theorem is proved by giving a corresponding instance for the case of $m = 3$. This instance could be extended to an arbitrary number $m > 3$ of machines, by adding a set of $m - 3$ new jobs with big leader weights w^1 and a $m - 3$ new machines, to schedule these big jobs on.

Example 3.7. Suppose an instance of the BWCTP with jobs $J = \{1, \dots, 5\}$ indexed according to $>_F$, unit length processing times and the following weights of the leader:

$$\begin{aligned} w_1^1 = w_5^1 &= 3 \\ w_2^1 = w_3^1 &= 2 \\ w_4^1 &= 10 \end{aligned}$$

Then the following schedule S, and therefore also the corresponding solution for $(\text{IP})_\text{p}$ generated by using Theorem 3.1, is optimal on $m = 3$ machines.

Pos.	1	2
	1	2
	3	5
	4	

It holds that $c(S) = 1 \cdot (3 + 2 + 10) + 2 \cdot (2 + 3) = 25$. To prove the optimality, note that job 4 has to be scheduled on the first position, otherwise the objective value of the schedule would be at least $2 \cdot 10 + 3 + 3 + 2 + 2 = 30$. The best jobs to schedule on the remaining two first positions for the leader would be 1 and 5. Then job 2 and 3 have to be scheduled behind 1, which results in an objective value of $10+3+3+2\cdot 2+3\cdot 2 = 26$, which is a higher value than the one of S. Since job 1 is always on position 1 and the jobs 2 and 3 are exchangeable, S is the best solution possible.

Now consider the following feasible fractional machine schedule $\bar{S}$ with 6 machines, having machine values $v_M \equiv \frac{1}{2}$.

Pos.	1	2
	1	2
	1	3
	2	3
	4	5
	4	
	5	

The objective value of the schedule, and hence also the one of the corresponding solution of $(\text{LP})_\text{p}$ using Theorem 3.4, is $c(\bar{S}) = 1\cdot(3+\frac{2}{2}+10+\frac{3}{2})+2\cdot(\frac{2}{2}+2+\frac{3}{2}) = 24.5$ and therefore smaller than the optimal integer solution's objective function value. Hence $(\text{LP})_\text{p}$ has no optimal integer solution in this case.

$\square$

Although in this example the optimal solution's objective values of $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ do not coincide, the lower bound of the LP relaxation seems to be relatively near to the optimal value and would therefore be a good basis for approximation algorithms.

One possible way to create a feasible integer solution for the BWCTP with $p \equiv 1$ is the heuristic presented by Kis and Kovács [KisKov2012] for the equivalent case of $w^1 \equiv 1$. As described in the end of Section 1.2.2, it is based on the iterative use of $(\mathrm{LP})_\mathrm{KK}$, or $(\mathrm{LP})_\mathrm{p}$ in the here considered case, on subproblems of the original instance, which consist of all still unscheduled jobs and machines. For this reason, it would be quite complicated to perform a worst case analysis of the heuristic, since it is not known in advance how the optimal solutions of the LP will look like. Additionally there are many possible ways to choose a job to machine assignment from the optimal solution of $(\mathrm{LP})_\mathrm{p}$ in the described way, which would all have to be considered in the analysis. Therefore it remains unclear if the heuristic presented by Kis and Kovács is an approximation algorithm.

Another approach is to use the "min-increase" algorithm, which was presented in Subsection 2.3.2. Here the optimal solution $\bar{x}^*$ of $(\mathrm{LP})_\mathrm{p}$ or the corresponding optimal fractional machine solution $\bar{S}^*$ is used to obtain an ordered list of the jobs, from which afterwards a feasible min-increase schedule is created. The value the order is based on is the *average position* $\tilde{\rho}_j$ of a job j, which is defined as:

$$\tilde{\rho}_j = \sum_{\rho=1}^{m_j} \rho \sum_{M \in \mathcal{M}_j^\rho} v_M = \sum_{\rho=1}^{m_j} \rho \cdot \bar{x}^*_{j,\rho}$$

Based on this the jobs are processed in *ascending average LP position order* $>_\sim$, which is defined for two jobs j and k as:

$$j <_\sim k \Leftrightarrow \tilde{\rho}_j < \tilde{\rho}_k$$

As for the other in Subsection 2.3.2 presented job orders, we can also prove for $>_\sim$, that the min-increase algorithm using a job list sorted according to the given order does not admit a constant approximation ratio for the min-increase scheduling.

Theorem 3.8. *There cannot be a constant value $\alpha > 0$ for each instance I of the BWCTP, such that for the corresponding solution S of the min-increase algorithm using a job list ordered according to $>_\sim$ and the optimal solution S^* of this instance I of the BWCTP holds:*

$$c(S) \leq \alpha c(S^*)$$

Proof. To prove the theorem we again provide a corresponding counter example instance.

Example 3.9. Let I be an instance of the BWCTP with $m = 3$ and $n+12$ jobs, which are indexed according to $>_F$ and have unit length processing times. In addition the following weights of the leader are given:

$$\begin{aligned} w_i^1 &= \epsilon && \forall i = 1, \dots, n \\ w_{n+i}^1 &= M && \forall i = 1, \dots, 12 \text{ with } M \gg \epsilon \end{aligned}$$

Then it is optimal for the BWCTP to schedule the light weighted jobs $1, \dots, n$ to one machine and distribute the heavy weighted jobs $n+1, \dots, n+12$ equally over the remaining two machines, since the heavy jobs should be scheduled as early as possible and each light job would shift a heavy job one position behind if being scheduled on the same machine. The objective value of the optimal solution S^* can be denoted by:

$$c(S^*) = \epsilon \cdot \frac{n(n+1)}{2} + M \cdot 2(\frac{6 \cdot 7}{2}) = M \cdot 42 + \epsilon \cdot \frac{n^2+n}{2}$$

The same argument holds also in the case of fractional machine schedules. Therefore the solution of $(\mathrm{LP})_\mathrm{p}$ corresponding to the following feasible fractional machine schedule, in which the first machine has a machine value of 1 and the other machines have a machine value of $\frac{1}{3}$, is also optimal with the same objective function value as S^*.

Pos.	1	2	3	4	5	6	7	...	n
	1	2	3	4	5	6	7	...	n
	$n+1$	$n+2$	$n+3$	$n+4$	$n+6$	$n+8$			
	$n+1$	$n+2$	$n+4$	$n+5$	$n+6$	$n+10$			
	$n+1$	$n+3$	$n+4$	$n+5$	$n+7$	$n+11$			
	$n+2$	$n+3$	$n+5$	$n+7$	$n+9$	$n+12$			
	$n+6$	$n+8$	$n+9$	$n+10$	$n+11$	$n+12$			
	$n+7$	$n+8$	$n+9$	$n+10$	$n+11$	$n+12$			

This schedule results in average position values of $\tilde{\rho}_j = j$ for $j \in \{1, \dots, n\}$ and the following values rounded to one decimal place for the other twelve jobs:

job	$n+1$	$n+2$	$n+3$	$n+4$	$n+5$	$n+6$
$\tilde{\rho}$	1.0	1.7	2.3	3.3	3.7	3.7

job	$n+7$	$n+8$	$n+9$	$n+10$	$n+11$	$n+12$
$\tilde{\rho}$	3.3	3.3	3.7	4.7	5.3	6.0

The list to process the jobs for the min-increase algorithm is then the following up to permutations of jobs with equivalent values:

$$1 \quad | \, n+1 \, | \, n+2 \, | \quad 2 \quad | \, n+3 \, | \quad 3 \quad | \, n+4 \, | \, n+7 \, | \, n+8 \, | \, n+5 \, | \, n+6 \, | \, n+9 \, | \ldots$$

Therefore the jobs 1, $n+1$ and $n+2$ will occupy the first positions on the three machines. Now each job in $2, \ldots, n$ will be scheduled on the machine of job 1, since it would otherwise shift some heavy weighted job one position behind. For the heavy weighted jobs it holds, that they will be assigned on that machine, which causes a minimal sum of positions of all so far scheduled heavy jobs, since this minimizes the overall objective function value.

Hence the min-increase schedule will have the following structure, up to possible permutations of the heavy jobs, after all jobs with an average position of 3.3 or less are scheduled:

Pos.	1	2	3	4
	1	2	3	
	$n+1$	$n+3$	$n+7$	
	$n+2$	$n+4$	$n+8$	

The next jobs to schedule are $n+5$, $n+6$ and $n+9$, occurring in an arbitrary order. Because of the described behavior in the assigning of heavy weighted jobs, at least one of them will be scheduled on the machine of the jobs 1, 2 and 3. This job will now be shifted successively to position $n+1$ on this machine, since the jobs $4, \ldots, n$ will be scheduled in front of it. Hence the final min-increase schedule S has an objective function value of:

$$c(S) > (n+1) \cdot M + \epsilon \cdot \frac{n^2+n}{2}$$

Therefore there is no constant α such that $c(S) \leq \alpha c(S^*)$ holds for any n.

$\square$

Another well known idea to create an integer solution out of some fractional solution is rounding. In the case of dealing with 0-1-variables like in the given model $(\mathrm{IP})_{\mathrm{p}}$, often the randomized rounding algorithm is used. It interprets the fractional solution as a set of probabilities and constructs a random integer solution based on that. The corresponding analysis of the objective function value of the random solution first evaluates the expected objective value and afterwards creates an approximative solution by derandomization techniques. This approach does not really fit to the given model $(\mathrm{IP})_{\mathrm{p}}$, since it is not clear, if the randomized solution respects the constraint (3.3) which guarantees the correct order of the follower. Even then the maximal number of jobs on each position can be exceeded. Therefore the randomized rounding approach is not applicable to the given optimal solution of $(\mathrm{LP})_\mathrm{p}$.

In the last approach to create a feasible integer solution from the optimal fractional machine schedule respectively the corresponding optimal solution $\bar{x}^*$ of $(\mathrm{LP})_\mathrm{p}$, a constant approximation ratio of the created integer solution is obtained by construction. Suppose a constant value of α_D is given. Then the task is to find a feasible schedule S using up to m machines and respecting the follower's order on the machines, such that each job j is scheduled on a position ρ_j smaller than its average position $\tilde{\rho}_j$ of the LP solution multiplied with α_D. Then the following result can be concluded, which proves the approximation factor of α_D:

$$c(S) = \sum_{j=1}^{n} \rho_j w_j^1 \leq \sum_{j=1}^{n} \alpha_D \tilde{\rho}_j w_j^1 = \alpha_D \sum_{j=1}^{n} \sum_{\rho=1}^{m_j} \rho x_{j,\rho} \cdot w_j^1 = \alpha_D c(\bar{x}^*)$$

The task of finding such a feasible solution S can be reformulated into the following deadline problem with deadlines $d_j = \alpha_D \cdot \tilde{\rho}_j$:

Problem 3.10. *Given a set J of jobs with unit length processing times, deadlines $d_j \geq 1$ for $j \in J$ and a global order $>_F$ of the jobs. Then the task is to find a schedule with job positions ρ_j for job j that minimizes the number of used machines under the following conditions:*

- *The deadlines are met, such that $\rho_j \leq d_j$ holds for each job $j \in J$.*
- *The jobs have to respect the global job order $>_F$ on each machine, that is, given two jobs j and k on one machine, it has to hold that $j >_F k \Rightarrow \rho_j < \rho_k$.*

By assuming the jobs are indexed according to $>_F$, this problem can be solved in polynomial time with the following algorithm $\mathcal{A}_D$ presented in Algorithm 5. In this algorithm the jobs are processed by ascending indices, that is, according to the follower's order. For each job j the machines M are checked by ascending index order for a possible assignment of j on M. Here job j is assigned to the last position of M, if the current load L_M of M is smaller than $d_j - 1$. If there is no machine that fulfills this condition, a new machine is created and job j is scheduled on the first position of this machines.

For this algorithm $\mathcal{A}_D$ we can prove the following theorem.

Theorem 3.11. *Algorithm $\mathcal{A}_D$ returns a feasible solution for Problem 3.10 in polynomial time.*

Algorithm 5: Algorithm $\mathcal{A}_D$

```
1 $\mathcal{M} \leftarrow \emptyset$ ; // set of used machines
2 foreach job $j \in \{1, \dots, n\}$ do
3     foreach machine $M \in \{1, \dots, |\mathcal{M}|\}$ do
4         if Load $L_M$ of $M$ is smaller than $d_j - 1$ then
5             Schedule $j$ on the last position of $M$;
6             Continue with next job;
7     Create machine $M = |\mathcal{M}| + 1$ and schedule $j$ on this machine;
8     $\mathcal{M} \leftarrow \mathcal{M} \cup \{M\}$;
```

Proof. Since the jobs are processed according to the follower's order $>_F$ and new jobs are scheduled on the last position of a machine, the global job order is respected on each machine. In addition a job j is only scheduled on a machine, if its deadline will be met, which is guaranteed by the if-clause in line 4, since the processing times are equal to 1. Hence algorithm $\mathcal{A}_D$ finds a feasible solution for the deadline problem 3.10. Furthermore it runs in polynomial time since the processing of each job needs at most $|\mathcal{M}| < n$ iterations. Therefore the total runtime is in $O(n^2)$. □

To prove the optimality of the algorithm, we first verify the following lemma.

Lemma 3.12. *In any current schedule S_j of algorithm $\mathcal{A}_D$, scheduling the jobs $\{1, \dots, j\}$, the load of the machines does not increase with increasing machine index.*

Proof. For S_1 there is only one machine in use. Hence the observation holds.

Suppose there is a schedule S_j at with the load L_M of some machine M is bigger than the load $L_{M'}$ of some machine $M' < M$ and let j be the smallest job of that kind. Let L^j be the load on the machines *before* the last so far scheduled job j was assigned to the them. Since j is the smallest job of that kind, the loads of the machines M and M' have to be according to the statement of the lemma, that is $L^j_{M'} \geq L^j_M$. Additionally the load of a machine increases exactly by 1 for each new job on this machine. Since $L_{M'} < L_M$ is true, we can conclude that $L^j_{M'} = L^j_M$ holds and that job j was assigned to machine M.

The assignment of job j was based on L^j_M. Therefore j could have been scheduled to M', too. This is a contradiction, because M' was checked for the assignment of j before M and therefore j would have been scheduled to M'. Hence there cannot be a schedule S_j as described. □

Based on this it is possible to prove the following theorem:

Theorem 3.13. *If is there a feasible solution S' for the deadline problem which uses m machines, then the solution S^D generated by algorithm $\mathcal{A}_D$ uses $|\mathcal{M}| \leq m$ machines. Hence the algorithm solves the deadline problem 3.10 to optimality.*

The proof of the theorem uses the following lemma:

Lemma 3.14. *If for all $j \in J$ it holds, that the position ρ'_j of j in S' is equal to the deadline d_j, then it holds for the position ρ^D_j of job j in S^D, that $\rho^D_j = \rho'_j$.*

Proof. The lemma is proved by induction on the job index. The first job can only be scheduled on the first position in S' as well as in S^D.

Now suppose as an induction hypothesis that for all jobs $k < j$ it holds that $\rho^D_k = \rho'_k$. Since $d_j = \rho'_j$, the job j is scheduled on the first machine with a load of less than ρ'_j jobs. In the subsolution S'_{j-1} of S' with jobs $1, \dots, j-1$, there is a machine with load of $\rho'_j - 1$, namely the one j will be scheduled on. Hence there are more jobs on position $\rho'_j - 1$ than on position ρ'_j in S'_{j-1}. Because of the induction hypothesis, this also holds in the current subsolution S^D_{j-1} of S^D with jobs $1, \dots, j-1$ and therefore there is also a machine with load of $\rho'_j - 1$ in S^D. Due to Lemma 3.12 the machines are checked for a possible assignment of job j in decreasing load order. Hence j will be scheduled on some machine with load $\rho'_j - 1$ on position $\rho^D_j = \rho'_j$. □

Now Theorem 3.13 can be proved.

Proof of Theorem 3.13. If for all jobs $j \in J$ it holds that $d_j = \rho'_j$, then $\rho' \equiv \rho^D$ holds. Thus the number of machines used in S' and S^D coincide, since it is equal to the number of jobs on position 1.

If there are jobs with $d_j > \rho'_j$, then compare S^D with the solution $S^=$ of algorithm $\mathcal{A}_D$ under the assumption of $d_j = \rho'_j$ for all jobs $j \in J$. We claim, that if the positions used by the jobs are both sorted descending, the i-th biggest position in the subschedule S^D_j of S^D with jobs $\{1, \dots, j\}$ is greater or equal to the i-th biggest position in the subschedule $S^=_j$ of $S^=$ with jobs $\{1, \dots, j\}$. From this it can be deduced that there are at least as many jobs on the first position in $S^=$ as in S^D. Therefore the schedule S^D uses at most as many machines as schedule $S^=$, which would prove the theorem, since $S^=$ uses as many machine as S'.

To prove the claim, note that the deadlines in S^D are greater or equal than those in $S^=$. Hence the jobs can take bigger positions in S^D than in $S^=$. Therefore there are only two possibilities for some job j to be scheduled on a lower position in $S^=$ than in S^D. The first is, that the position it takes in $S^=$ is blocked in S^D by some job scheduled on a bigger position and earlier machine in S^D than in $S^=$. In the second case, a job preceding job j in $S^{=}$ is scheduled on some earlier machine on higher position in S^D. Thus the position of j in $S^{=}$ has no direct job in front of it in S^D and can therefore not be taken.

In both cases, there is a free slot to schedule j in S^D, which was occupied by some job $k < j$ in $S^{=}$. Therefore the positions coincide also in these situations, which proves the theorem. $\square$

Corollary 3.15. *From Theorem 3.11 and Theorem 3.13 above follows, that algorithm $\mathcal{A}_D$ solves the Problem 3.10 to optimality in polynomial time.*

However, the corollary does not guarantee, that the optimal solution of the problem is smaller or equal to m for the given value of α_D. To prove this, we would need an argument for some constant α_D, that ensures the existence of a feasible solution of the corresponding deadline problem. In this case of such an argument algorithm $\mathcal{A}_D$ would find a feasible solution of this kind, as shown in the above theorem. Unfortunately such an argument is not known so far for any constant α_D.

3.2. An integer quadratic programming model for the general BWCTP

The last approach to find a good solution for the BWCTP is based on a second model of the problem, which holds for the general case without fixed processing times or leader weights. Here the variables $a_{i,j} \in \{0,1\}$ for some job $j \in J$ and a machine i of the set of machines $\mathcal{M}$ denotes, if job j is scheduled on machine i or not. Then the general BWCTP is exactly described by the following integer quadratic program (IQP), assuming the jobs being indexed according to $>_F$.

(IQP)

$$\text{Objective:} \qquad \min_a c(a) = \sum_{i \in \mathcal{M}} \sum_{j \in J} a_{i,j} w_j^1 \left(p_j + \sum_{k \in J: k<j} a_{i,k} p_k \right)$$

$$\sum_{i \in \mathcal{M}} a_{i,j} = 1 \quad \forall j \in J \tag{3.9}$$

$$\text{assignment variables:} \qquad a_{i,j} \in \{0,1\} \quad \forall i \in \mathcal{M}, j \in J \tag{3.10}$$

In the corresponding non-integer quadratic programming relaxation (QP), the last constraint (3.10) is replaced by $a_{i,j} \geq 0 \quad \forall i \in \mathcal{M}, j \in J$, where the missing upper bound of 1 is already covered by constraint (3.9), similar to $(\mathrm{LP})_\mathrm{p}$.

For the model the following theorem could be proved.

Theorem 3.16. *Each feasible schedule S of the $BWCTP$ corresponds to a feasible solution a of (IQP) with equivalent objective value and vice versa. Therefore it holds, that the formulation (IQP) is an exact description of the $BWCTP$.*

Proof. To prove the theorem, we observe that the made decision in (IQP) is equivalent to the decision of the leader in the BWCTP. In both cases for each job exactly one machine is chosen to schedule the job on, which is in (IQP) guaranteed by constraint (3.9). Therefore each feasible S, which is based on a corresponding job to machine assignment and the given order $>_F$ of the follower, corresponds to a feasible solution a of (IQP).

Hence the only thing left to be proved is the equivalence of the objective function value of S and a. For S the objective function value is given by $\sum_{j\in J} w_j^1 C_j$, where C_j denotes the completion time of job j. Since the jobs have to be ordered ascending by index on the machines, the completion time of some job j is equal to the sum of processing times of jobs with smaller or equal index, that are scheduled on the same machine as j. This is exactly, what the term $\left(p_j + \sum_{k\in J:k<j} a_{i,k} p_k\right)$ in the objective function of (IQP) describes for some job j on machine i. Therefore the objective function of (IQP) coincides with the one of the BWCTP. $\square$

Note that the objective function of (IQP) can be reformulated as follows:

$$c(a) = \sum_{i\in\mathcal{M}} \sum_{j\in J} a_{i,j} w_j^1 \left(\sum_{k\in J:k<j} a_{i,k} p_k \right) + \sum_{j\in J} \sum_{i\in\mathcal{M}} a_{i,j} w_j^1 p_j$$

Because of constraint (3.9), the last summand is equal to the constant value $\sum_{j\in J} w_j^1 p_j$ and could hence be skipped in the objective function without changing the optimal solution set.

As another result, we can prove that (QP) has always an optimal integer value. From this follows, that solving (IQP) is equivalent to solving (QP), which will be shown in the following theorem.

Theorem 3.17. *The quadratic programming relaxation (QP) of the integer quadratic programming formulation (IQP) always admits an optimal integer solution.*

Proof. Suppose $\bar{a}^*$ is an optimal fractional solution for (QP) and j is a job, such that there is a machine i with $0 < \bar{a}^*_{i,j} < 1$.

Let $c_{i,j}$ be the objective contribution of variable $a_{i,j}$. This means that a change of the value of $a_{i,j}$ by some ϵ results in a change of the objective function value of $\epsilon \cdot c_{i,j}$, if the other variables stay the same. Since each $a_{i,j}$ appears in the objective function as a multiple of w_j^1 in front of the completion time term and also in the completion time term itself, $c_{i,j}$ can be denoted as:

$$c_{i,j} = w_j^1 \cdot \left(p_j + \sum_{k \in J : k < j} a_{i,k} p_k \right) + p_j \sum_{k \in J : k > j} a_{i,k} w_k^1$$

Let $\bar{c}_{i,j}$ be the objective contribution of $\bar{a}^*_{i,j}$ for the given solution of $\bar{a}^*$. Then it is possible to identify machine $\bar{i}$, such that $\bar{c}_{\bar{i},j} \leq \bar{c}_{i,j}$ holds for all $i \in \mathcal{M}$. Since for some job j, the value of $\bar{c}_{i,j}$ does not depend on $\bar{a}^*_{i',j}$ for $i' \in \mathcal{M}, i' \neq i$, the value $\bar{c}_{i,j}$ and the choice of $\bar{i}$ stay unchanged if we change the value of $\bar{a}^*_{i,j}$ for this job j and some $i \in \mathcal{M}$. Based on this, a new solution $\bar{a}$ of (QP) is defined as:

$$\begin{aligned} \bar{a}_{i,j'} &:= \bar{a}^*_{i,j'} && \forall\, j' \in J, j' \neq j \\ \bar{a}_{i,j} &:= 0 && \forall\, i \in \mathcal{M}, i \neq \bar{i} \\ \bar{a}_{\bar{i},j} &:= 1 \end{aligned}$$

The constraint (3.9) is obviously met by $\bar{a}$ for j and since $\bar{a}^*$ was feasible, it is also fulfilled for all other jobs. For the objective function values of $\bar{a}$ and $\bar{a}^*$ it holds that:

$$\begin{aligned} c(\bar{a}) - c(\bar{a}^*) &= \sum_{i \neq \bar{i}} \bar{c}_{\bar{i},j} \bar{a}^*_{i,j} - \sum_{i \neq \bar{i}} \bar{c}_{i,j} \bar{a}^*_{i,j} \\ &= \sum_{i \neq \bar{i}} \underbrace{[\bar{c}_{\bar{i},j} - \bar{c}_{i,j}]}_{\leq 0 \text{ because of Def. of } \bar{i}} \bar{a}^*_{i,j} \leq 0 \end{aligned}$$

Hence $\bar{a}$ is also an optimal solution. We iterate this procedure for each remaining job j with fractional variables $a_{i,j}$ in $\bar{a}$ to obtain an optimal integer solution. □

From the proof of the theorem, the following can be deduced:

Corollary 3.18. *To obtain an optimal integer solution from an optimal fractional solution $\bar{a}^*$ of (QP), it is possible to choose for any job j assigned to multiple machines an arbitrary machine i with $\bar{a}^*_{i,j} > 0$ to schedule j on.*

Proof. Given a job j, such that $0 < \bar{a}^*_{i,j} < 1$ holds for a machine $i \in \mathcal{M}$. Suppose that there is $\bar{i} \neq i$ as defined in the proof of Theorem 3.17 and it holds that $\bar{c}_{\bar{i},j} < \bar{c}_{i,j}$. Then for the feasible solution $\bar{a}$ defined as in the proof, it would follow that $c(\bar{a}) - c(\bar{a}^*) < 0$, which would be a contradiction to the optimality of $\bar{a}^*$. Hence we can deduce, that $\bar{c}_{i,j} = \bar{c}_{\bar{i},j}$ holds for all $i \in \mathcal{M}$ with $0 < \bar{a}^*_{i,j} < 1$. Therefore $\bar{i}$ is arbitrarily chosen from these machines i, which proves the corollary. □

From the theorem follows, that the integer quadratic model above can be solved in the relaxed case without the integrality constraint to obtain a solution for the BWCTP. This is useful, but does not help to generate a feasible solution for the BWCTP in polynomial time, since quadratic programming is still $\mathcal{NP}$-hard in general, see [Sah1974].

For a similar integer quadratic programming model as the one stated above for the BWCTP, Skutella presented in [Sku2001] a way to convert it into a polynomially solvable model and create a 2-approximation algorithm for the one-level WCTP based on it. In his approach, he studied the problem of scheduling unrelated parallel machines, which is equivalent to the WCTP apart from the fact, that each j has specific processing times $p_{i,j}$ for each machine i. The problem can be denoted in short by $R||\sum_j w_j C_j$ in the three-field notation of Graham et al. [GraLawLenK1979] and is a generalization of the WCTP. Since also in the case of unrelated machines the "Smith's Rule" order $>_i$ on basis of the specific processing times on each machine i yield an optimal sorting for the jobs on the machines, it was possible to formulate an integer quadratic programming model similar to (IQP), which is given below as (IQP)$_\mathrm{R}$:

(IQP)$_\mathbf{R}$

$$\text{Objective:} \qquad \min_a c(a) = \sum_{i\in\mathcal{M}} \sum_{j\in J} a_{i,j} w_j \left(p_{ij} + \sum_{k\in J: k <_i j} a_{i,k} p_{ik} \right)$$

$$\sum_{i\in\mathcal{M}} a_{i,j} = 1 \quad \forall\, j \in J$$

$$\text{assignment variables:} \qquad a_{i,j} \in \{0,1\} \quad \forall\, i \in \mathcal{M}, j \in J$$

The differences between (IQP) and (IQP)$_\mathrm{R}$ are, that in the objective function the processing times p_j and p_k are replaced by the corresponding processing times $p_{i,j}$ respectively $p_{i,k}$ on machine i and that the sum of jobs with smaller index $\sum_{k\in J:k<j}$ is replaced by the sum over jobs with bigger smith rule value $\sum_{k\in J:k<_i j}$ on machine i. In addition, the leader's weights w^1 are replaced by the given weights w in the one-level case. It was proved in the paper of Skutella, that Theorem 3.17 and Corollary 3.18 also hold for (IQP)$_\mathrm{R}$ by using a randomized rounding algorithm and derandomization techniques. However, the result could also have been achieved by adopting the proofs given above.

To obtain the 2-approximation for the problem $R||\sum_j w_j C_j$ the objective function of (IQP)$_\mathrm{R}$ was reformulated as:

$$c(a) = b^T a + \frac{1}{2} a^T D a$$

Here, $a \in \mathbb{R}^{mn}$ is a vector consisting of the variables $a_{i,j}$, which are ordered ascending by machine indices and for variables of one machine i according to the "Smith's Rule" order $>_i$. Furthermore $b \in \mathbb{R}^{mn}$ is a vector given by $b_{i,j} = w_j p_{i,j}$ and $D = (d_{(ij)(hk)})$ is a symmetric $mn \times mn$-matrix with:

$$d_{(ij)(hk)} = \begin{cases} 0 & \text{if } i \neq h \text{ or } j = k \\ w_j p_{i,k} & \text{if } i = h \text{ and } k <_i j \\ w_k p_{i,j} & \text{if } i = h \text{ and } j <_i k \end{cases}$$

Because of the chosen order of variables $a_{i,j}$ in a, D consists of m diagonal block matrices D_i corresponding to the machines $i \in \mathcal{M}$. By denoting $p_{i,j}$ by p_j in a block D_i, it follows that:

$$D_i = \begin{pmatrix} 0 & w_2p_1 & w_3p_1 & \cdots & w_np_1 \\ w_2p_1 & 0 & w_3p_2 & \cdots & w_np_2 \\ w_3p_1 & w_3p_2 & 0 & \cdots & w_np_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_np_1 & w_np_2 & w_np_3 & \cdots & 0 \end{pmatrix}$$

A quadratic programming formulation as given it can be solved in polynomial time if the objective function c is convex, see for example [KhaKozTar1980] or [ChuMur1981]. In addition c is known to be convex if and only if D is positive semidefinite, which is the case if and only if the eigenvalues of each D_i are greater or equal to zero.

To produce a positive semidefinite matrix D' from D, Skutella adds a multiple γ of the vector b to the diagonal of D. By denoting with $\operatorname{diag}(b)$ the diagonal matrix with diagonal entries equivalent to b, the matrix D' can be stated as $D' = D + \gamma \operatorname{diag}(b)$. It was shown in the paper, that D' is positive semidefinite for $\gamma = 1$. Afterwards it was concluded, that it is a 2-approximation algorithm for the problem $R||\sum_j w_jC_j$ and hence also for the WCTP, to use the modified convex objective function $c'(a) = (1 - \frac{\gamma}{2}) \cdot b^T a + \frac{1}{2} a^T (D + \gamma \operatorname{diag}(b)) a$ to solve $(\text{IQP})_\text{R}$ and apply a derandomized rounding algorithm on the received optimal solution of the convex quadratic program.

In the case of the BWCTP, the same argumentation cannot be used to generate a constant factor approximation algorithm, which follows from the theorem below.

Theorem 3.19. *If we adopt the approach of Skutella [Sku2001] of generating a 2-approximation for the problem $R||\sum_j w_jC_j$ by using the integer quadratic programming model $(IQP)_R$ to the given model (IQP) of the BWCTP, the adopted matrix D' is not positive semidefinite for any arbitrary value of γ. Therefore the corresponding transformed quadratic program cannot be solved in polynomial time.*

Proof. To prove the theorem we try to adopt the approach of Skutella to the (IQP). Therefore we define the vector a in the bilevel case to be again sorted ascending by machine indices first and in the case of identical machines by the follower's order $>_F$, that is, ascending by job indices. Then $b \in \mathbb{R}^{mn}$ is a vector given by $b_{i,j} = w_j^1 p_j$ and $D = (d_{(ij)(hk)})$ is a symmetric $mn \times mn$-matrix with:

$$d_{(ij)(hk)} = \begin{cases} 0 & \text{if } i \neq h \text{ or } j = k \\ w_j^1 p_k & \text{if } i = h \text{ and } k < j \\ w_k^1 p_j & \text{if } i = h \text{ and } j < k \end{cases}$$

Thus the diagonal block matrices D_i of D have exactly the same structure as in the approach of Skutella and are additionally all equivalent for different machines i, since $p_{i,j} = p_j$ holds.

Now consider the following instance of the BWCTP with the jobs 1 and 2:

$$\begin{aligned} p_1 = p_2 = w_1^1 = w_2^2 &= 1 \\ w_1^2 &= 2 \\ w_2^1 &= \gamma^2 + 1 \end{aligned}$$

Here the jobs are indexed according to $>_F$ and for any machine $i \in \mathcal{M}$ the diagonal block D_i' of $D' = D + \gamma \operatorname{diag}(b)$ can be denoted as:

$$D_i' = \begin{pmatrix} \gamma w_1^1 p_1 & w_2^1 p_1 \\ w_2^1 p_1 & \gamma w_2^1 p_2 \end{pmatrix}$$

Then we can finally conclude for the determinant of D_i':

$$\det(D_i') = \gamma^2 \cdot w_1^1 p_1 \cdot w_2^1 p_2 - (w_2^1 p_1)^2 = w_2^1(\gamma^2 - (\gamma^2 + 1)) = -w_2^1 < 0$$

Thus D_i' has a negative eigenvalue in this case and is therefore not positive semidefinite. Hence for each value of γ there is an instance of the BWCTP, in which c' is not convex. Therefore the corresponding transformed quadratic model is not polynomial solvable, which proves the theorem. □

Apart from the theoretical study of the model (IQP) given above, it was also analyzed empirically on a set of random instances in the following chapter.

4. Experimental evaluation

In the last chapter of this thesis, we evaluated some of the algorithms presented above to solve the BWCTP empirically for their performance in terms of the runtime of the algorithms and the quality of the resulting solution. First we will analyze the integer programming formulation $(\mathrm{IP})_\mathrm{p}$ and its LP relaxation $(\mathrm{LP})_\mathrm{p}$ presented in Section 3.1. Here the main focus lies on the size of the integrality gap and the possible factors it depends on. After this we examine the quadratic programming model (IQP) from Section 3.2, where we consider only one, the integral or the relaxed model, since both have the same optimal solution value because of Theorem 3.17. The last regarded algorithm is the min-increase scheduling algorithm, presented in part 2.3.2. Since it depends on the given list to process the jobs, the analysis concentrates on the comparison of the different job orders presented in the above thesis.

To evaluate the different approaches, we run each of them on a set of randomly generated instances of the BWCTP. The number of jobs n in the instances was chosen from $\{10, 15, \dots, 45, 50, 60, \dots, 150\}$. We created for each value of n, a set of 10 different instances, where one instance consists of n jobs with randomly chosen values for the processing times p, the leader's weights w^1 and the follower's weights w^2. The random values for the different parameters of each job were equally distributed integer number from the set $\{1, 2, \dots, 10^5\}$. In addition to the instances on arbitrarily processing times, we created a set of 10 instances with fixed processing times $p \equiv 1$ for each job number, since for example $(\mathrm{IP})_\mathrm{p}$ is only applicable to this kind of instances. The weights were again randomly chosen in the described way.

An instance of the ones given above was then run on a set of m machines, where m could have values from $\{2, 5, 10, 20, 50\}$. For a given number n of jobs only those machine values m are valid, for which $m \leq n$ holds. Furthermore, for each run of n jobs on m machines we executed a second run, in which the weights of the leader were re-sorted, such that $>_L$ is reversed to the given order $>_F$ of the follower. This case was proved to be the worst possible job order for the leader in Theorem 2.4 and will therefore be referred to as the worst case run.

To analyze the performance of the runs, we classify the corresponding results by one of the following criteria, which will be defined below.

- number n of jobs
- number m of machines
- job to machine ratio
- ratio r_1
- ratio r_{12}
- ratio r_{12}^J
- inversion ratio

The number of jobs and the number of machines can be directly observed from the given instance. Based on these two values, the job to machine ratio is defined as the value $\frac{n}{m}$. The ratio r_1 describes the range in the leader's weights and was defined in Section 2.4 as:

$$r_1 := \max_{j,k \in J} \left\{ \frac{w_j^1}{w_k^1} \right\}$$

It was chosen, since some counter examples, like Example 2.26, use the fact that r_1 is unbounded. Additionally there is an approximation algorithm presented in the end of Section 2.4, whose worst cast ratio depends on r_1.

The results are also compared on the basis of the two ratios r_{12} and r_{12}^J. They are defined as:

$$r_{12} := \max \left\{ \max_{j,k \in J} \{ \frac{w_j^1}{w_k^2} \}, \max_{j,k \in J} \{ \frac{w_j^2}{w_k^1} \} \right\}$$
$$r_{12}^J := \max \left\{ \max_{j \in J} \{ \frac{w_j^1}{w_j^2} \}, \max_{j \in J} \{ \frac{w_j^2}{w_j^1} \} \right\}$$

From the definition it follows, that r_{12} represents the biggest possible factor between a weight of the leader and a weight of the follower. Therefore it is a measure to describes how similar the corresponding weight functions are. The ratio r_{12}^J is similar to r_{12}, since it represents the biggest occurring factor between the leader's and follower's weight of one job.

Both ratios r_{12} and r_{12}^J have the problem, that they depend on the actual weights w^2 of the follower and not on the resulting order of the follower $>_F$, which describes his behavior more exact. For this reason the last criterion to compare the results on, is the *inversion ratio* of an instance, which aims to measure the similarity of the leader's order $>_L$ and the follower's order $>_F$. Given a permutation π of size n, the number of inversions is defined as the number of pairs (i, j) with $i < j$ and $\pi(i) > \pi(j)$. Therefore the maximal number of inversions over all permutations is equal to the number of possible pairs, that is, $\binom{n}{2} = \frac{n\cdot(n-1)}{2}$. Based on this the inversion ratio of an instance of n jobs is defined as:

$$\frac{2(\#\text{pairs } j,k \in J \text{ with } j >_L k \text{ and } j <_F k)}{n \cdot (n-1)}$$

Note that if $j =_L k$ or $j =_F k$ holds, the pair (j,k) does not admit an inversion.

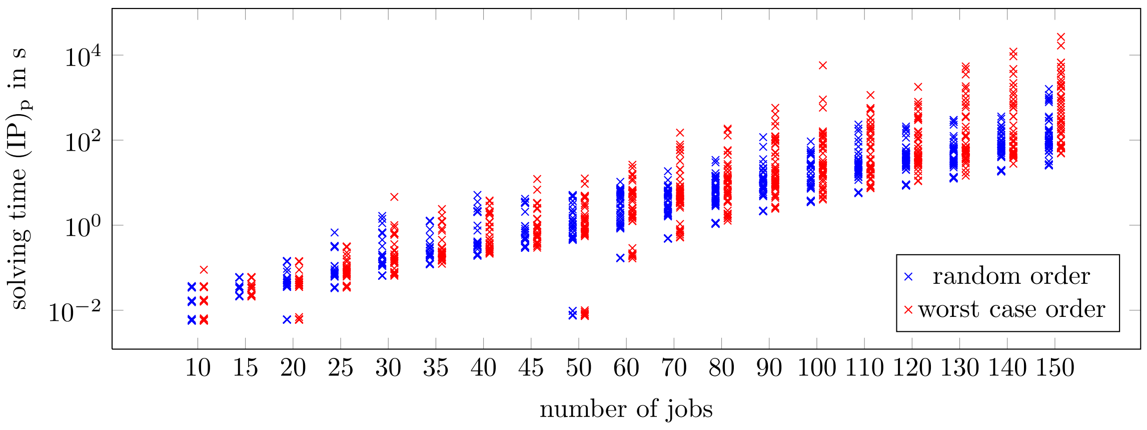
In the evaluation of the different runs, we will always choose exactly one of the above presented criteria in each picture. Additionally we will distinguish in each picture between the worst case instances, which will be colored red, and the randomly sorted instances, which will have the color blue. For all of the inversion ratio pictures, only the randomly sorted instances are printed, since the instances of the worst case have a constant inversion ratio of 1.

To obtain the presented results we used a computer with a 2.93 GHz quad-core Intel(R) Core(TM) i7 CPU and a memory of 15.6 GB RAM. For the solving of the models $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ of Section 3.1 as well as the model (IQP) of Section 3.2, the SCIP Optimization Suite (`http://scip.zib.de/`, last checked on July 11th, 2014) [Ach2009] was used. In addition the IPOPT library (`https://projects.coin-or.org/Ipopt`, last checked on July 11th, 2014) was included into SCIP to solve the non-linear models, which for its part uses HSL (*"HSL(2013). A collection of Fortran codes for large scale scientific computation. `http://www.hsl.rl.ac.uk`"*, last checked on July 11th, 2014) as a subroutine.

4.1. The integer programming model $(\mathrm{IP})_\mathrm{p}$

In this first section we analyze the results regarding $(\mathrm{IP})_\mathrm{p}$ and its LP relaxation $(\mathrm{LP})_\mathrm{p}$. Both were run for all $n \in \{10, 15, \ldots, 45, 50, 60, \ldots, 150\}$ and all machines m with $m \leq n$ on the instances with unit length processing times $p \equiv 1$.

The execution times for solving $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ to optimality are presented in Figure 4.1 classified by the number of jobs. We first observe, that the LP relaxation is solved much faster than the IP model. While all LP instances are solved after at most 62 seconds, the slowest instance of the IP needs nearly 1600 seconds in the case of the random order and 26811 seconds, which are over 7 hours, in the worst case. Apart from this result, it can be seen, that the solving times correlate with the given number of jobs. The sole exceptions are those instances with $n = m$ at the job numbers 10, 20 and 50, whose optimal solution of scheduling one job per machine is quickly detected by the solver. It is noticeable, that for $(\mathrm{IP})_\mathrm{p}$ the worst case instances are solved slower than the ones with randomly ordered jobs, especially for higher numbers of jobs. Surprisingly, in the case



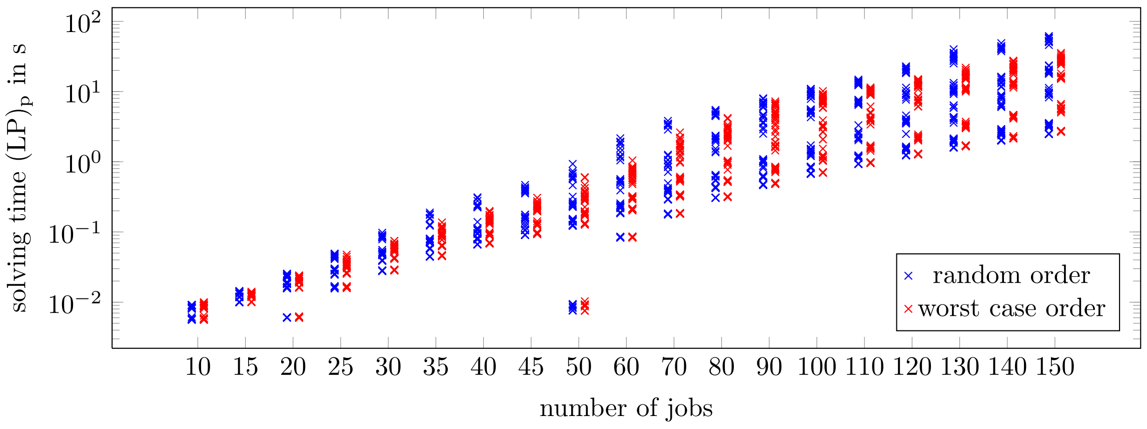


Figure 4.1.: Runtime of $(IP)_p$ and $(LP)_p$ by number of jobs

of $(LP)_p$, the exact opposite behavior can be observed. Here most of the solving times of the worst case instances are below those of the randomly sorted cases. Except for these findings, we noticed no other correlations between the solving time of the models and one of the other criteria mentioned above.

As a second result, the so-called integrality gap δ^I between the objective value of the optimal IP solution S^* and the objective value of the corresponding optimal LP solution $\bar{S}^*$ will be analyzed, which can be defined as follows:

$$\delta^I := \frac{c(S^*) - c(\bar{S}^*)}{c(\bar{S}^*)}$$

In Figure 4.2 we see the integrality gap of the instances for each number of jobs n. Note that all values of δ^I are smaller than 2% and all but five values are smaller than 1%. Thus the gaps are quite small and might therefore be to near to the smallest noticeable difference of the solver. In this case the obtained results may differ significantly from the correct theoretical optimal solution values. For this reason the following interpretations should be handled with care.

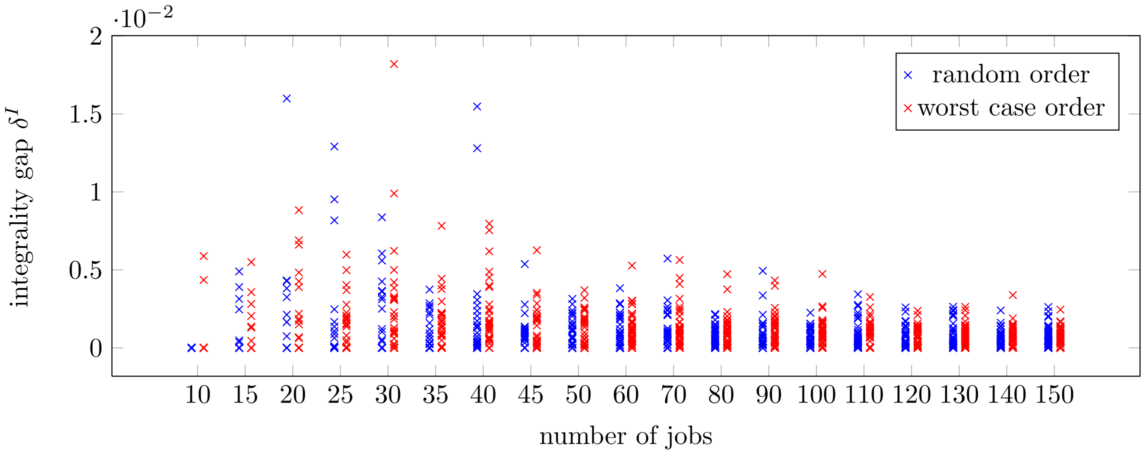


Figure 4.2.: Integrality gap δ^I between $(\text{IP})_\text{p}$ and $(\text{LP})_\text{p}$ by number of jobs

In contrast to the solving time of the instances, the integrality gap does not seem to depend on the distinction between the random ordered and the worst case, which can be deduce from Figure 4.2. However, the results vary for different n. While for big amounts of jobs the gap seems to be very small, it increases with decreasing n. There are also much more outliers for small n than for bigger ones. On the other hand, Figure 4.3 shows, that the portion of instances with integrality gap equals to zero also increases with decreasing number of jobs. For $n = 10$ it is almost one in the worst case and exactly one for the randomly ordered case.

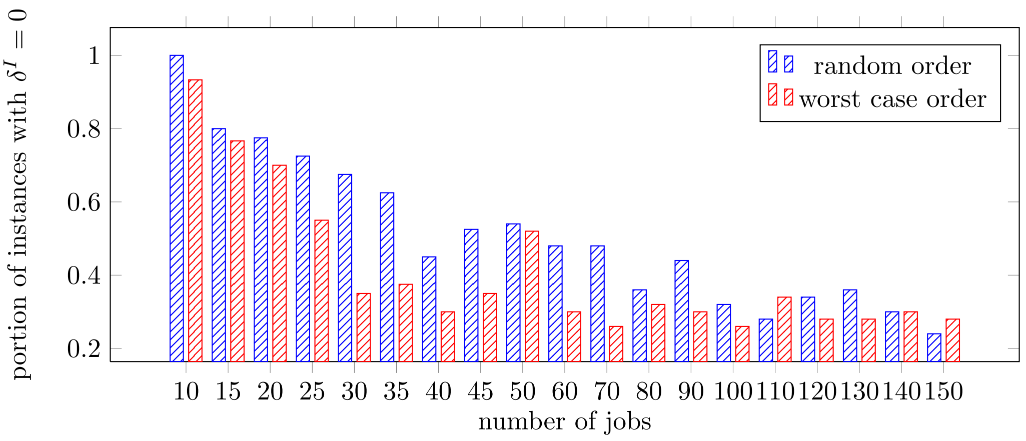


Figure 4.3.: Portion of instances with $\delta^I = 0$ by number of jobs

A possible explanation would be, that for small n the number of possible integer solutions is small and therefore also the number of integer solutions close to the optimal value $(\text{LP})_\text{p}$. Thus if the optimal solution value of $(\text{IP})_\text{p}$ is not equal to the corresponding optimal LP relaxation solution value, the possibility of a bigger integrality gap is also bigger for small n.

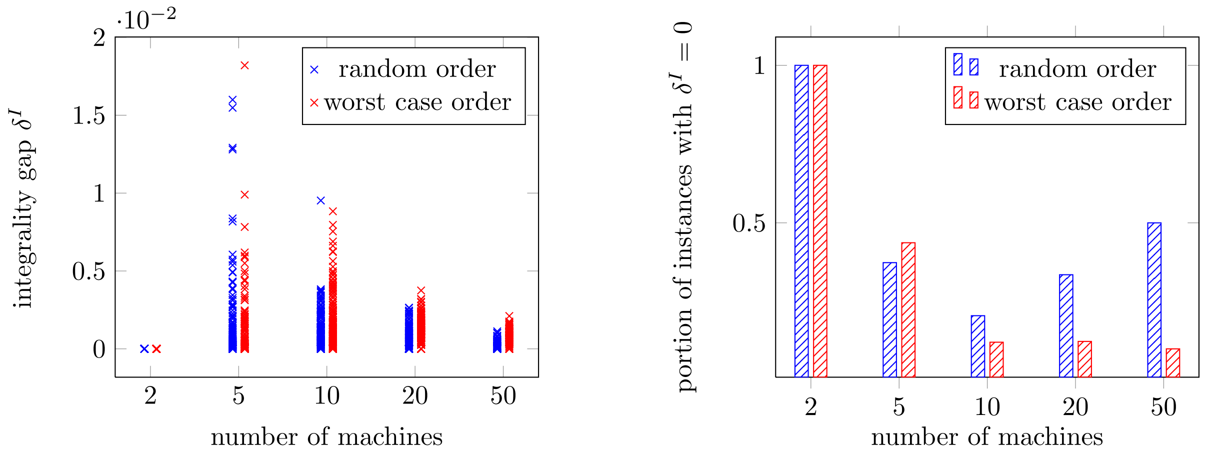


Figure 4.4.: Integrality gap δ^I between $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ by number of machines

When classifying the integrality gap results by the number of machines as in Figure 4.4, we instantly notice that it is equal to zero for every single instance if $m = 2$. The observation was already mentioned in Section 3.1 and led to Conjecture 3.5, stating an integrality gap of zero for arbitrary instances of the BWCTP for $p \equiv 1$ and $m = 2$. Apart from the $m = 2$ case, the gap decreases with increasing number of machines. Additionally correlations concerning the portion of instances with $\delta^I = 0$ or the worst case instances cannot be identified in the results.

The same holds for the other classifications, which can be seen from the Figures A.1 to A.5 in the appendix. Here the integrality gap of the instances is plotted against the job to machine ratio, the ratios r_1, r_{12} and r_{12}^J as well as the inversion ratio. Note that in these pictures the axis of the integrality gap is logarithmic to avoid that most of the marks stick together. Therefore only those instances with positive integrality gap $\delta^I > 0$ are plotted.

4.2. The quadratic programming model (IQP)

The next evaluated approach is the integer quadratic programming model (IQP) presented in Section 3.2. There it was shown in Theorem 3.17, that the optimal solution values of the model and its non-integer relaxation (QP) coincide. Hence the integrality gap would always be equal to zero. Therefore only one of the models needs to be solved. In some preliminary evaluations, we observed that both models could only be solved for small values of n and m. Furthermore (IQP) seems to be a bit faster than (QP). The reason for this unexpected behavior could be, that in the solving process of the linear relaxation many

LP submodels are solved, which is very time consuming. In contrast to this, the model having binary variables uses less of those submodel iterations. The branch and bound algorithm of the solver may rather exploit the relatively small number of possible integer solutions of the problem for small values for n and m, resulting in a faster overall solving process. Thus only (IQP) has been used for the analysis and not its linear relaxation.

As already mentioned above, the model is solves to optimality for a small number of machines and jobs. Therefore we considered only those instances with $n \in \{10, 15, 20, 25\}$ and $m \in \{2, 5, 10\}$ for this approach. In addition we analyzed only instances with $p \equiv 1$ to be able to compare the results to those of $(\mathrm{IP})_{\mathrm{p}}$. As a second reason for this restriction, we noticed that already the solving process of these instances lasts very long.

The main aspect of the analysis in this section is the solving time of the models respectively the achieved optimality gap δ^O. If we denote by S the best found feasible solution and by L the biggest found lower bound value for the optimal solution, δ^O can be defined as:

$$\delta^O := \frac{c(S) - L}{L}$$

Since the solving time and δ^O should both not depend on the parameters given by the specific weights of the leader and the follower, the obtained results are only classified by the number of jobs, the number of machines and the job to machine ratio.

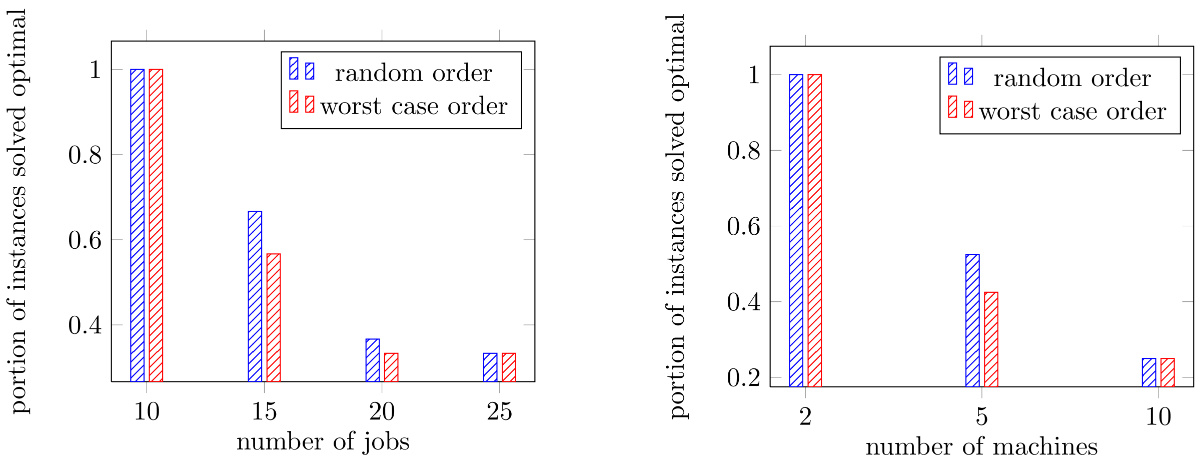


Figure 4.5.: Portion of instances that solved (IQP) to optimality by n and m

In Figure 4.5 the portion of those instances is shown, for which (IQP) could be solved to optimality. Only for the case of $n = 10$ and the case of $m = 2$ the proved optimal solution could be found for each corresponding instance. For each other value of n respectively m there are instances, for which the solving process was interrupted, because it took to much memory. In the cases of $n = 25$, even all instances apart from the $m = 2$ case cannot be solved. The same holds for instances of $m = 10$ apart from those having only 10 jobs. Thus the portion of solved instances decreases with increasing number of jobs or machines.

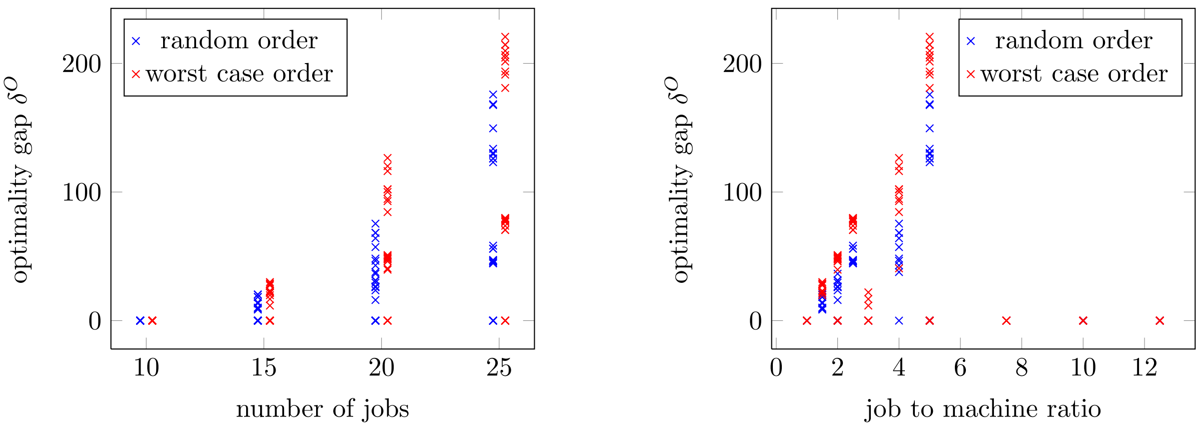


Figure 4.6.: Optimality gap δ^O for (IQP) by n and job to machine ratio

An overview of the achieved optimality gap for the instances is given in Figure 4.6. It becomes apparent, that the optimality gap increases with increasing number of jobs. Furthermore the optimality gap also increases for increasing job to machine ratio, apart from those instances solved to optimality, and hence also for a decreasing number of machines. The only exception are the instances of job to machine ratio equals to 3, that is, those instance with $n = 15$ and $m = 5$. This combination is the only one apart from the $n = 10$ respectively $m = 2$ combinations, in which the main portion of instances could be solved to optimality.

In contrast to Figure 4.5, where the worst case does not seem to be relevant, it clearly has an influence in Figure 4.6. Here, the optimality gap of the worst ordered instances is significantly higher than the one of the randomly sorted instances.

The last point we observed is, that for some combination of n and m, for which the model was not solved to optimality, the corresponding gap has not been improved for a long time before the solving process was interrupted. In single cases the share of the time in the solving process, at which the gap already has been equal to the finally achieved one, was higher than 95%. The Figures A.6 and A.7 in the appendix show this portion of solving time, in which the final gap is already achieved, plotted against the number of jobs, the number of machines and the job to machine ratio. From these pictures it can be seen, that there is no obvious pattern for the appearance of these instances. In addition, also the general solving time of the instances classified by n and m can be found in the appendix in Figure A.8.

4.3. Min-increase scheduling

In this last section of the experimental evaluation of algorithms for the BWCTP the min-increase scheduling algorithm is examined. As a list based approximation algorithm it depends on a given order, in which the jobs have to be processed. We have considered five different job orders for min-increase scheduling in this thesis:

- the "Smith's Rule" order using the leader's weights $>_L$
- the "Smith's Rule" order using the follower's weights $>_F$
- the sum of weighted processing times order $>_+$
- the product of weighted processing times order $>_\times$
- the ascending average LP position order $>_\sim$

The first four were defined in part 2.3.2, while the last one was introduced as a possibility to convert an optimal solution of the LP relaxation of $(\mathrm{IP})_\mathrm{p}$ into a feasible integer solution in Section 3.1.

For the analysis of the min-increase scheduling algorithm, all instances defined above were used. The evaluation is split based on the used processing times. First the $p \equiv 1$ case will be studied, where the main focus lies on the evaluation of the gap δ^* between the objective value of the min-increase solution S and the objective value of the optimal solution S^* of the BWCTP with $p \equiv 1$. The optimal solution of the BWCTP is given by the optimal solutions of $(\mathrm{IP})_\mathrm{p}$ in Section 4.1. Thus δ^* is defined as:

$$\delta^* := \frac{c(S) - c(S^*)}{c(S^*)}$$

In the case of arbitrary processing times, the job order $>_\sim$ is not available, since $(\mathrm{IP})_\mathrm{p}$ and therefore also the LP relaxation $(\mathrm{LP})_\mathrm{p}$ is only defined for $p \equiv 1$. For the same reason, the optimal solution of the BWCTP is unknown and therefore δ^* could not be determined. To deal with this we will compare the objective function value of a min-increase solution with the minimal min-increase solution objective value for a given instance of the BWCTP. Thus the different job orders can be evaluated, even if the optimal solution value of the corresponding instance of the BWCTP is not known. We denote by S^L, S^F, S^+ and $S^\times$ the min-increase solution obtained by the algorithm using the order $>_L$, $>_F$, $>_+$ respectively $>_\times$. Then S^M is that min-increase solution with the minimal objective function value for a given instance of the BWCTP, thus it holds that $c(S^M) = \min\{c(S^L), c(S^F), c(S^+), c(S^\times)\}$. Now we can define the gap δ^M between the objective value of a min-increase solution S and the corresponding minimal min-increase solution S^M for this instance as:

$$\delta^M := \frac{c(S) - c(S^M)}{c(S^M)}$$

As stated above, the gap value δ^M will be used to compare the different solutions of the min-increase scheduling using the four remaining job orders for the case of arbitrary processing times.

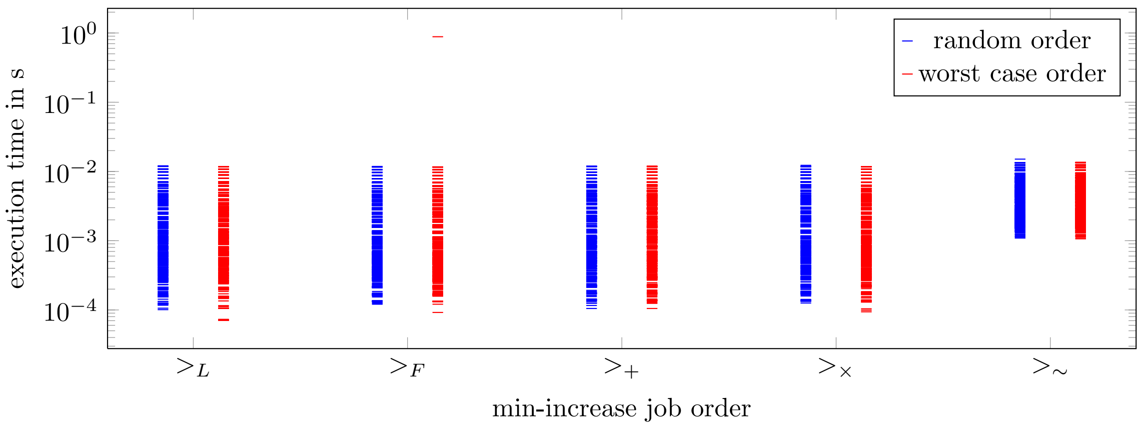


Figure 4.7.: Execution time of min-increase by job order

Results for $p \equiv 1$. We will first compare the five different orders for the min-increase algorithm on instances with $p \equiv 1$. Regarding the execution time of the algorithm presented in Figure 4.7, there are no big differences between them. Only for the last order $>_\sim$ the execution time of the fastest instances is slightly slower. This is due to the need to read the file containing the corresponding solution of the LP relaxation model, on which the order of the jobs is based. Nevertheless the min-increase algorithm is very fast for all job orders compared to the solving times of $(\mathrm{IP})_\mathrm{p}$ and (IQP) presented above.

In contrast to the execution time, the examination of the gap δ^* between the min-increase objective value and the optimal solution value yields different results for the job orders, which can be seen in Figure 4.8. The values for the min-increase algorithm using $>_\sim$ are the smallest in average. They are followed by the ones using $>_+$ and $>_\times$, that is, the orders based on both types of weights. The orders based only on the leader's or the follower's weight yield the worst results in average.

Apart from the average gap values, the orders behave also different when applying the worst case job sorting. While $>_F$ and $>_\sim$ have quite similar gap values comparing the random and the worst case order, the values for the other min-increase orders are

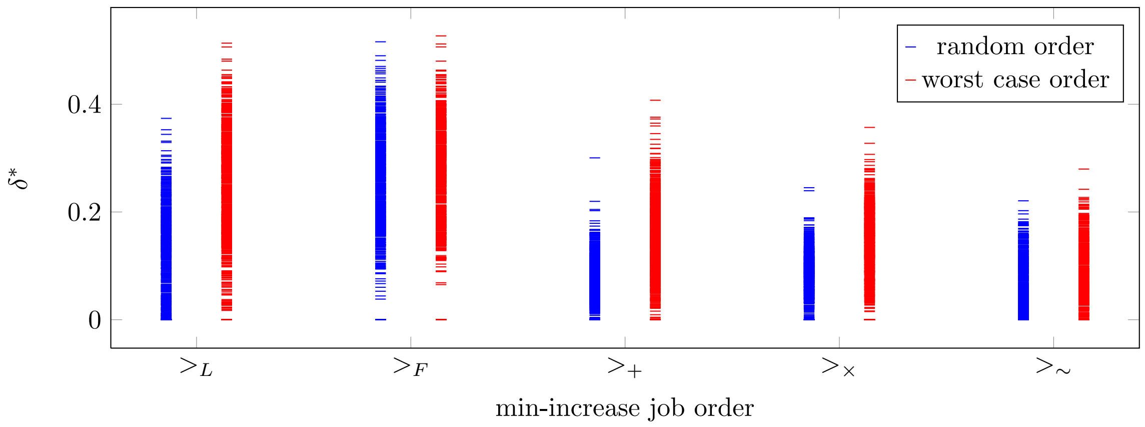


Figure 4.8.: Gap δ^* to the optimal solution by min-increase job order

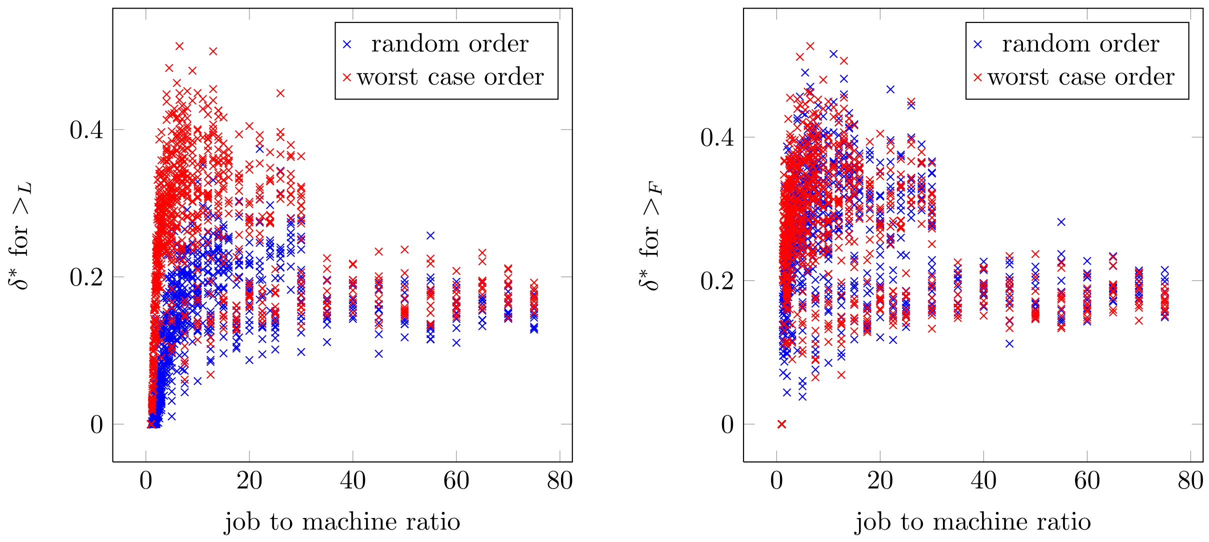
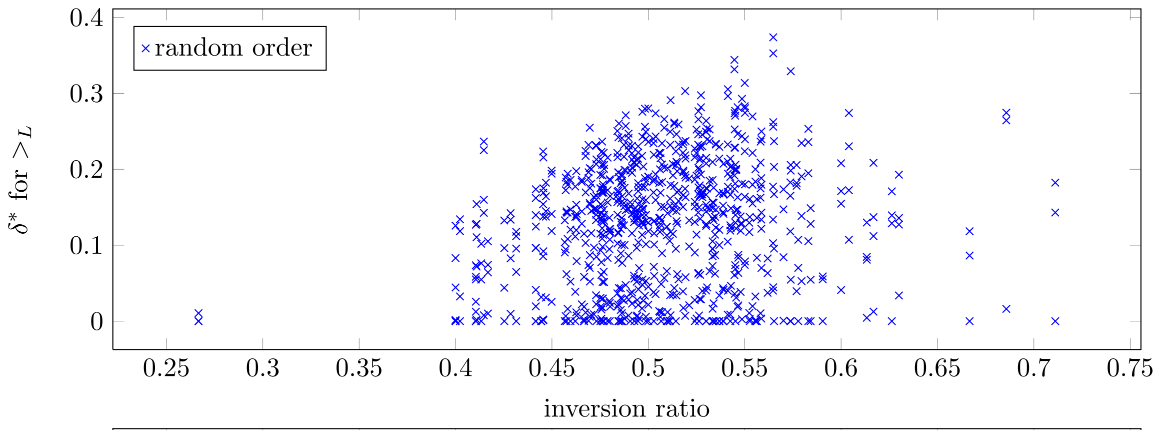


Figure 4.9.: Gap δ^* to the optimal solution for min-increase using $>_L$ and $>_F$ by job to machine ratio

significantly higher for the worst case instances. This can also be observed regarding Figure 4.9, where the gap for the job order $>_L$ and $>_F$ is plotted against the job to machine ratio. Here we can well recognize, that the worst case order influences the gap in the case of the leader's weight order, but not in the case of the follower's weight order. Note that the pictures plotting the gap against the job to machine ratio of the other orders can be looked up in the appendix in the Figures A.9, A.10 and A.11 for the job orders $>_+$, $>_\times$ and $>_\sim$ respectively.

A possible explanation for this behavior could be, that on the one hand the job order $>_F$ does not depend on the leader's weights at all. Since in the worst case these leader's weights are resorted, this does not affect the quality of the min-increase algorithm using a job list sorted according to $>_F$. On the other hand, the order $>_\sim$ is based



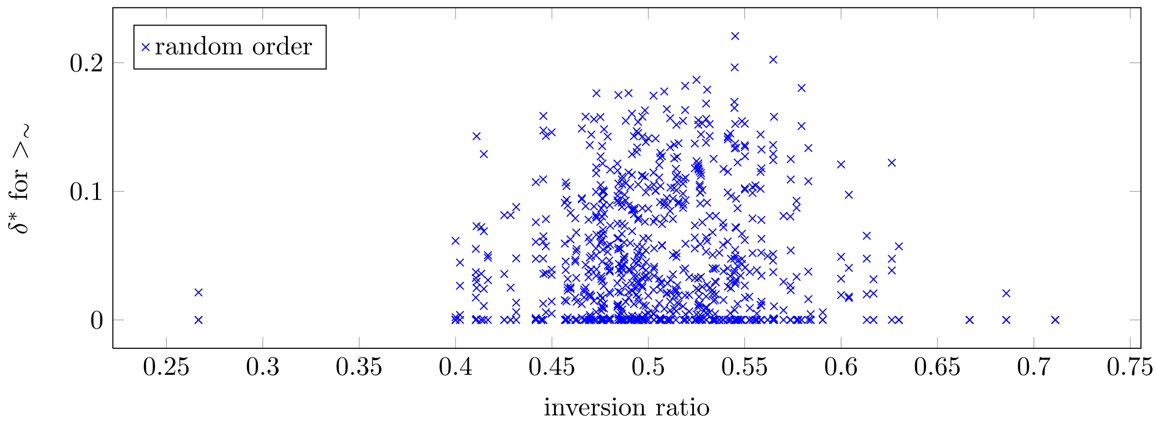


Figure 4.10.: Gap δ^* to the optimal solution for min-increase using $>_L$ and $>_\sim$ by inversion ratio

on the solution of $(\mathrm{LP})_\mathrm{p}$. Therefore this job order is likely to react best on the special structure of the problem having $>_L$ reversed to $>_F$. Hence also the min-increase schedule resulting from using a job list sorted according to $>_\sim$ depends only little on the worst case job sorting.

In contrast to this, the other min-increase job orders clearly admit worse gaps for the worst case order of the leader's weights. Thus one would guess, that this dependency on the order of the leader's weight could also be observed regarding Figure 4.10, in which the gap is plotted against the inversion ratio of the different instances for the job orders $>_L$ and $>_\sim$. In the case of the dependent job order $>_L$, we can indeed see a nearly linear increase in the maximum gaps with increasing inversion ratio in the interval from 0.4 to 0.55. On the other hand, a similar behavior is not visible for the values below the maximum or outside the chosen interval. Furthermore the distribution of the gap values look very similar in the case of the $>_\sim$ order, which does not seem to be much worst case dependent so far. Thus the examination of the inversion ratio pictures provide no clear result. This could also be due to the fact, that the inversion ratio is not equally distributed over $[0,1]$, but has most of the values lying around 0.5. Again the Figures A.12, A.13 and A.14 plotting the gap δ^* against the inversion ratio for the other job orders $>_F$, $>_+$ and $>_\times$ are presented in the appendix.

For the other classification criteria, that is, the number of jobs respectively machines and the ratios r_1, r_{12} and r_{12}^J, we did not observe any additional correlations to the gap δ^* of the min-increase solution value to the optimal one. This also holds for the above presented job to machine ratio.

Results for arbitrary p**.** Now we will present the results for the min-increase scheduling algorithm on instances with arbitrary processing times. As already explained above, there are neither optimal solutions of $(\mathrm{IP})_\mathrm{p}$ nor optimal solutions of its LP relaxation $(\mathrm{LP})_\mathrm{p}$ available. Therefore the job order $>_\sim$ cannot be formed and the gap value δ^M is calculated in comparison to the minimal objective value of a min-increase solution. Furthermore the "Smith's Rule" order of the leader and the follower are no longer independent, since both depend on the particular processing times of the instance.

The evaluation is again started by looking at the algorithms execution times. Like in the $p \equiv 1$ case, they do not differ for the different min-increase job orders, which is shown in Figure A.15 in the appendix. In Figure 4.11 it can be seen that this does not hold for the gap value.

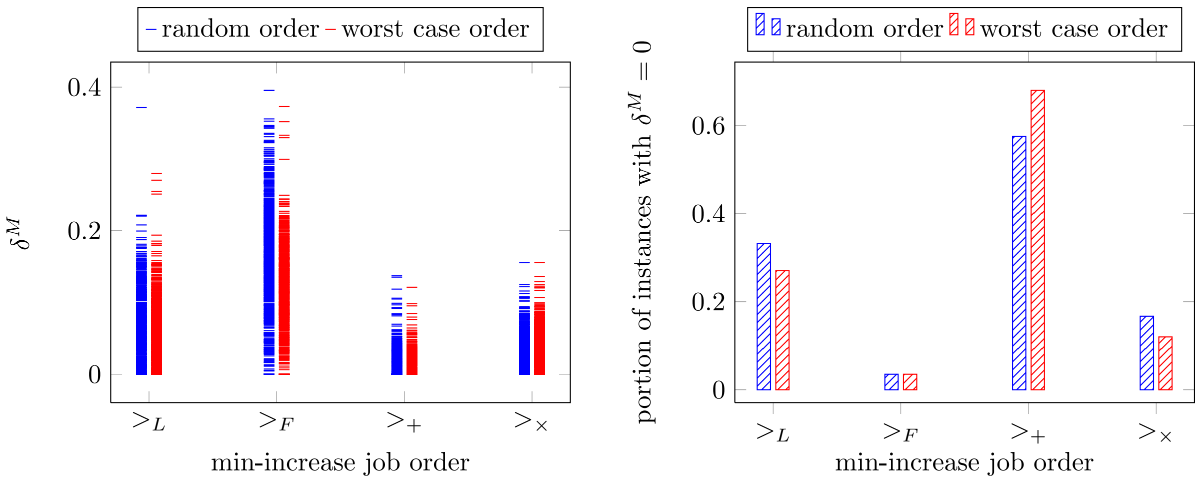


Figure 4.11.: Gap δ^M to minimal min-increase solution by min-increase job order

Here the job order $>_+$ clearly outperforms the other orders regarding both, the average of the gap value δ^M as well as the portion of instances, in which it is the best of the four orders. The next best algorithms are the ones using $>_L$ and $>_\times$, where the $>_\times$ algorithm has the better gap value on average but is also less often the one with the minimal objective value. The worst performance is achieved by using the follower's order as a list for the min-increase algorithm, since it has the worst average gap value and only yields the minimal objective value solution if $n = m$ holds and hence all min-increase schedules are optimal.

If we look at the worst case ordered instances, we observe that they are on the same gap levels as the randomly sorted instances for all but the $>_F$ order. In the case of $>_F$ the worst case instances perform even better than the randomly sorted ones. This behavior can be explained by recalling the worst case behavior of the orders in the $p \equiv 1$ case. Here only the follower's order $>_F$ and $>_\sim$ achieved the same gap levels for the random and the worst case order. The other orders performed significantly worse for the worst case. Since in the case of arbitrary processing times, each minimal objective value is achieved by a min-increase solution using one of the orders $>_L$, $>_+$ or $>_\times$, also the minimal min-increase solution is worse in the case of leader's weights ordered reversed to the follower's order $>_F$. Thus, the gap between the min-increase solution using the $>_F$ order and the minimal min-increase solution value is smaller in the worst case, because the min-increase algorithm using $>_F$ is not worst case dependent.

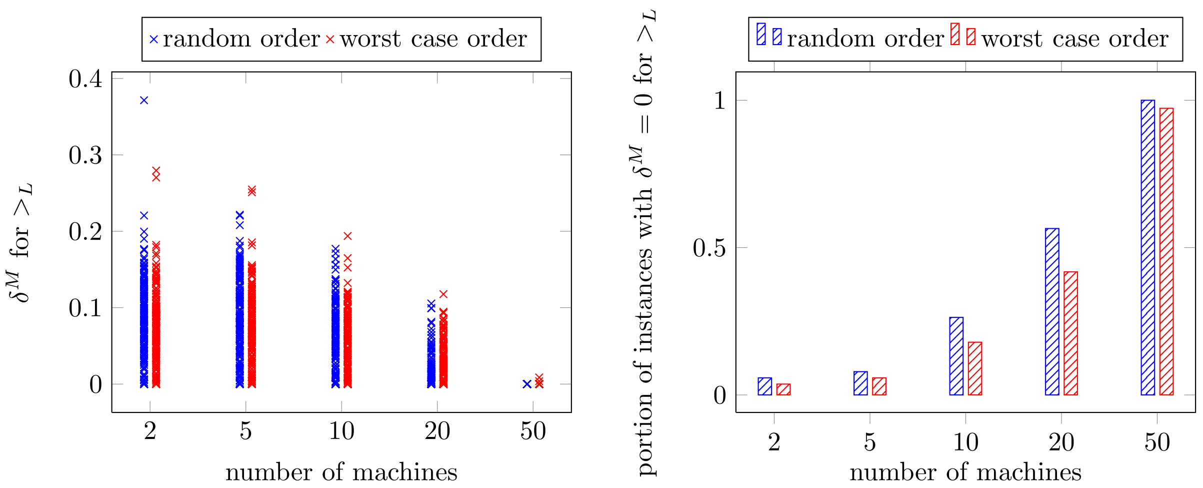


Figure 4.12.: Gap δ^M to minimal min-increase solution for $>_L$ by number of machines

As a last point of this section, we will now analyze which job order is minimal for which values of m and n. Therefore look at Figure 4.12 showing the gap value δ^M and portion of minimal instances of the min-increase algorithm using the weights of the leader $>_L$ classified by the number of machines. We observe, that for an increasing number of machines the average gap value decreases, while the portion of minimal instances increases. For the classification based on the number of jobs, presented in the pictures in Figure 4.13, the behavior is the other way around. Here the average gap value decreases and the portion of minimal instances increases for a decreasing number of jobs. Thus, we can conclude, that the leader's job order performs best on instances with small job to machine ratio.

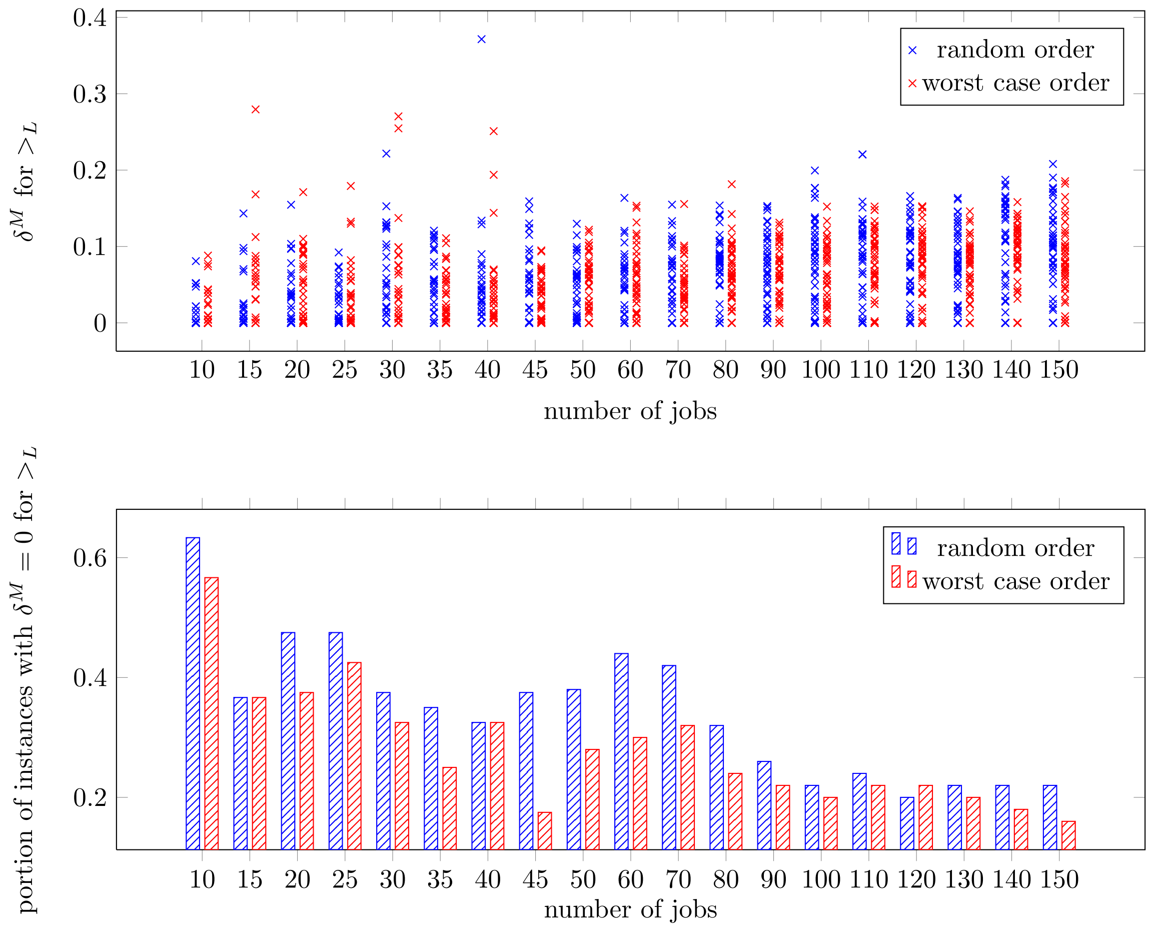


Figure 4.13.: Gap δ^M to minimal min-increase solution for $>_L$ by number of jobs

The behavior is likely caused by the fact, that the m jobs with biggest leader weight are separately scheduled on one of the m machines first in the min-increase algorithm using the job order $>_L$. Now suppose that the job to machine ratio is small, then the number of subsequent jobs to be scheduled after the first m ones is relative small and hence the heavy weighted jobs are scheduled on early positions. On the other hand, if the job to machine ratio is big, it might have been better to put the heavy jobs on a smaller subset of machines and leave the rest of the machines for jobs with less leader's weights. This explains the observed performance of the min-increase algorithm using the $>_L$ job order.

In comparison to $>_L$ we will now analyze the gap values and portions of minimal instances for the order $>_+$ by using the Figures 4.14 and 4.15, which are classified by the number of machines and the number of jobs respectively. Here it can be observed, that some values are distributed opposed to the $>_L$ order, that is, the number of minimal instances increases with increasing number of jobs and decreasing number of machines.

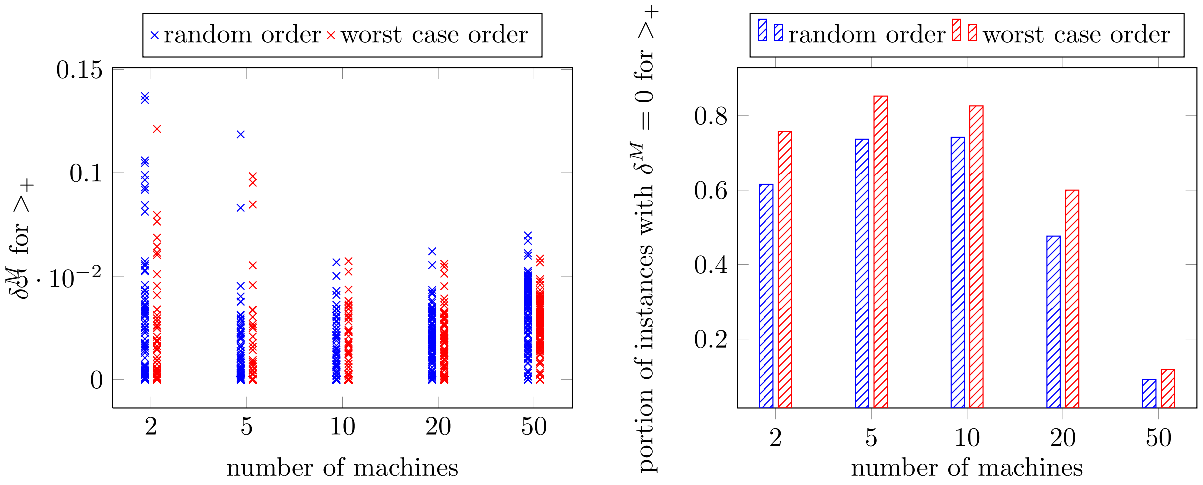


Figure 4.14.: Gap δ^M to minimal min-increase solution for $>_+$ by number of machines

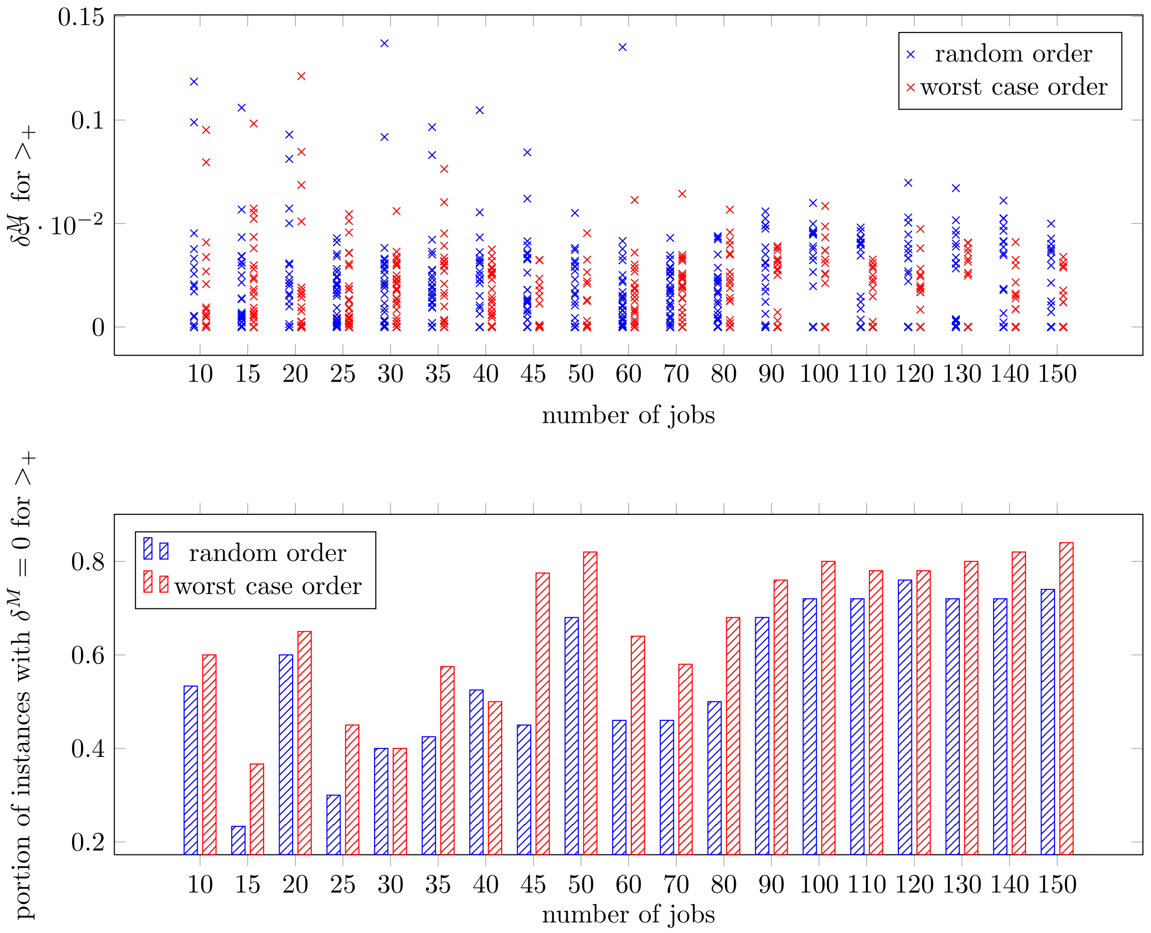


Figure 4.15.: Gap δ^M to minimal min-increase solution for $>_+$ by number of jobs

For the average gap δ^M it is not possible to make the corresponding statement, since the highest gap values exist for small machine numbers in the left picture of Figure 4.14. Furthermore the top picture of Figure 4.15 shows, that the gap does not significantly decrease, apart from outliers, for an increasing number of jobs. Thus the two job orders are not completely complementary. This is due to the fact, that some instances are also optimal for the job order $>_\times$, whose gap distribution by the number of machines and jobs can be seen in the Figures A.16, A.17 and A.18 in the appendix.

As for the other evaluated algorithms above, we did not find any dependencies for the other classification criteria, like the ratios r_1, r_{12} and r_{12}^J or the inversion ratio.

4.4. Evaluation summary

At the end of this chapter we will now summarize the results of the experimental evaluation. We first analyzed the use of the integer programming model $(\mathrm{IP})_\mathrm{p}$ for solving the special case of the BWCTP with $p \equiv 1$. Although the solving times were quite large for some instances with higher job numbers, especially for those being worst case ordered, the model is solved on all of the given instances to optimality. Thus, $(\mathrm{IP})_\mathrm{p}$ provides a good possibility to find the optimal solution for the $p \equiv 1$ special case of the BWCTP having not more than 150 jobs. For higher values of n we suppose that the solving times would further increase in the specified machine environment, which could lead to an abort of the solving process. Hence an approximation based on the optimal solution $(\mathrm{LP})_\mathrm{p}$ might be the better choice, since the observed integrality gap values of the LP relaxation are very small. Therefore the lower bound of the optimal LP solution is very near to the optimal solution of the BWCTP.

In contrast to $(\mathrm{IP})_\mathrm{p}$, the integer quadratic programming model (IQP) as stated in Section 3.2 seems to be no alternative to solve the BWCTP in practice. Even in those cases, in which the optimal solution is found, the solving time is higher than for the corresponding formulation of $(\mathrm{IP})_\mathrm{p}$. In addition it is unlikely that the optimal solution is found for values of $n \geq 20$ or $m \geq 10$, apart from the trivial case of $n = m$. Thus, the solving of (IQP) is not a suitable approach to find provable good solutions for the BWCTP.

If we are looking for an approach that yields reasonable solutions in little time, we could use the min-increase scheduling algorithm. The analysis on $p \equiv 1$ shows, that the best job list is obtained by $>_\sim$ in this case, processing the jobs in ascending average LP position order. Here the smallest average gap values are achieved and the algorithm is only little affected by the worst case resorting of the leader's weights. However the execution time of the min-increase algorithm would be increased enormously, if we add the needed solving time of the corresponding LP relaxation. Furthermore the LP solution is only available for unit weight processing times. Hence it might be better to choose the $>_+$ job order for the min-increase scheduling algorithm, which performs best in the case of arbitrary processing times. This algorithm has the lowest average normalized difference to the minimal min-increase solution and it yields the minimal solution in most of the analyzed cases. Only for instances with a small job to machine ratio, the job order $>_L$ mainly leads to better results. Again, all the min-increase scheduling algorithms are very fast, but the obtained solutions can only be compared to other min-increase solutions, since no optimal solution is known in the case of arbitrary processing times. Therefore no statement on the overall quality of the min-increase results can be made here.

Throughout all evaluated approaches, the results did not admit any relation to the ratios r_1, r_{12} and r_{12}^J, which describe the difference in the weight functions of the leader and the follower. The same holds for the inversion ratio, which only depends on the job orders $>_L$ and $>_F$ following from the weight functions and processing times. Even for algorithms admitting a dependency on the worst case job order, no clear correlation could be found for the inversion ratio. As already explained in the results part for $p \equiv 1$ of the min-increase scheduling algorithm this could be due to the fact, that the inversion ratios are not distributed equally for the given instances. Furthermore a more differentiated analysis of the inversion ratio and the weight dependent ratios could possibly lead to better results. It would, for example, be interesting to compare the gap against the inversion ratio only on those instances having a similar number of jobs or job to machine ratio, which is, however, beyond the scope of this thesis.

Summary

In this thesis we studied the Bilevel Weighted Completion Time Problem, or BWCTP in short, and gave some structural results as well as several approaches to find corresponding approximation results for the problem. We first showed in the beginning Chapter 2 that the special cases of the BWCTP with $w^1 \equiv 1$ and $p \equiv 1$ are equivalent. Afterwards we presented as a first approximation result an FPTAS for the BWCTP with a fixed number m of machines in Section 2.2. Therefore the following part of the thesis concentrated on the case of m being part of the input. In Section 2.3 we studied two different list based algorithms, the list scheduling and the min-increase algorithm, which are proved constant factor approximation algorithms for the one-level base problem of the BWCTP, the Weighted Completion Time Problem or WCTP. We proved that the list scheduling algorithm in general cannot be a constant factor approximation algorithm for the BWCTP, even if the processing times have unit length. Furthermore we showed for all four considered orders of the jobs, that the min-increase algorithm processing the jobs in the given order does not admit a constant worst case approximation factor as well. In the last Section 2.4 of this chapter we presented different approximation algorithms based on the lower bound of the optimal solution of the one-level WCTP. We provided two approximations algorithms with approximation ratios depending on the gap between the weights of the leader and the weights of the follower, respectively the gap between the leader's weights and the processing times of the jobs. Additionally we gave an approximation algorithm with worst case ratio of $\beta(1+\epsilon)^2$, where β denotes the gap between the optimal solution values of the WCTP and the BWCTP. We concluded the section by proving β to be unbounded even for the $p \equiv 1$ case of the BWCTP and showed that we can restrict β to $m \sqrt[m]{r_1}$, where r_1 denotes the biggest factor between the leader's weights of the jobs.

In the next Chapter 3 we presented the integer programming formulation $(\mathrm{IP})_\mathrm{p}$ for the BWCTP with $p \equiv 1$ and the integer quadratic programming formulation (IQP) for the general BWCTP. We first studied $(\mathrm{IP})_\mathrm{p}$ in Section 3.1 and corrected a wrong proof of Kis and Kovács [KisKov2012], that states the existence of an optimal integer solution for the linear relaxation $(\mathrm{LP})_\mathrm{p}$ of $(\mathrm{IP})_\mathrm{p}$ for some special instances. Afterwards we provided different approaches constructing an approximative solution for the BWCTP with $p \equiv 1$

from the optimal solution of $(\mathrm{LP})_\mathrm{p}$. In one approach we used the optimal solution of the relaxation to define another job order for the min-increase algorithm, which again was proved to admit no constant approximation ratio. Another approach resulted in a deadline problem for which an polynomial time algorithm was given. However, it remained open if it can be proved, that the received solution is an approximative solution for the BWCTP with $p \equiv 1$. In the second Section 3.2 of the chapter, we showed that the non-integer relaxation of (IQP) admits an integer optimal solution. Additionally we presented an approach of Skutella [Sku2001], who gave an 2-approximation of the $R||\sum_j w_j C_j$ based on a similar quadratic programming formulation, by convexifying the objective function. We showed that this approach cannot be adopted to (IQP), since the corresponding objective function yields no convex counterpart as in [Sku2001].

The last Chapter 4 gave an empirical evaluation of the min-increase algorithm as well as the two presented models $(\mathrm{IP})_\mathrm{p}$ and (IQP) on a set of randomly generated instances. This led to the conjecture of another case of the BWCTP with $p \equiv 1$, for with $(\mathrm{LP})_\mathrm{p}$ admits an optimal integer solution, namely the case of $m = 2$.

Apart from this conjecture, there are still other open questions regarding the BWCTP, that motivate future research. Firstly the complexity of the $p \equiv 1$ respectively $w^1 \equiv 1$ special case of the BWCTP is still unknown. Furthermore we do not known if there is a constant factor approximation algorithm for the general BWCTP or if there cannot be one. At last the existence of an approximation based on the min-increase algorithm remained an open question.

Bibliography

[Ach2009] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1): 1–41, July 2009. `http://mpc.zib.de/index.php/MPC/article/view/4`, last checked on July 11th, 2014.

[Bru2007] Peter Brucker. *Scheduling Algorithms.* Springer, 2007.

[BruLenK1977] P. Brucker, J.K. Lenstra, and A.H.G. Rinnooy Kan. Complexity of machine scheduling problems. In B.H. Korte P.L. Hammer, E.L. Johnson and G.L. Nemhauser, editors, *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 343 – 362. Elsevier, 1977.

[CapCarLodWoe2014] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J Woeginger. A study on the computational complexity of the bilevel knapsack problem. *SIAM Journal on Optimization*, 24(2): 823–838, 2014.

[ChuMur1981] Sung J Chung and Katta G Murty. Polynomially bounded ellipsoid algorithms for convex quadratic programming. *Nonlinear programming*, pages 439–485, 1981.

[EasEveIsa1964] W. L. Eastman, S. Even, and I. M. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management science*, 11(2): 268–279, 1964.

[GarJoh1979] M. R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[GasKli2009] Elisabeth Gassner and Bettina Klinz. The computational complexity of bilevel assignment problems. *4OR*, 7(4): 379–394, 2009.

[Gol2004] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57. Elsevier, 2004.

[GraLawLenK1979] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling:

a survey. In E.L. Johnson P.L. Hammer and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[Gro2010] Martin Grötschel. Vorlesungsskript lineare und ganzzahlige programmierung. Technical report, Institut für Mathematik, Technische Universität Berlin, August 2010.

[Joh2011] Berit Johannes. *New Classes of Complete Problems for the Second Level of the Polynomial Hierarchy.* PhD thesis, Technische Universität, 2011.

[Kar1972] Richard M. Karp. Reducibility among combinatorial problems. 1972.

[KarWan1996] John K Karlof and Wei Wang. Bilevel programming applied to the flow shop scheduling problem. *Computers & operations research*, 23(5): 443–451, 1996.

[KawKya1986] Tsuyoshi Kawaguchi and Seiki Kyan. Worst case bound of an lrf schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4): 1119–1129, 1986.

[KhaKozTar1980] Leonid G Khachiyan, Mikhail K Kozlov, and Sergei P Tarasov. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5): 223–228, 1980.

[KisKov2011] Tamás Kis and András Kovács. Constraint programming approach to a bilevel scheduling problem. *Constraints*, 16(3): 317–340, 2011.

[KisKov2012] Tamás Kis and András Kovács. On bilevel machine scheduling problems. *OR spectrum*, 34(1): 43–68, 2012.

[LawLenKanShm1993] Eugene L Lawler, Jan Karel Lenstra, Alexander HG Rinnooy Kan, and David B Shmoys. Sequencing and scheduling: Algorithms and complexitylaw. *Handbooks in operations research and management science*, 4: 445–522, 1993.

[LukRosSo2008] Zrinka Lukač, Višnja Vojvodić Rosenzweig, and Kristina Šorić. Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research*, 187(3): 1504–1512, 2008.

[Moe2010] Rolf Möhring. Vorlesungsskript graphen- und netzwerkalgorithmen (adm i). Technical report, Institut für Mathematik, Technische Universität Berlin, September 2010.

[Sah1974] Sartaj Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4): 262–279, 1974.

[SchWoe2001] Petra Schuurman and Gerhard J Woeginger. Approximation schemes-a tutorial. *Lectures on scheduling*, 2001.

[ShmWil2010] David B. Shmoys and David P. Williamson. *The Design of Approximation Algorithms*. Cambridge University Press, 2010.

[Sku2001] Martin Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM (JACM)*, 48(2): 206–242, 2001.

[SkuWoe2000] Martin Skutella and Gerhard J Woeginger. A ptas for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1): 63–75, 2000.

[Smi1956] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2): 59–66, 1956.

A. Additional graphics

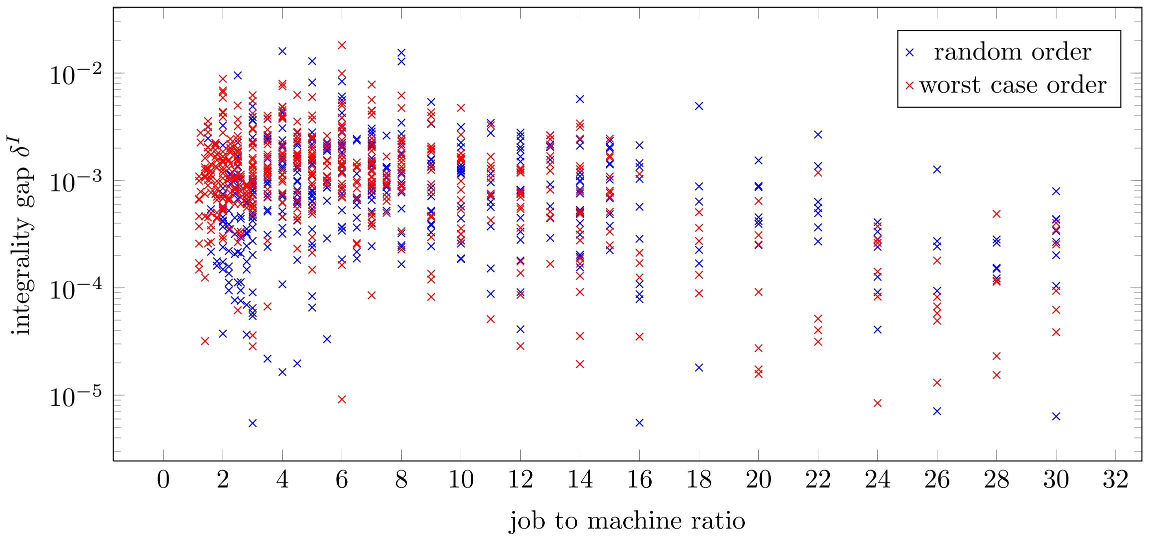


Figure A.1.: Integrality gap δ^I between $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ by job to machine ratio

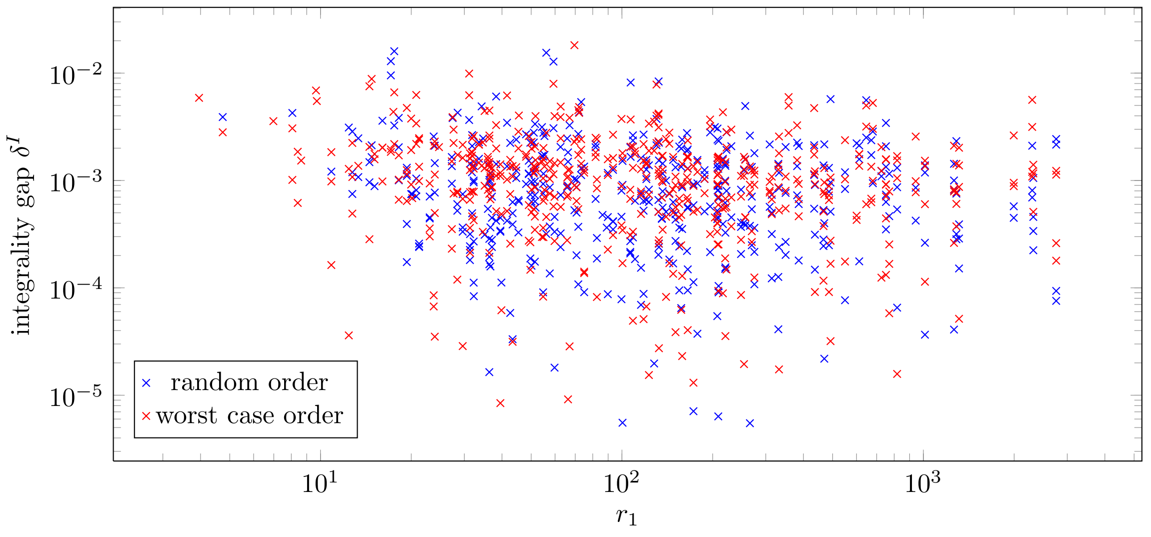


Figure A.2.: Integrality gap δ^I between $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ by ratio r_1

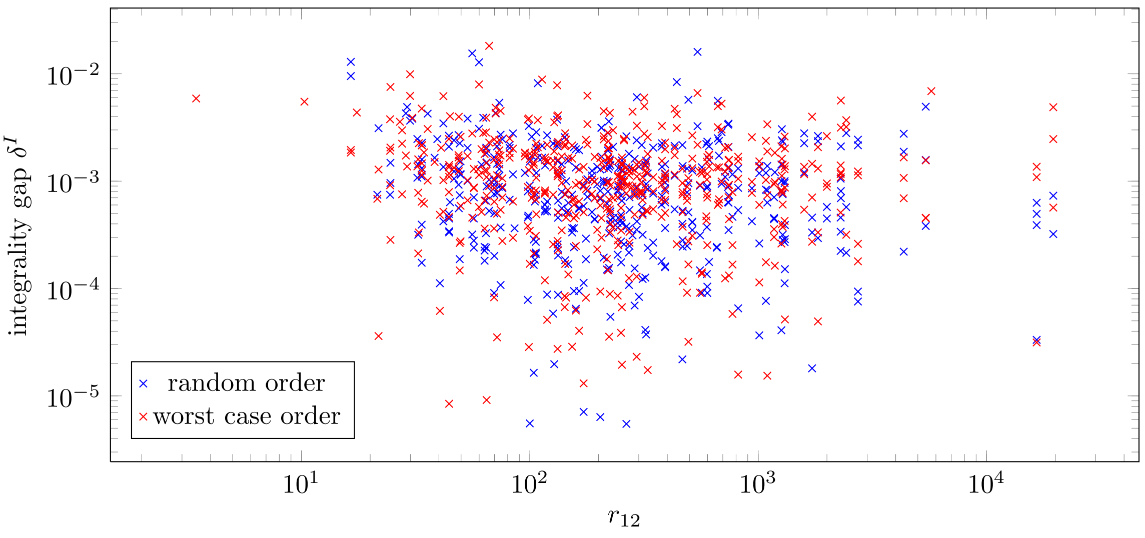


Figure A.3.: Integrality gap δ^I between $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ by ratio r_{12}

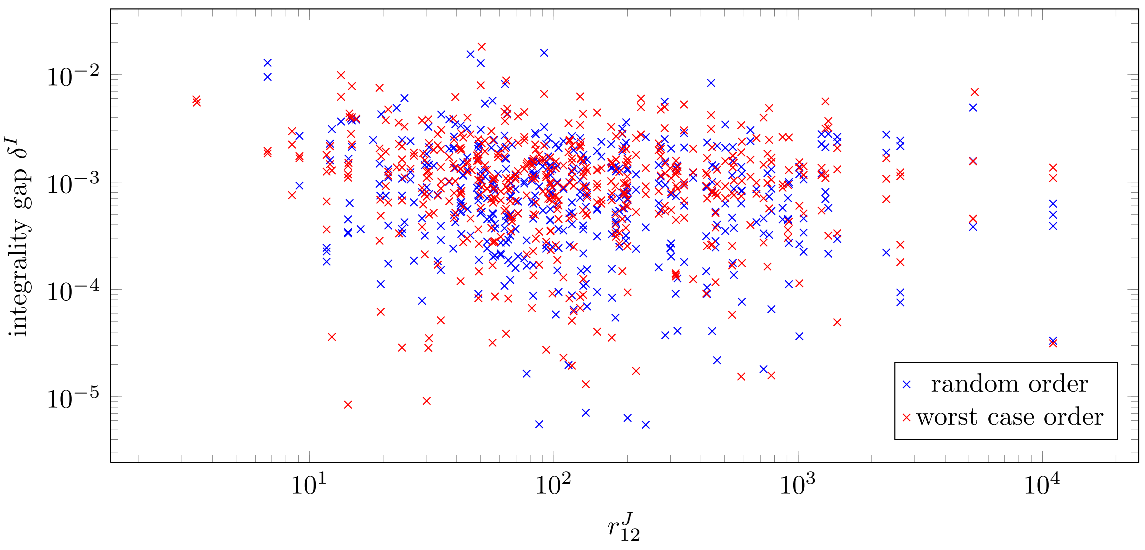


Figure A.4.: Integrality gap δ^I between $(\mathrm{IP})_\mathrm{p}$ and $(\mathrm{LP})_\mathrm{p}$ by ratio r_{12}^J

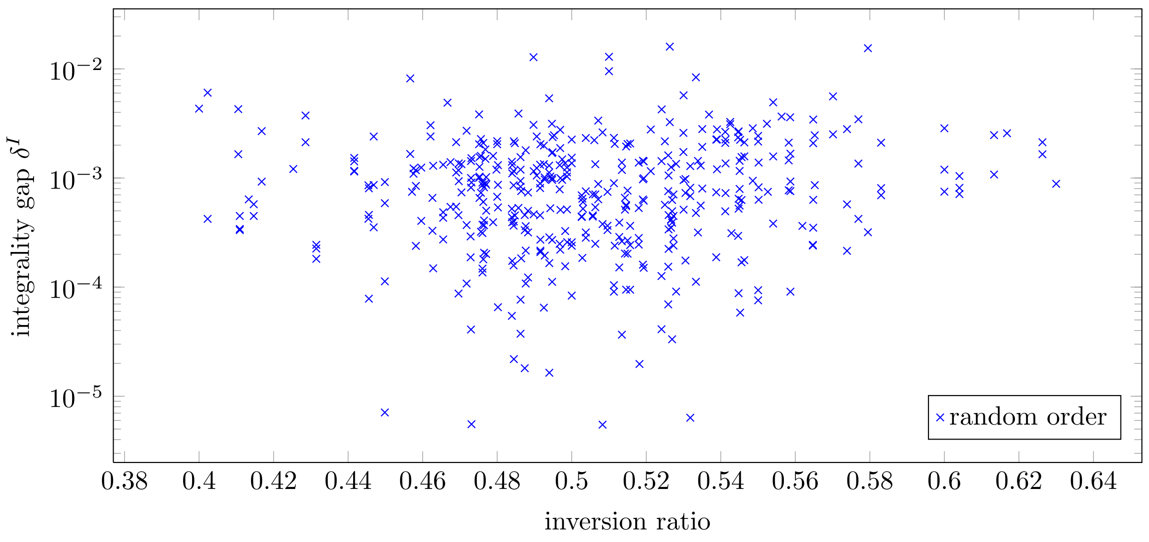


Figure A.5.: Integrality gap δ^I between $(\text{IP})_\text{p}$ and $(\text{LP})_\text{p}$ by inversion ratio

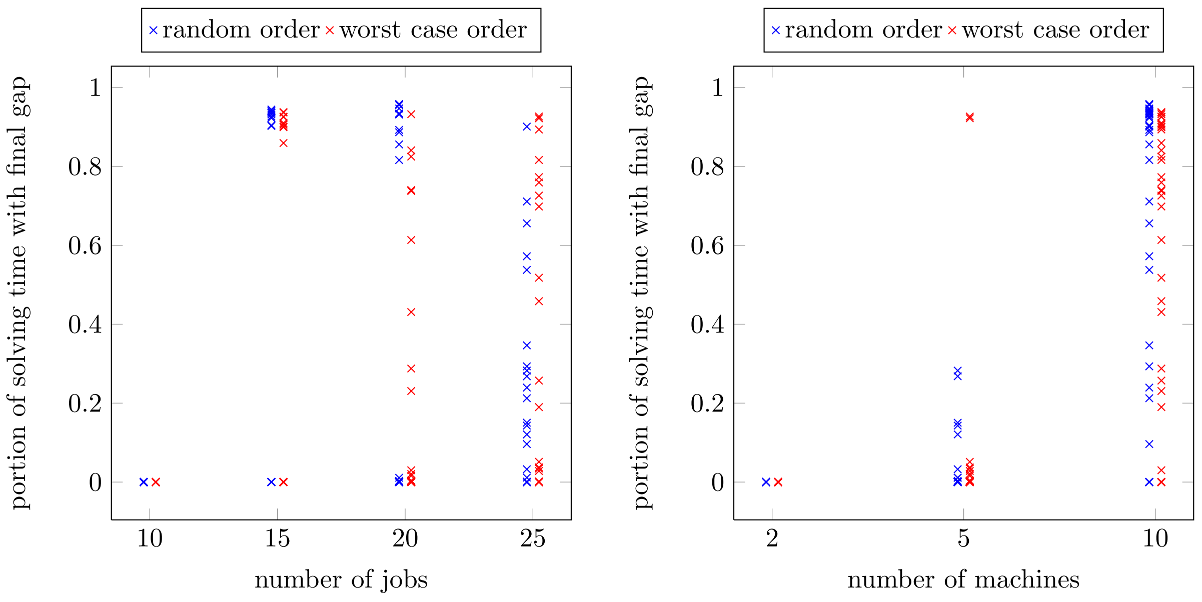


Figure A.6.: Portion of (IQP) solving time with final gap by n and m

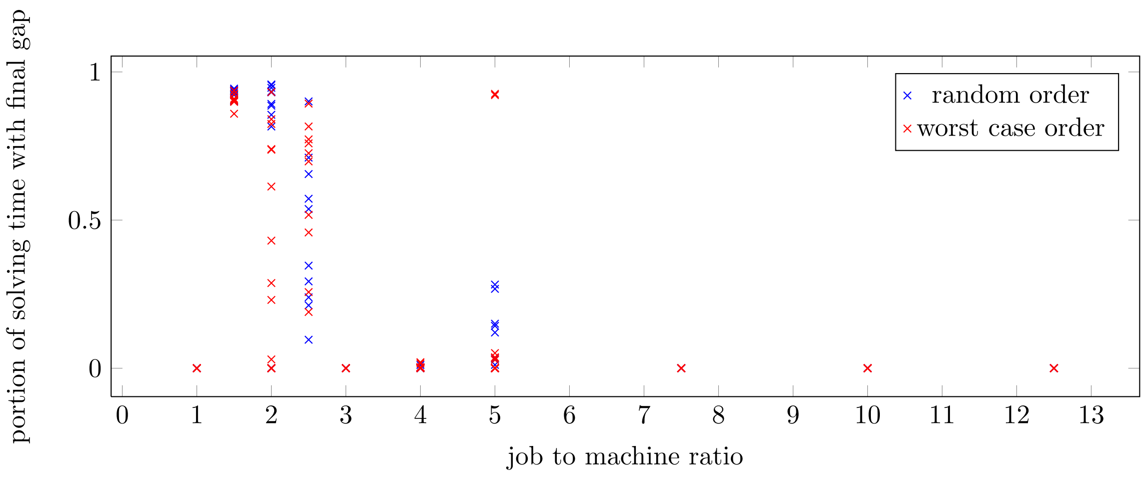


Figure A.7.: Portion of (IQP) solving time with final gap by job to machine ratio

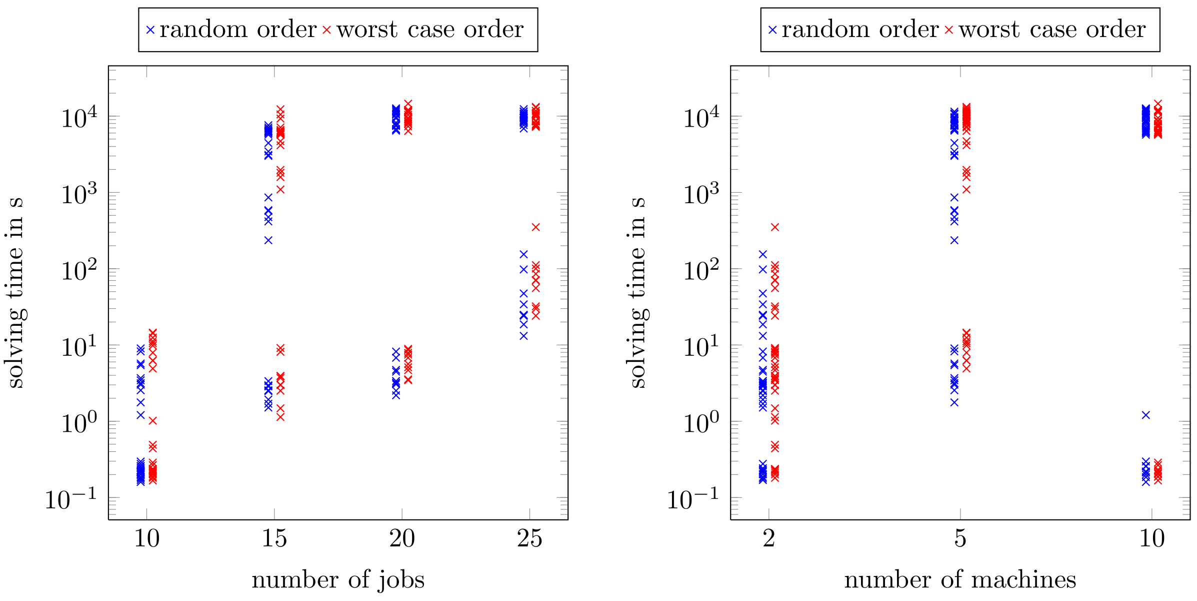


Figure A.8.: (IQP) solving time by n and m

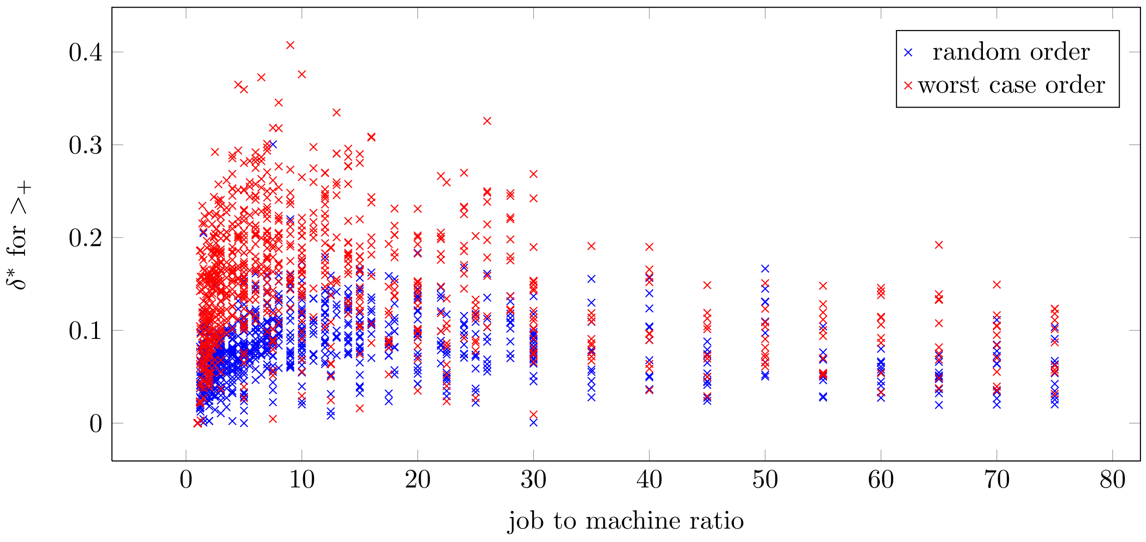


Figure A.9.: Min-increase gap δ^* to the optimal solution for $>_+$ by job to machine ratio

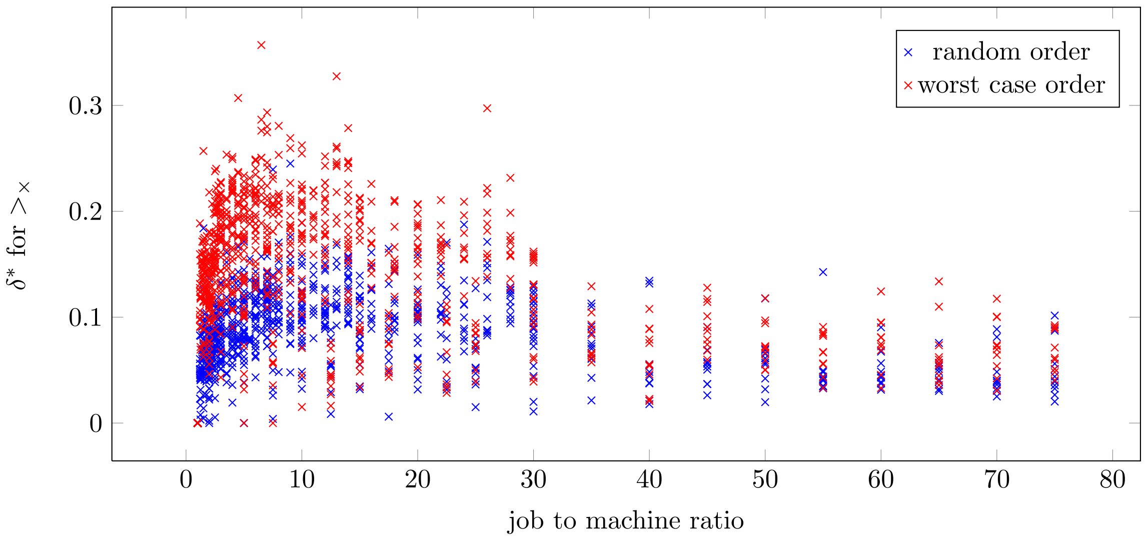


Figure A.10.: Min-increase gap δ^* to the optimal solution for $>_\times$ by job to machine ratio

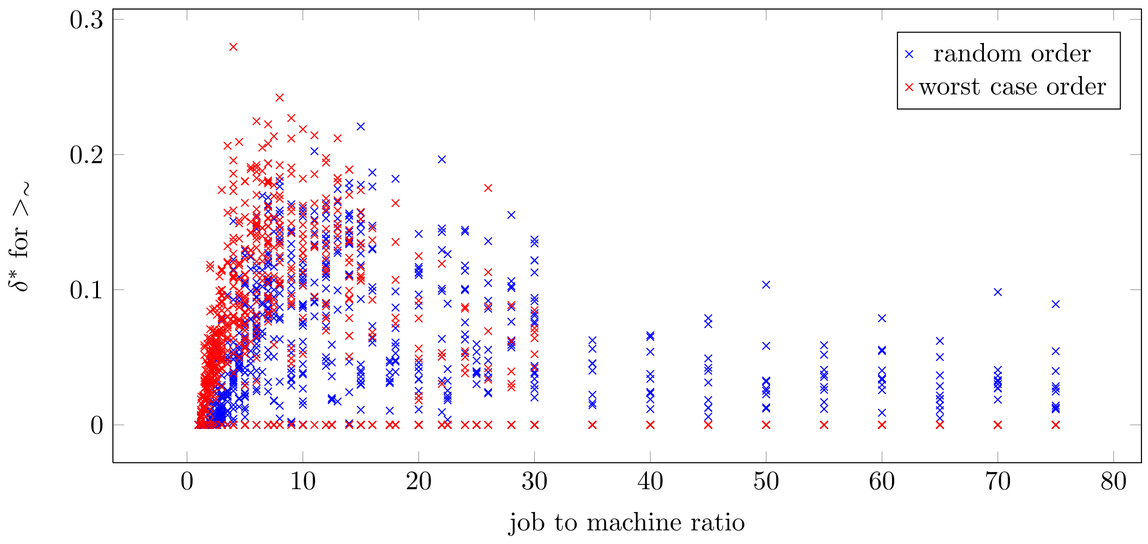


Figure A.11.: Min-increase gap δ^* to the optimal solution for $>_\sim$ by job to machine ratio

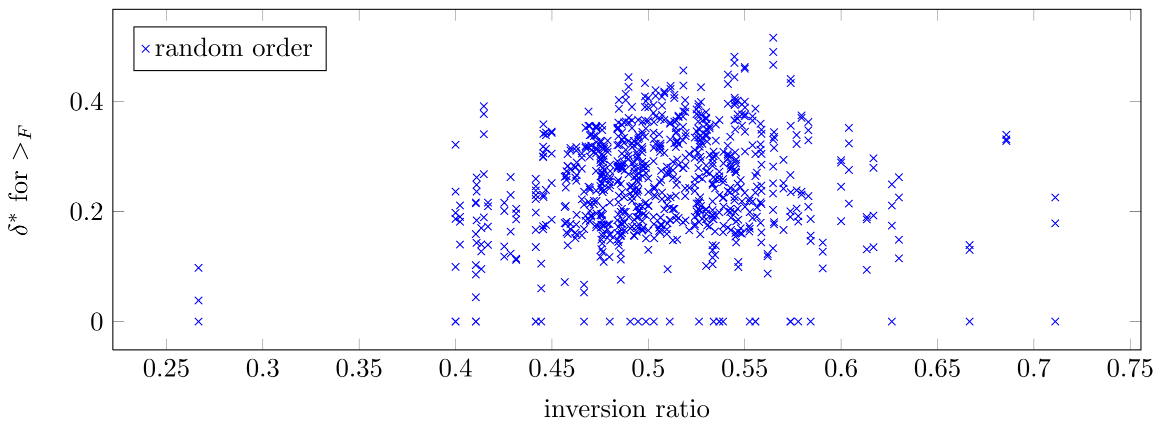


Figure A.12.: Min-increase gap δ^* to the optimal solution for $>_F$ by inversion ratio

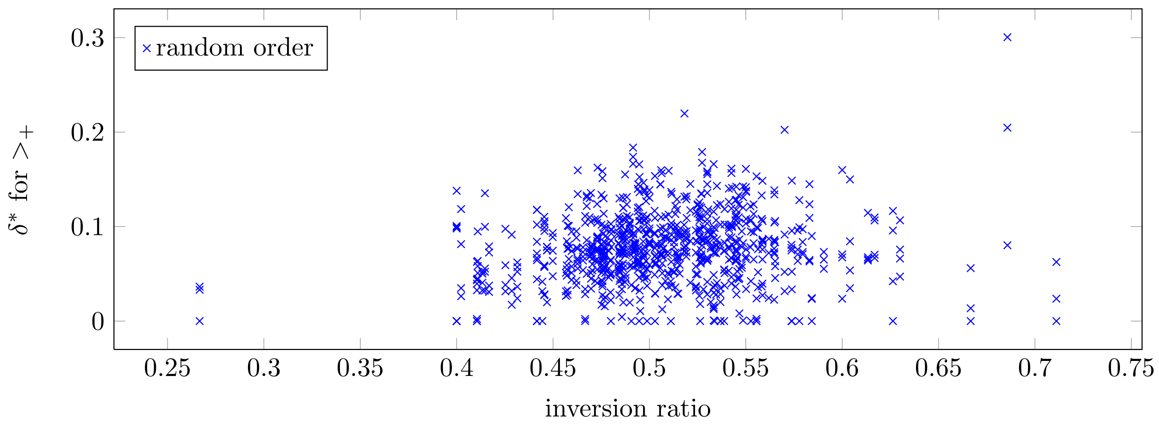


Figure A.13.: Min-increase gap δ^* to the optimal solution for $>_+$ by inversion ratio

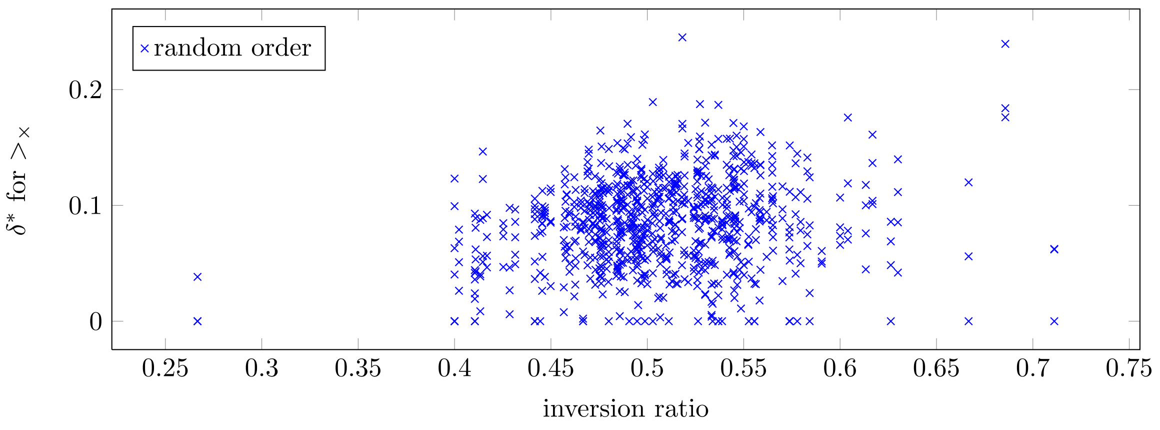


Figure A.14.: Min-increase gap δ^* to the optimal solution $>_\times$ by inversion ratio

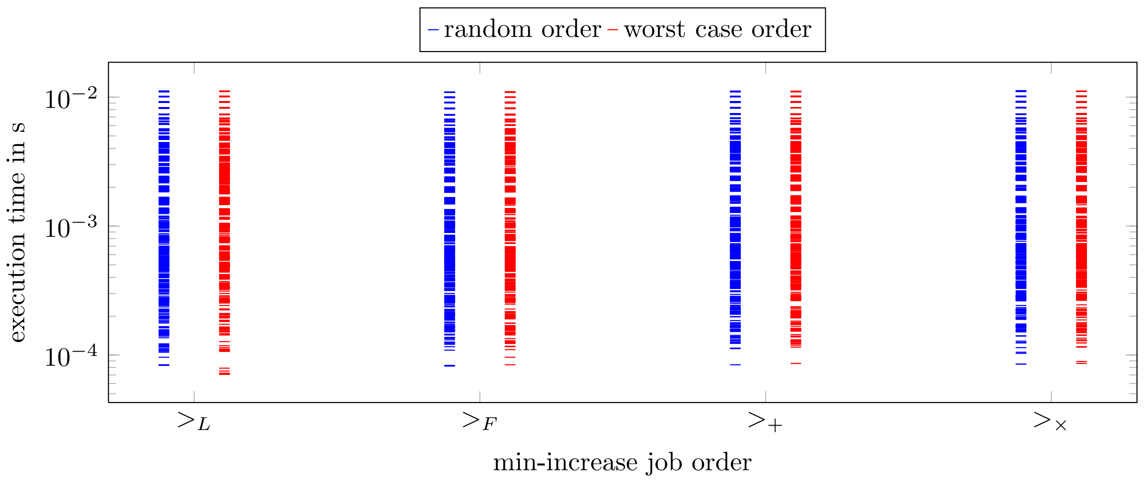


Figure A.15.: Min-increase execution time by min-increase job order on arbitrary p

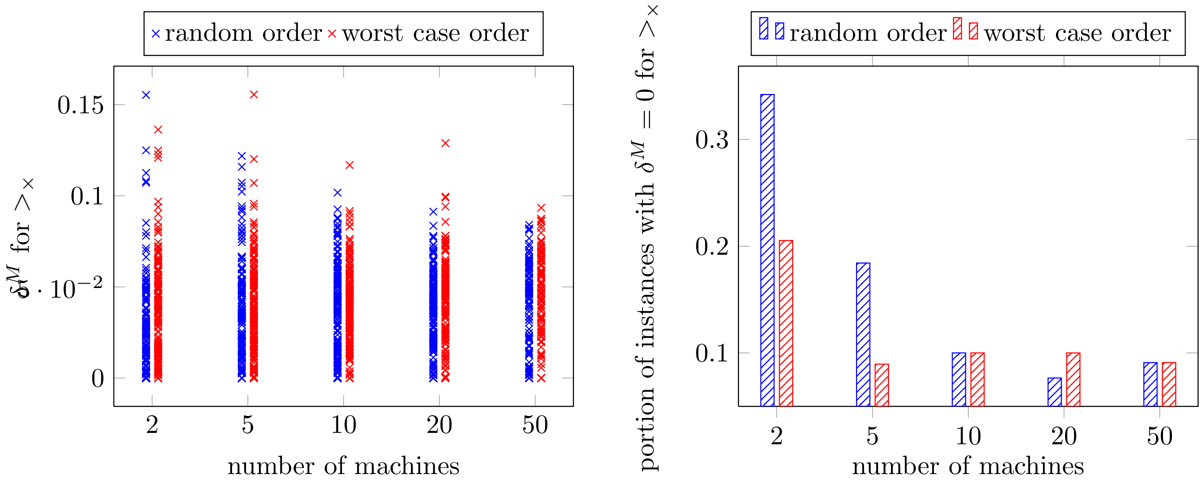


Figure A.16.: Min-increase gap δ^M on arbitrary p for $>_\times$ by number of machines

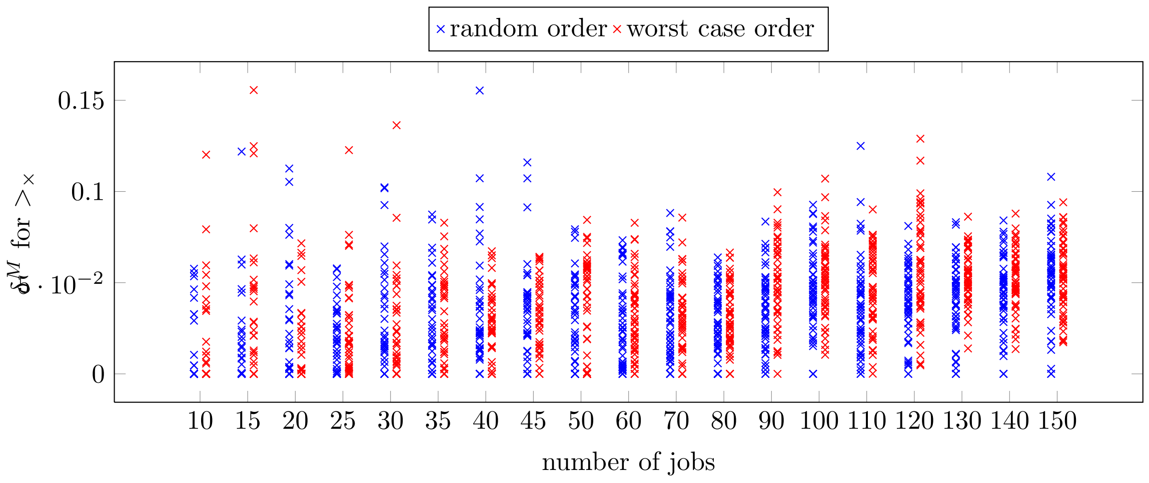


Figure A.17.: Min-increase gap δ^M on arbitrary p for $>_\times$ by number of jobs

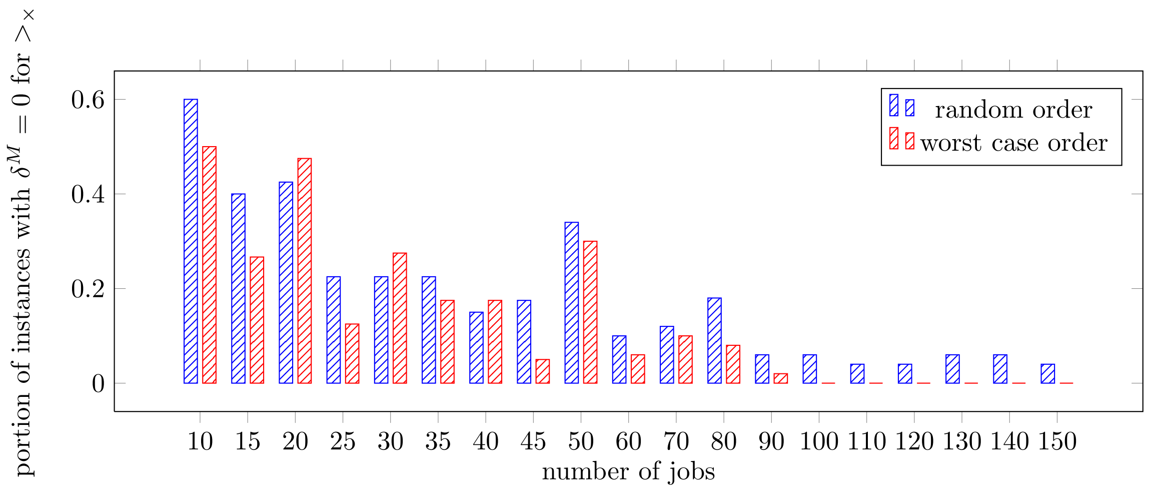


Figure A.18.: Ratio of minimal instances on arbitrary p for $>_\times$ by number of jobs