A Parallel Machine Schedule Updating Game with Compensations and Clients Averse to Uncertain Loss[†]

Mikhail Y. Kovalyov[a], Dominik Kress[b,*], Sebastian Meiswinkel[b], Erwin Pesch[b,c]

[a] *United Institute of Informatics Problems, National Academy of Sciences of Belarus, 220012 Minsk Belarus*
[b] *Department of Management Information Science, University of Siegen, Kohlbettstr. 15, 57068 Siegen, Germany*
[c] *Center for Advanced Studies in Management, HHL Leipzig, Jahnallee 59, 04109 Leipzig, Germany*

Abstract

There is a finite number of non-cooperating clients, who are averse to uncertain loss and compete for execution of their jobs not later than by their respective due dates in a parallel service environment. For each client, a due date violation implies a cost. In order to address the minimization of the total scheduling cost of all clients as a social criterion, a game mechanism is suggested. It is designed such that no client has an incentive to claim a false due date or cost. The game mechanism allows the clients to move their jobs to complete earlier in a given schedule. However, they must compensate costs of those clients whose jobs miss their due dates because of these moves. Algorithmic aspects are analyzed. Furthermore, a polynomial time algorithm that determines an equilibrium of the considered game is suggested and embedded into the game mechanism. Computational tests analyze the performance and practical suitability of the resulting game mechanism.

Keywords: Scheduling, Logistics, Game theory, Algorithmic mechanism design

1. Introduction

When designing and analyzing algorithms for scheduling problems, we often assume that a central decision maker - who is in control of some algorithm - has access to all relevant data that defines a problem instance. However, there exist many real world applications where this assumption does not hold. We can, for example, think of due dates, processing times, or other relevant parameters of a scheduling domain, which are private information of the jobs, machines, or their respective owners. When acting selfishly, these owners (also referred to as players, agents, or clients) may try to influence the solution determined by the scheduling algorithm by submitting false data. In some cases, however, the decision maker can extract the true information by designing an appropriate algorithm that sets the right incentives for the owners. The design of such algorithms is subject of a field of research that is usually referred to as *algorithmic mechanism design* [1, 2].

1.1. Related Literature

Algorithmic mechanism design is an intersection of multiple disciplines, namely (algorithmic aspects of) computer science, game theory and economic theory, and thus belongs

*Corresponding author
Email addresses: kovalyov_my@newman.bas-net.by (Mikhail Y. Kovalyov), dominik.kress@uni-siegen.de (Dominik Kress), sebastian.meiswinkel@uni-siegen.de (Sebastian Meiswinkel), erwin.pesch@uni-siegen.de (Erwin Pesch)

[†] This is an Accepted Manuscript of an article published by Elsevier in **Computers & Operations Research**, available online: https://doi.org/10.1016/j.cor.2018.11.003

to the broader field of *algorithmic game theory* (an excellent introduction and overview is given in [2]). Most research articles in this field focus on auction contexts (see, for instance, [3]). Recently, however, there has been a growing interest in taking a game theoretic perspective on machine scheduling settings, which has resulted in a fairly large amount of research articles. As stated in [4], there are two main streams of related publications. The first stream assumes that solely the machines possess private information (*machine agents*). These articles follow the seminal work of [5, 6]. Similarly, the second group of publications assumes that only the jobs are selfish agents (*job agents*), which is the perspective taken in this paper. Prominent examples of the latter stream are [7, 8, 9, 10]. We abstain from a detailed overview of the literature, because up-to-date overviews and reviews of algorithmic mechanism design in a scheduling context are given in [4, 11, 12]. Furthermore, for the sake of brevity, we assume the reader to be familiar with the basic theory and terminology of machine scheduling problems and the main concepts of game theory, and refer to [13, 14] and [15, 16, 17] for comprehensive introductions to these fields of research.

1.2. Motivation and Outline of the Problem

In this paper, we consider a scheduling problem in which multiple clients, each owning a single job, compete for the execution of their jobs not later than by their respective due dates on parallel processing units (machines) of a service provider (also referred to as operator). A processing time, a due date, and a weight representing the cost of missing the due date, is associated with each job. Any job can be processed by any machine. Each machine processes jobs one after another, non-overlappingly and non-preemptively, starting from time zero, without idle time. The clients are assumed to be non-cooperative; that is, they cannot form coalitions to exchange information and generate a group decision. Competition of the clients is regulated by a game mechanism, which receives information claimed by the service provider and the clients and suggests rules for generating a schedule of processing the jobs. The processing times of the jobs are truly claimed by the service provider, because the revenue from processing a job is fixed. Weights and due dates, however, are claimed by their respective clients and can differ from the true values.

As the service provider aims to find a schedule that respects the interests of all players, we consider minimizing the total scheduling cost of all clients as a social criterion, which the service provider would like to address. The suggested game mechanism is such that if the clients claim true weights and due dates, then they are certain that their true total cost (loss) is implied by the social criterion. If they lie, then the outcome becomes uncertain, both with respect to the value of the social criterion and the loss of particular clients, including the one who is lying. We assume that the clients are *fully averse to uncertain loss*. Due to this fear of loss, they claim true weights and due dates.

Our study is motivated by planning operations of a railway container unloading terminal (see, e.g., [18]). There are multiple parallel railway tracks and each track is served by an associated single hoist crane (corresponding to one of the machines). Each crane can process at most one train at a time. For a given planning period, the terminal financial manager negotiates service contracts with the shipping agents. Each contract specifies the unloading of a train (i.e., a job of the scheduling problem). For a shipping agent, the contract is associated with the train unloading due date and a cost, which is paid if and only if the train misses the due date. For the terminal owner, this contract is associated with the train unloading time and a fixed revenue. During the negotiation process, the manager would like to work out a schedule for unloading the trains by the cranes which is satisfactory to the interested shipping agents, and then to specify it in the corresponding contracts.

1.3. Contribution and Overview of the Paper

The contribution of this paper is twofold. First, the problem studied in this paper is a generalization of the problem studied earlier by us [19], in which there is only a single machine. This necessitates a variation of the mechanism suggested in [19] as well as a reinvestigation of the corresponding theoretical properties. Thus, our paper contributes both to the scheduling literature and the literature on algorithmic mechanism design, where only very few papers consider agents which are fully averse to uncertain loss or risk (see [4]). Second, in contrast to [19], we will present computational results in this paper. These results demonstrate that the suggested game mechanism enables a service provider to provide high quality solutions that feature an appropriate level of communication between the service provider and the clients.

The remainder of this paper is structured as follows. In Section 2, we introduce the required notation and define the problem under consideration in more detail. The game mechanism is presented in Section 3. Truthfulness of the clients is discussed in Section 4. An algorithmic analysis of the problem is provided in Section 5. An $O(n^2)$ algorithm to find a game equilibrium is presented and embedded into the game mechanism in Section 6, where we additionally present the aforementioned computational results. The paper concludes with a short summary of the results and suggestions for future research in Section 7.

2. Notation and Detailed Problem Description

We will denote the set of clients - or their respective *jobs* - by $J = \{1, \ldots, n\}$. Each job $j \in J$ is associated with a processing time $p_j \in \mathbb{N}$, a due date $d_j \in \mathbb{N}$, and a weight $w_j \in \mathbb{N}$ representing the cost of missing the due date. The set of parallel *machines* of the service provider is referred to by $M = \{1, \ldots, m\}$, where $m \leq n - 1$.

As mentioned above, competition of the clients is regulated by a game mechanism. The processing times p_j are truly claimed by the service provider for all $j \in J$, because the revenue from processing a job is fixed. The parameters w_j and d_j, however, are claimed by their respective clients $j \in J$, and can differ from the true values, denoted by w_j^{true} and d_j^{true}. Because of business confidentiality, the values p_j, $j \in J$, and, possibly, the number of clients n and the number of machines m are not revealed to the clients. Moreover, the claimed values w_j and d_j of client j are not revealed to the other clients, and the true values w_j^{true} and d_j^{true} of client j are neither revealed to the service provider nor to the other clients.

We assume that the clients are fully averse to uncertain (unpredictable, with unknown probability distribution) loss, so that they do not lie about their due dates and weights if lying induces an uncertain loss situation while claiming true values induces no unexpected loss. The *loss aversion* phenomenon is observed in many economical situations. According to it, people prefer avoiding losses much more than acquiring gains [20, 21, 22] and they work harder to avoid losses [23, 24]. People also prefer avoiding unmeasurable uncertainty by selecting actions with known probabilistic outcomes rather than those with unknown outcomes. This phenomenon is called *ambiguity aversion* in the scientific literature [25, 26] and has been confirmed both in experimental market settings [27] and for business owners and managers [28]. Both these phenomena stimulated us to impose the assumption of full aversion to uncertain loss, which makes the model and the solution of the schedule updating game problem understandable and easy to implement.

A *schedule* for processing the jobs, which is satisfactory to all clients, has to be determined. The clients would like to have an influence on the completion times of their jobs. We therefore suggest that the game mechanism proposes an initial schedule and that the clients can update this schedule by following the rules described in this paper. A schedule is represented by the job sequences on the machines. Given a schedule, let C_j denote the

completion time of job $j \in J$. Job j is said to be *on-time* with respect to a given due date d_j, if it completes before or at this due date. If job j is not on-time, it is said to be *late*. A job $j \in J$ that is on-time with respect to its *true* due date incurs no cost to client j. However, if job $j \in J$ is late with respect to its true due date, then it incurs cost w_j^{true}.

Our *game mechanism* determines an initial schedule and rules of moving a late job to complete earlier. The financial regulation of the game works as follows. Client j, whose job is late in an initial schedule, is presented a set of strategies to move her job to an earlier position in the new schedule. For a specific move of job j, let J_j denote the set of jobs that are on-time in the initial schedule and late in the new schedule. If this move is realized, then client j will pay an amount of w_i to each client $i \in J_j$. The move is presented to the client only if the cost reduction of client j is greater than the corresponding compensation paid to the other clients. After a certain number of moving operations, a final schedule is obtained and the jobs are processed according to this schedule. It is clear that the total compensation paid by all clients is equal to the total compensation received by all clients.

Define a cost indicator function $U(d_j, C_j)$, such that $U(d_j, C_j) = 0$ if $C_j \leq d_j$ and $U(d_j, C_j) = 1$ if $C_j > d_j$, $j \in J$. Client $j \in J$ aims at minimizing her *loss function*

$$F_j := w_j^{true} U(d_j^{true}, C_j) + V_j^- - V_j^+,$$

where $w_j^{true} U(d_j^{true}, C_j)$ is her scheduling cost, V_j^- is the total compensation paid by her and V_j^+ is the total compensation paid to her. When there is no ambiguity about the due dates and the schedule, we will simplify the notation by writing U_j instead of $U(d_j, C_j)$ in the following. According to the game rules described above, $\sum_{j=1}^{n}(V_j^- - V_j^+) = 0$.

We consider minimizing the total scheduling cost of all clients, $\sum_{j=1}^{n} F_j = \sum_{j=1}^{n} w_j U_j$, as a *social criterion*, which the service provider would like to address. The deterministic problem to find a schedule that minimizes $\sum_{j=1}^{n} w_j U_j$ is denoted by $P||\sum w_j U_j$ in the scheduling literature [29]. It is known to be NP-hard in the strong sense [30]. However, it admits computation of an exact solution in $O(n^3)$ time for the special case with identical job processing times, $p_j = p$, for all jobs j, and due dates proportional to p, i.e. $d_j = k_j p$, where k_j is an integer number, by a reduction to an assignment problem [31]. This problem is denoted by $P|p_j = 1|\sum w_j U_j$ in the scheduling literature. The special case with a constant number of machines m, denoted by $Pm||\sum w_j U_j$, is solvable in $O(\sum_{j=1}^{n}(d_j)^{m-1})$ time [32].

With respect to the classification scheme proposed by [4], the setting under consideration can be classified as $P|averse, priv_d\{w_j, d_j\}, U_j|\sum w_j U_j$. In the first field, P indicates that we are considering a setting with parallel machines. The second field refers to the characteristics of the clients (jobs). Here, $averse$ highlights the fact that the clients are averse to uncertain loss and $priv_d\{w_j, d_j\}$ indicates that they possess private information on their weights and due dates, while it is publicly known that these values are elements of discrete sets, i.e. the natural numbers. U_j refers to the fact that each client aims for a schedule where her job is on-time. Finally, in the last field, $\sum w_j U_j$ indicates that the global optimality criterion is the minimization of the total scheduling cost.

The notation is summarized in Table 1.

3. Game Decision Mechanism

We now propose a game mechanism, i.e. a process of decisions that generates a schedule which is satisfactory to all clients. It is initialized by each client $j \in J$ claiming parameters w_j and d_j. Then, an initial schedule is generated. Any approach can be used here. For example, if the clients agree to have equal chances to take any position on any machine, it can be randomly generated. If the clients agree that the mechanism applies any rule to

Table 1: Notation used throughout the paper

J	set of clients	$J = \{1, \ldots, n\}$
M	set of parallel identical machines	$M = \{1, \ldots, m\}$
C_j	completion time of job j	$C_j \in \mathbb{N}$
p_j	processing time of job j	$p_j \in \mathbb{N}$
d_j	due date of job j	$d_j \in \mathbb{N}$
w_j	weight of job j	$w_j \in \mathbb{N}$
d_j^{true}	true due date of job j	$d_j \in \mathbb{N}$
w_j^{true}	true weight of job j	$w_j \in \mathbb{N}$
$U(d_j, C_j)$	cost indicator function	$U : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$
V_j^+	total compensation paid to client j	$V_j^+ \in \mathbb{N}$
V_j^-	total compensation paid by client j	$V_j^- \in \mathbb{N}$
F_j	loss function of client $j \in J$	$F_j = w_j^{true} U(d_j^{true}, C_j) + V_j^- - V_j^+$

generate the initial schedule, then we suggest that it is the best schedule with respect to minimizing the total claimed scheduling cost $\sum_{j=1}^{n} w_j U_j$, which the mechanism can find within a given time limit. The rules of developing this schedule can be known to the clients or not.

Let the initial schedule be aimed at minimizing $\sum_{j=1}^{n} w_j U_j$. If $n \geq 2m + 1$, then the mechanism iterates over all machines that process at least two jobs in the initial schedule and updates this schedule by swapping the job in the first position with the job in the second position. This is done to prevent any client $j \in J$ from claiming false values w_j and d_j in order to take the first position on a machine such that it cannot be taken away by any other job because of a high compensation payment. If $n \leq 2m$, then the number of the second positions in any schedule is less than or equal to the number of the first positions. In this case, let $\hat{k}$ be the number of machines that process at least two jobs in the initial schedule, i.e. $1 \leq \hat{k} \leq n - m$. The mechanism randomly selects $k < \hat{k}$ of these machines and swaps the first and the second job of these machines. If all $n - m$ jobs in the second positions were moved in this case, then a client $j \in J$ who knows that $n \leq 2m$ could claim false values w_j and d_j in order to take a second position and be placed in the first position after the interchange.

If the initial schedule is randomly generated, then no job interchange is needed. Furthermore, if none of the clients knows n and m, then specific handling of the case $n \leq 2m$ is not needed.

Denote the updated schedule as $S^{old} = (S_1^{old}, \ldots, S_m^{old})$, where S_h^{old} is the job sequence on machine h, $h \in M$. This is the input schedule for the first iteration of the game mechanism. Each client $j \in J$, whose job is not in the first position on a machine, receives the completion time of her job in this schedule and a set of possible completion times obtained by placing her job in every position on a machine where it will be completed earlier, assuming that the relative sequence of the other jobs on this machine remains unchanged. She also receives a set E_j of *eligible local strategies*, which includes every strategy (j, g, s) of moving her job to a position s on machine g in S^{old} such that her *claimed savings* emerging from applying this strategy are positive. The definition of claimed savings is given below. The claimed savings are presented together with the corresponding eligible local strategy.

Denote by S^{new} the schedule obtained from S^{old} by applying a job moving strategy (j, g, s). Denote by C_i^{old} and C_i^{new} the completion time of a job $i \in J$ in the old and new schedules, respectively. The claimed savings of client $j \in J$ associated with strategy (j, g, s)

are calculated as follows:

$$A_j(g,s) = w_j U(d_j, C_j^{old}) - w_j U(d_j, C_j^{new}) - \sum_{i \in S_g^{old}: C_i^{new} > C_i^{old}} \Big(w_i U(d_i, C_i^{new}) - w_i U(d_i, C_i^{old}) \Big). \quad (1)$$

Note that the summation is taken over jobs i on machine g which are completed after job j in the new schedule.

If every job moving strategy results in non-positive claimed savings for client $j \in J$, then solely the "no move" strategy is eligible for her, i.e. $E_j = \{\text{"no move"}\}$. For each client $r \in J$, whose job is first on a machine, $E_r = \{\text{"no move"}\}$ by definition.

Each client $j \in J$, whose set E_j contains a job moving strategy different from "no move", submits one of these strategies. Hereafter, the service provider selects and applies one of them. Again, any selection approach can be used here, for example, random selection. If the clients agree that the mechanism applies any selection rule, then we suggest that it selects a strategy that minimizes the total claimed cost $\sum_{j=1}^{n} w_j U_j$. The resulting schedule serves as the input sequence S^{old} in the next iteration of the game mechanism. When making a choice, the client can rank her job moving strategies. For example, she may consider maximizing savings or minimizing job completion time.

Iterations of the decision process repeat until a final schedule is obtained for which no set E_j, $j \in J$, contains an eligible local strategy different from "no move". The jobs are processed by the service provider according to the final schedule. All compensation payments are realized.

We call the described process a *schedule updating game with compensations*. It is summarized in Algorithm 1. In this game, clients are *players*. We define an *equilibrium* (EQ) of

Algorithm 1 Schedule updating game with compensations

Step 1 Each client $j \in J$ claims values w_j and d_j.

Step 2 Calculate an initial schedule $S = (S_1, \ldots, S_m)$ with a given algorithm. Depending on the algorithm used, potentially update S by performing interchanges of jobs in the first and second positions of the job sequences on the machines.

Step 3 Determine the set E_j of eligible local strategies for each client $j \in J$. If $E_j = \{\text{"no move"}\}$ for all $j \in J$, then go to Step 4. Else, each client $j \in J$ with $E_j \neq \{\text{"no move"}\}$ submits exactly one local strategy. Apply one of the strategies, which is selected based on a given algorithm, to update schedule S. Update the compensation payments of all players. Repeat Step 3.

Step 4 Apply schedule S. Realize the compensation payments.

this game as a schedule such that "no move" is the only eligible local strategy for each client. In Section 5, we will show that Algorithm 1 terminates with an EQ in a finite number of iterations.

4. Truthfulness of Clients

In the described game, the clients $j \in J$ can claim false parameters w_j and d_j if there is no chance that this action will increase their loss. Let us show that, for any problem instance and any client, there exist situations in which lying increases a client's loss, and in the remaining situations a client's loss is independent of her truthfulness. Note that no information is available about which of the situations will actually happen, both in a deterministic and probabilistic sense.

Consider a client $j \in J$ who lies about her parameters. Recall that all the parameters are claimed before the schedule updating procedure starts. In any iteration of this procedure, including the first iteration, there is a chance that job j is not in a first position. Hence, assume that this is the case and that a certain strategy is applied. There are four cases to consider:

1. Job j is moved to complete earlier because its moving strategy is applied.
2. Job j is moved to complete earlier because a preceding job is moved to another machine.
3. Job j is moved to complete later because another job is placed before it on the same machine.
4. The completion time of job j does not change.

In Cases 2 and 4, the change of the loss of client j does not depend on the claimed parameters. Hence, we will only have to analyze Cases 1 and 3 in the remainder of this section.

Consider Case 1. Denote by A^{false} the claimed savings of client j, and by A^{true} her claimed savings calculated for the true values w_j^{true} and d_j^{true} and the same completion time. Note that $A^{false} > 0$ because the applied move is an eligible strategy.

If $A^{false} \leq A^{true}$, then $A^{true} > 0$. This implies that the same move would be eligible for the true parameters w_j^{true} and d_j^{true} of client j and, moreover, her loss would be the same or smaller than that for the false parameters.

Assume $A^{false} > A^{true}$. Since the parameters w_i, d_i and p_i, $i \in J \setminus \{j\}$, of other clients are unknown to client j and they can be arbitrary, there is a chance that, in the case of false parameters w_j and d_j, job j will delay jobs whose cost increase client j is not able to compensate from her true cost reduction, i.e. $A^{true} < 0$. In this case, lying will increase the loss of client j. For example, let job j be sequenced immediately after job $i \in J$ on the same machine. Furthermore, let $p_i = p_j$, let both jobs have the same due date being equal to the completion time of job i, and let the true and claimed weights of job j be equal to W and $W + \delta$, $\delta > 0$, respectively, and the claimed weight of job i be equal to $W + \delta/2$. Assume that job j is moved to stand immediately before job i. Then the claimed savings of client j are $A^{false} = \delta/2$, while her true savings are $A^{true} = -\delta/2$. This is illustrated in Figure 1, where the situation after the move is depicted on the right. Now assume that, in the above

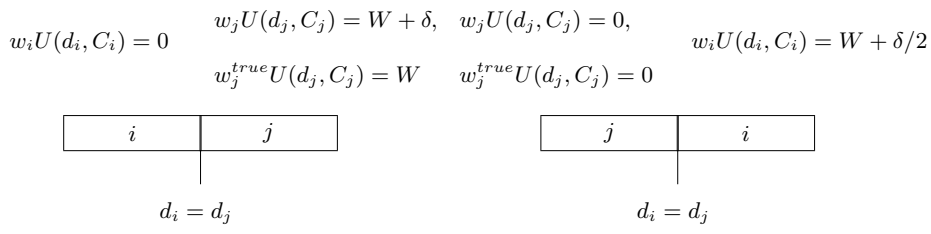


Figure 1: False weight w_j

example, the true and claimed weights of job j are equal to $W + \delta$, $\delta > 0$, but the true due date is equal to $d_j + p_j$. If job j is moved to stand immediately before job i, then the claimed savings of client j are $A^{false} = \delta/2$, while her true savings are $-(W + \delta/2)$. Figure 2 illustrates this setting. Again, the situation after the move is depicted on the right.

Consider Case 3, in which client j may be paid by another client whose job is moved to stand before job j on the same machine. Denote by I^{false} and I^{true} the cost increase of

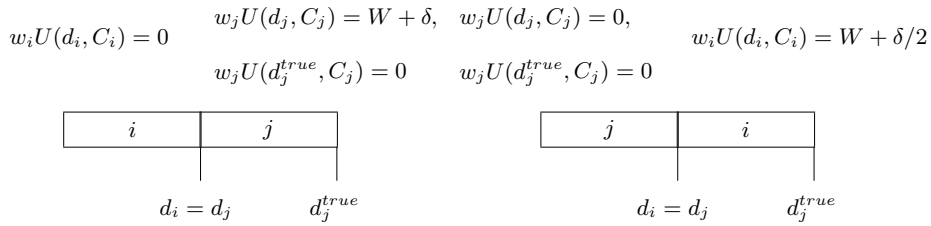


Figure 2: False due date d_j

client j calculated with respect to her claimed and true parameters w_j and d_j, respectively. The compensation paid to client j in Case 3 is equal to I^{false}, and her loss is increased by I^{true}. If $I^{false} \leq I^{true}$, then lying is not profitable. Furthermore, there is no guarantee that Case 3 in which $I^{false} > I^{true}$ will happen with any probability. An increase of the cost of client j from zero to w_j can make job moving strategies of other clients ineligible, while for true values d_j and w_j the cost could be increased to the value w_j^{true} affordable for compensation. Consider the examples depicted in Figures 1 and 2 and assume that job j is sequenced immediately before job $i \in J$ on the same machine (situation on the right) and that job i is moved to stand immediately before job j (situation on the left). For the example in Figure 1, the cost increase of client j with respect to her claimed cost function is equal to $I^{false} = W + \delta$ and the cost increase with respect to her true cost function is equal to $I^{true} = W$. Job i cannot overtake job j in the case of its false weight w_j, because savings of client i are equal to $-\delta/2$ in this case. If weight w_j is true, then job i can overtake job j and the loss of client j can be decreased by $\delta/2$. For the example depicted in Figure 2, the loss of client j is the same with respect to the claimed and true parameters. For false d_j, job i cannot overtake job j, and for true d_j, job j is not compensated.

Thus, we have shown that no client $j \in J$ has an incentive to claim false values w_j or d_j if she is fully averse to uncertain loss. We assumed that the clients are such. Therefore, in the suggested game, they claim true values w_j and d_j.

5. Algorithmic Analysis

This section aims to establish relations between EQs and solutions of the problem $P||\sum w_jU_j$. Some of the proofs of the considered propositions are similar to those in [19] for the problem with $m = 1$.

Let *opt*, F^{opt}, $F^{best\ EQ}$, $F^{worst\ EQ}$, and $F(S)$ denote a social optimum, its (objective function) value, and the values of a best EQ, a worst EQ, and a given schedule S, respectively, for the studied schedule updating game. The ratios $F^{worst\ EQ}/F^{opt}$ and $F^{best\ EQ}/F^{opt}$ are called the *price of anarchy* [33, 34], and the *price of stability* [35, 36, 37], respectively.

Proposition 1. *opt is an EQ and the price of stability is equal to one.*

Proof. Let S^* be *opt* and not an EQ. Then, by the definition of an EQ, some job $j \in J$ can be moved to complete earlier in S^*, so that its cost is decreased from w_j to zero, costs of jobs shifted to complete earlier can also decrease, and the total cost increase of the jobs that became completed later is less than w_j, see (1). This means that $\sum_{j=1}^{n} w_jU_j$ is decreased, which contradicts the optimality of S^*. Hence, *opt* is an EQ. This fact immediately implies that, for the studied game, an EQ always exists and that the price of stability is equal to one. □

The following proposition shows that not every EQ is good with respect to the social criterion $\sum_{j=1}^{n} w_jU_j$. It is proven in [19].

Proposition 2. *The price of anarchy can be infinitely large.*

Note that if EQs are limited to those produced by the designed mechanism, then the price of anarchy depends on the initial schedule. For example, if the total cost $F(S)$ of the initial schedule S satisfies $F(S)/F^{opt} \leq \Delta < \infty$ for any instance, then the price of anarchy does not exceed Δ.

It is interesting to know if a given schedule is an EQ.

Proposition 3. *Any schedule can be recognized as an EQ in $O(n^3)$ time.*

Proof. Let S be an arbitrary schedule. Re-number the jobs such that $t_1 \leq t_2 \leq \cdots \leq t_n$, where t_j is the start time of job j in S for all $j \in J$. There are at most $n(n-1)/2$ schedules that can be obtained from moving each job j, $j \in \{2, \ldots, n\}$, to start at time $t_i < t_j$, for all $i \in \{1, \ldots, j-1\}$. For each schedule, the savings of the moved job can be calculated in $O(n)$ time by formula (1). If the savings are non-positive for all these schedules, then the original schedule S is an EQ. Otherwise, it is no EQ and one of the new schedules has a strictly smaller value $\sum_{j=1}^{n} w_j U_j$ than the one of the schedule S. □

Now we show that an arbitrary EQ can be found in pseudo-polynomial time. Let B denote an upper bound on the value F^{opt}, and let $S^{(B)}$ be a schedule with value B. We have $F(S^{(B)}) = B \leq \sum_{j=1}^{n} w_j$.

Proposition 4. *An EQ can be found in $O(n^3(B - F^{opt} + 1))$ time.*

Proof. Consider schedule $S^{(B)}$. Apply the procedure described in the proof of Proposition 3 to verify in $O(n^3)$ time if $S^{(B)}$ is an EQ. If it is not an EQ, then the procedure finds another sequence, say S', such that $F(S') \leq F(S^{(B)}) - 1 = B - 1$. In this case, apply the EQ verification procedure for the sequence S'. Since cost functions are integer valued and *opt* is an EQ, at most $B - F^{opt} + 1$ sequences will be verified until an EQ is found. □

It follows from Proposition 4, that an EQ can be found in $O(n^4)$ time if all weights are unit, i.e., if $w_j = 1$ for all $j \in J$. Note that the $O(n^3(B - F^{opt} + 1))$ time process of finding an EQ described in the proof of Proposition 4 is not needed for special cases of $P|averse, priv_d\{w_j, d_j\}, U_j|\sum w_j U_j$, if the corresponding special cases of the NP-hard problem $P||\sum w_j U_j$ admit a faster optimal algorithm. For example, if $p_j = p$ for all $j \in J$, then an optimal solution of the problem $P|p_j = p|\sum w_j U_j$ (and thus EQ) can be found in $O(n^3)$ time.

An immediate consequence of the preceding propositions and their proofs is as follows.

Corollary 1. *Algorithm 1, i.e. the proposed schedule updating game with compensations, terminates in a finite number of iterations.*

Observe that a local optimum of the problem $P||\sum w_j U_j$ in the neighborhood defined such that a neighbor of a given schedule is obtained from this schedule by moving a single job to complete earlier, is an EQ. Furthermore, an EQ is a local optimum in the defined neighborhood if $m = 1$. If $m \geq 2$, then an EQ may not be a local optimum, because even a non-eligible move of a job to another machine can decrease the function $\sum_{j=1}^{n} w_j U_j$ due to the possible cost decrease of the jobs sequenced after it. For $m = 1$, the studied schedule updating game is an *exact potential game* because the change of loss of any client as a result of applying its eligible local strategy is equal to the change of the total loss $\sum_{j=1}^{n} F_j = \sum_{j=1}^{n} w_j U_j$, which satisfies the definition of a *potential function* [38]. For $m \geq 2$, the studied problem is a *generalized ordinal potential game* because the loss decrease of any

client as a result of applying its eligible local strategy implies a decrease of the total loss but not vice versa.

The following section elaborates more on the question of whether it is possible to find an EQ in polynomial time.

6. Computational Analysis

In this section, we will computationally analyze Algorithm 1. Most important, we aim to analyze the question of whether or not the number of iterations of Step 3 of Algorithm 1 is significantly affected by the algorithm used in Step 2 of Algorithm 1. The importance of this question stems from the fact that in potential applications (cf. Section 1.2) one typically wants to avoid being faced with communication intensive processes.

6.1. An $O(n^2)$ Algorithm to Determine an EQ

The idea of the following Algorithm 2, which we refer to as WEDD (Weighted Earliest Due Date, cf. also [39]), is based on the fact that there exists an optimal schedule to an instance of $P||\sum w_jU_j$, such that the on-time jobs on each machine are sequenced in the *earliest due date* (EDD) order and the late jobs follow on-time jobs on each machine (see [40]). Here, the EDD order of jobs is such that a job with smaller due date appears in a sequence on a machine earlier than a job with larger due date.

Algorithm 2 WEDD

Step 1 Initialize a list L by sorting all jobs according to their weights in non-increasing order, breaking ties arbitrarily. Let $L[j]$ denote the j-th element of L. Initialize an empty set S_h^{early} of on-time jobs for each machine $h \in M$, an empty set S^{late} of late jobs, and an empty auxiliary set S^{temp}. Set $k := 1$.

Step 2 If $k = n + 1$, then perform Step 4. Else, set $h := 1$.

Step 3 If $h = m + 1$, then set $S^{late} := S^{late} \cup \{L[k]\}$, $k := k + 1$ and perform Step 2. Else, set $S^{temp} := S_h^{early} \cup \{L[k]\}$. Construct an EDD sequence S^{EDD} of the jobs in the set S^{temp}, breaking ties arbitrarily. If all jobs of sequence S^{EDD} are on time, then set $S_h^{early} := S^{temp}$, $k := k + 1$ and go to Step 2. If at least one job of the sequence S^{EDD} is late, then set $h := h + 1$ and repeat Step 3.

Step 4 Output a schedule $S^* = (S_1^*, \ldots, S_m^*)$, which is constructed as follows. On-time jobs are scheduled according to the EDD sequence of S_h^{early} on each machine $h \in M$. Afterwards, late jobs of the set S^{late} are scheduled arbitrarily after the on-time jobs.

In the algorithm WEDD, the jobs are re-numbered in non-increasing order of their weights. The common iteration of WEDD is as follows. A current job k is attempted to be assigned as an on-time job to a current machine h, considering machines in the order $1, \ldots, m$. The on-time jobs already assigned to h and job k are tentatively sequenced in their EDD order. If all these jobs are on-time on machine h, then job k is classified as an on-time job on machine h, and the common iteration repeats for job $k+1$. If at least one job is late on machine h, then job k is attempted to be assigned as an on-time job to machine $h + 1$. If at least one job is late when trying to assign job k to any of the m machines, then job k is classified as late, and the common iteration repeats for job $k + 1$. The algorithm terminates after all jobs have been considered.

Since one iteration k of Step 2 of Algorithm 2, which includes at most m iterations of Step 3, can be implemented to run in $O(n)$ time, algorithm WEDD can be implemented to run in $O(n^2)$ time.

Proposition 5. *Algorithm WEDD constructs an EQ.*

Proof. Consider schedule S^* determined by WEDD. Observe that moving an on-time job to complete earlier in this schedule is not an eligible strategy because its scheduling cost is zero, it cannot be decreased, and possible compensation payments to other jobs can only decrease the utility of this job.

Consider a late job $k \in J$ in the schedule S^*. It follows from the definition of WEDD and the fact that the EDD sequence minimizes the maximum deviation of job completion times from their due dates, that, if job k was moved to stand first on a machine or after any of the jobs $1, \ldots, k-1$ in the schedule S^*, then at least one of the jobs $1, \ldots, k$ would be late. If job $j \leq k-1$ was late, then compensation $w_j \geq w_k$ would have to be paid. If job k itself was late, then the savings of client k would be zero. In either case, moving job k is not an eligible strategy. We deduce that S^* is an EQ. □

Note that the social value $F(S^*)$ of an EQ S^* that is determined by WEDD may be arbitrarily far away from F^{opt}. To see this, assume the opposite, i.e. $F(S^*) \leq \Delta \cdot F^{opt}$ for a given number $\Delta < \infty$ and any problem instance. Then WEDD could hypothetically be used to solve the problem of recognizing $\sum_{j=1}^{n} w_j U_j = 0$ in polynomial time, because for any yes-instance of the latter problem it will produce an EQ S^* with value $F(S^*) \leq \Delta \cdot 0 = 0$. Since recognizing $\sum_{j=1}^{n} w_j U_j = 0$ is an NP-complete in the strong sense problem for the parallel machine environment [30], WEDD cannot do this, unless $\mathcal{P} = \mathcal{NP}$. This fact raises another question to be answered by our computational tests, namely, whether the solutions determined by Algorithm 1 when applying WEDD in Step 2 are of acceptable quality from a practical point of view.

6.2. Computational Tests

We conducted a series of computational experiments on randomly generated instances to analyze the quality of Algorithm 2 with respect to the social criterion $\sum_{j=1}^{n} w_j U_j$ when being used in Step 2 of Algorithm 1. Algorithm 2 and all further algorithms that we will describe in the following were implemented in C++. The computational experiments were performed on a PC with 16 GB of memory and an Intel® Core™ i7 CPU, running at a speed of 3.4 GHz. The operating system was Windows 8.1, 64 bit.

In order to generate benchmark solutions, we implemented twelve versions of Algorithm 1, i.e. the actual game mechanism suggested in Section 3. They differ in the way they generate the initial schedule (Step 2 of Algorithm 1) and in the behavior of clients when selecting an eligible move (Step 3 of Algorithm 1).

We consider six approaches of generating an initial schedule: random generation (denoted by *rand*), algorithm WEDD, a complete enumeration approach (denoted by *enum*) that determines optimal schedules of the underlying scheduling problem $P||\sum w_j U_j$, and three priority rule based approaches. These latter approaches first generate a sorted list of all jobs based on a priority rule (see [41]), namely EDD, shortest processing times (SPT), or weighted shortest processing times (WSPT). Based on this ordering, the jobs are then iteratively assigned to the machines. In each iteration, the algorithm chooses a machine with minimum load. In order to improve the readability of the following figures, we combine the latter three approaches to an algorithm that sequentially executes the priority rule based approaches and returns the overall best solution. We refer to this algorithm as *prio*.

With respect to Step 3 of Algorithm 1, we study two variants of the behavior of clients:

- *Greedy clients*: clients always pick a move with largest saving among their list of eligible moves.

- *Randomly acting clients*: clients randomly select a move out of their list of eligible moves.

We additionally assume that the service provider applies a strategy with largest saving out of the set of strategies that have been submitted by the clients. Algorithm 1 terminates if no player has an eligible move different from "no move", i.e. when an EQ is reached. We denote the eight resulting variants of the game mechanism by $game^{initSchedule}_{behavior}$, where $initSchedule \in \{rand, WEDD, enum, prio\}$ refers to the procedure of generating the initial schedule, and $behavior \in \{greedy, random\}$ refers to the behavior of the clients.

We generated two groups of test instances. The first group features small instances with $m \in [2, 3]$ and $n \in [5, 20]$, for which we were able to generate optimal initial schedules with respect to the social criterion $\sum_{j=1}^{n} w_j U_j$ with the complete enumeration approach. The second group relates to large instances with $m \in [2, 30]$ and $n \in [20, 400]$, that we could not solve to optimality. For both groups, processing times p_j, weights w_j, and due dates d_j, $j \in J$, were randomly drawn from uniform distributions over the intervals $[1, 100]$, $[100, 200]$, and $[p_j, P/m]$, respectively, where $P := \sum_{j=1}^{n} p_j$. If $p_j > P/m$ for some $j \in J$, we set $d_j = p_j$.

For each pair of n and m, we generated a total of ten test instances. For each of the resulting instance sets, we rate the performance of a specific variant of the schedule updating game by calculating the arithmetic mean of the corresponding solution qualities, which are defined as F^*/F', where F^* is the total scheduling cost of the solution determined by the specific variant of the game mechanism, and F' is the scheduling cost of the best solution obtained by any of the considered algorithms. Note that the algorithms that apply complete enumeration are solely considered in the first test set.

In our computational tests, all variants of the schedule updating game, except the ones that apply the complete enumeration approach, terminated in less than 0.6 seconds, even for the largest instances with $n = 400$ and $m = 30$. Hence, runtime comparisons of the distinct procedures in Steps 2 and 3 of Algorithm 1 are of minor interest. Our major interest is to analyze the number of iterations of Step 3, because the corresponding updating procedures include potentially time consuming communication processes between the clients and the operator. For our comparisons, we use the arithmetic mean of the number of iterations of Step 3 for each set of instances.

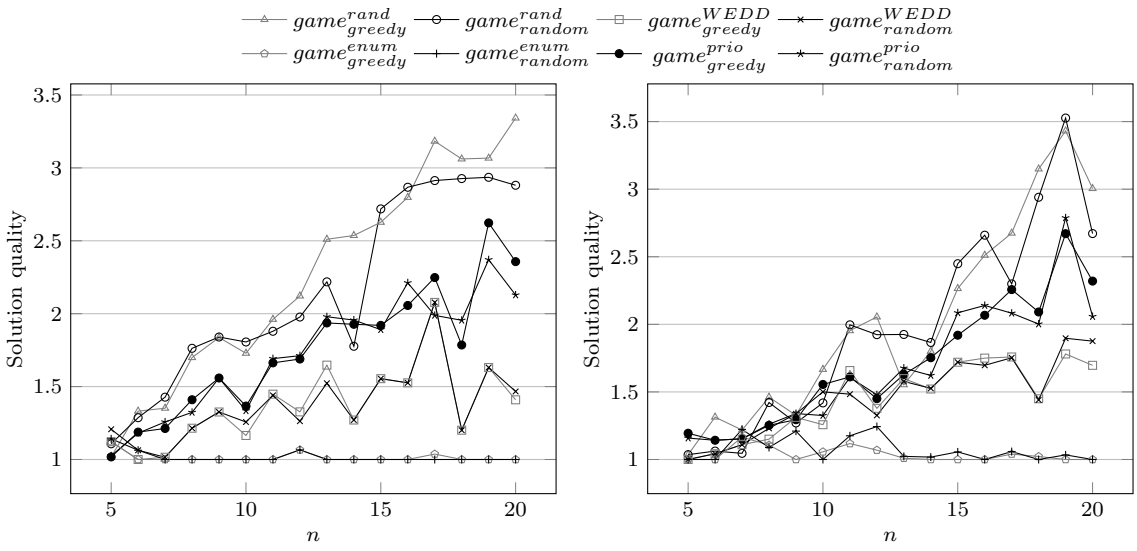
We will first analyze the results for the small instances. In Figure 3, we plot the algorithms' corresponding average qualities over the number of jobs for instances with two (Figure 3a) and three (Figure 3b) machines.

As to be expected, integrating the complete enumeration approach ($game^{enum}_{greedy}$ and $game^{enum}_{random}$) results in the best overall solution quality. However, applying WEDD to determine an initial schedule ($game^{WEDD}_{greedy}$ and $game^{WEDD}_{random}$) results in fairly good solution qualities. Moreover, in comparison to the random generation of initial solutions ($game^{rand}_{greedy}$ and $game^{rand}_{random}$) and the priority rule based generation ($game^{prio}_{greedy}$ and $game^{prio}_{random}$), the average solution quality degrades significantly slower when increasing the number of jobs.

Figure 4 plots the average number of iterations of Step 3 over the number of jobs for instances with two (Figure 4a) and three (Figure 4b) machines.

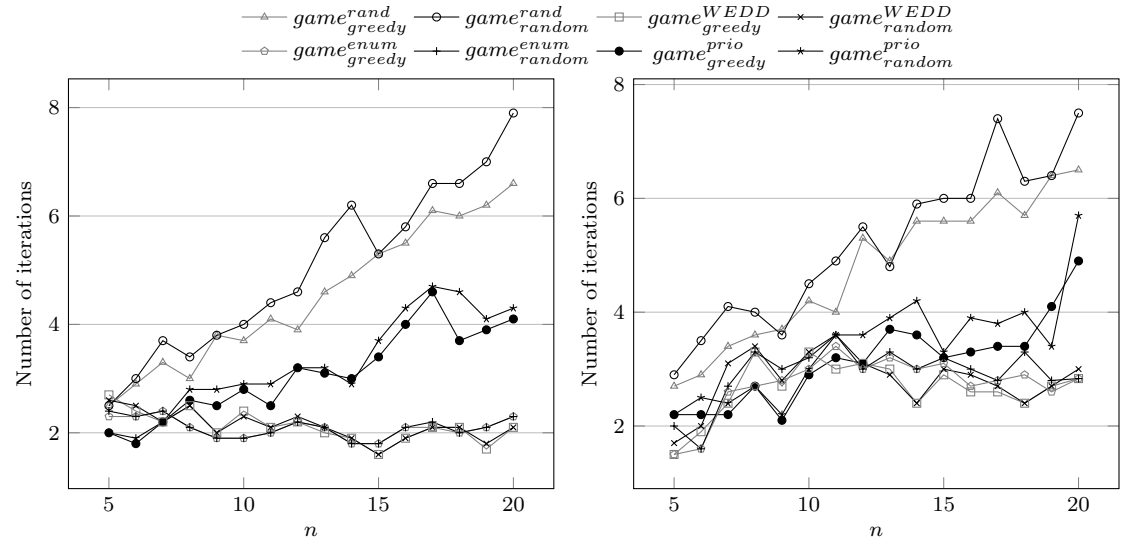
The algorithms based on an initial schedule generated by WEDD or by complete enumeration clearly outperform $game^{rand}_{greedy}$ and $game^{rand}_{random}$ as well as the priority rule based algorithms. We believe that this is a result of Propositions 1 and 5, i.e. the fact that the solutions determined by complete enumeration and WEDD are EQs. These solutions are only slightly modified in Step 2 of the game mechanism, so that less iterations of Step 3 are needed in order to restore the EQ.

We will now turn our attention to the large test instances. Figure 5 plots the corre-



(a) Two machines (b) Three machines

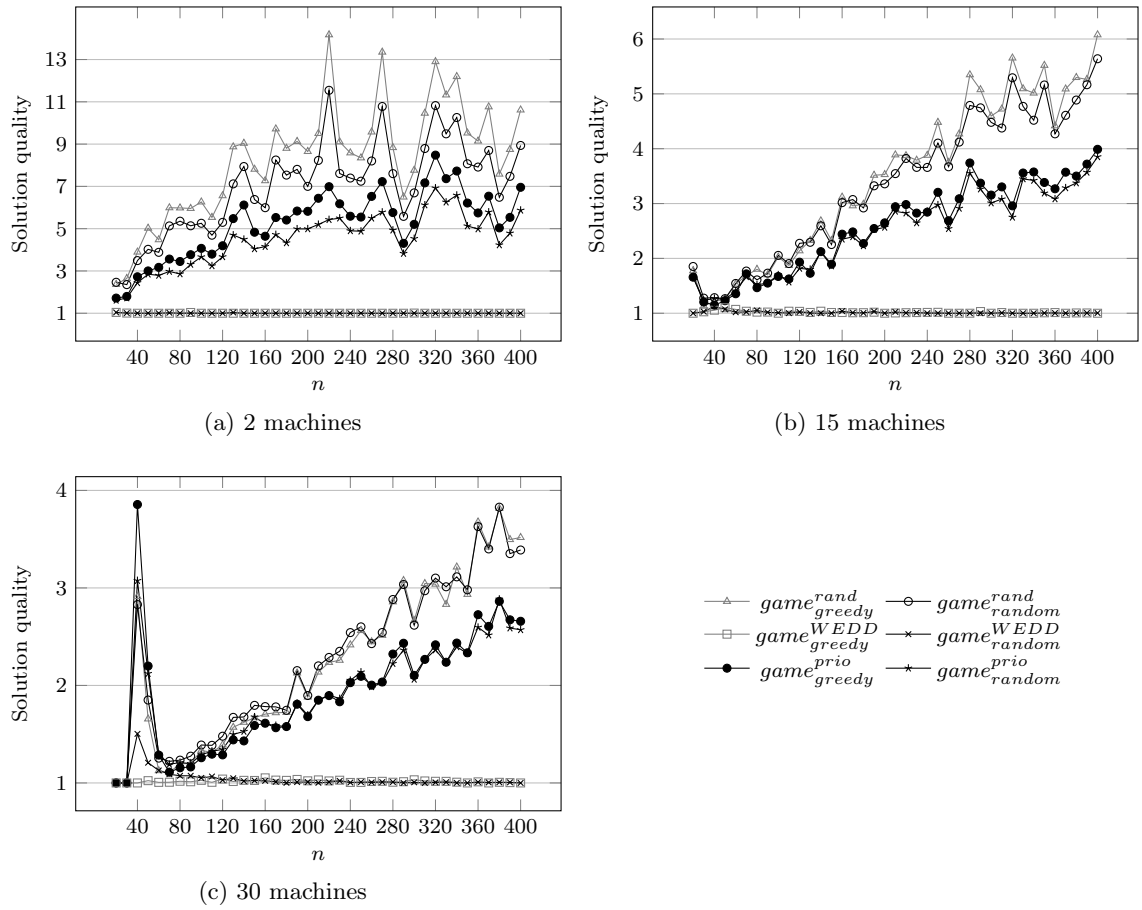
Figure 3: Small test instances - solution quality



(a) Two machines (b) Three machines

Figure 4: Small test instances - number of iterations of Step 3

sponding results on the average solution quality.



(a) 2 machines

(b) 15 machines

(c) 30 machines

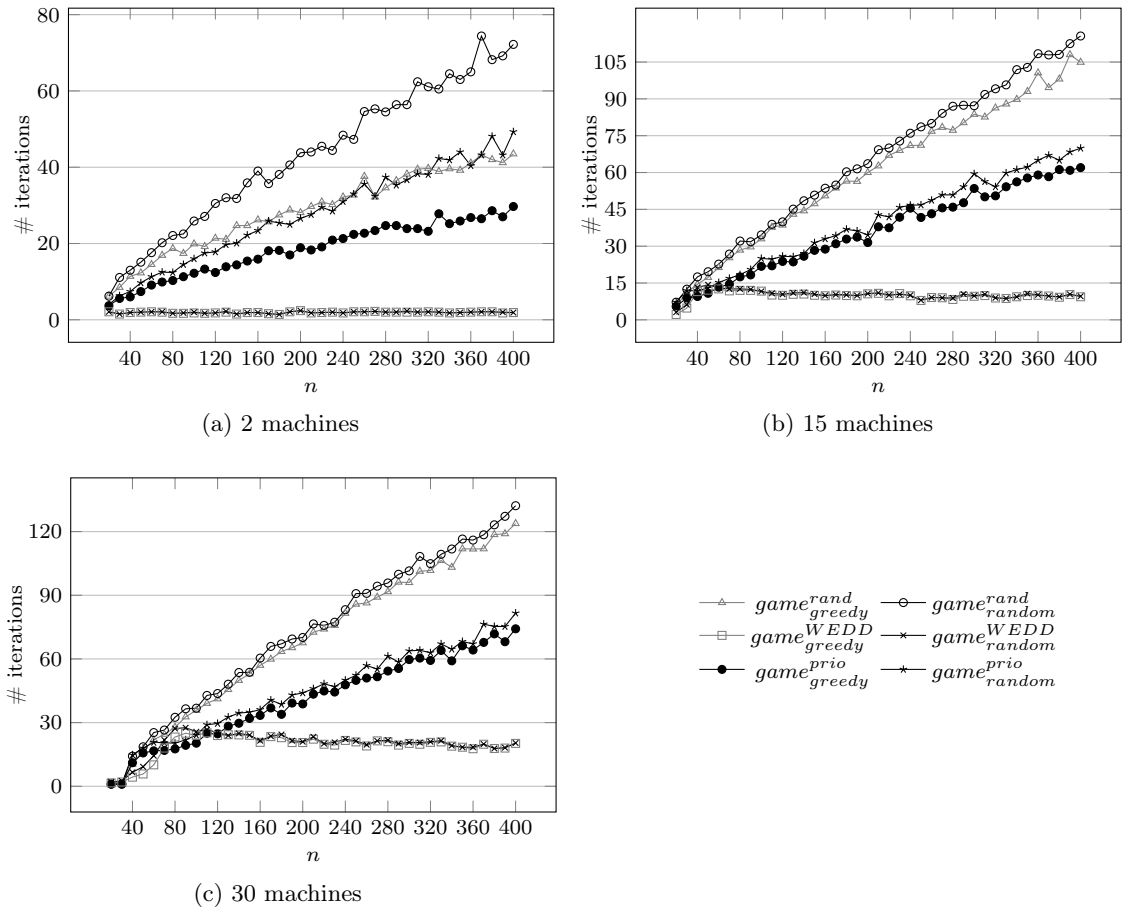
Figure 5: Large test instances - solution quality

Again, generating initial schedules by using WEDD clearly pays off when compared to random and priority rule based generation of initial schedules. The difference of these strategies with respect to solution quality increases for larger ratios n/m. An interesting detail is that, especially in the case of two machines, $game^{rand}_{random}$, on average, results in better solutions than $game^{rand}_{greedy}$. The same can be seen for $game^{prio}_{random}$ and $game^{prio}_{greedy}$. This effect does not ocuur when using WEDD.

Figure 6 depicts the average number of iterations of Step 3 of Algorithm 1 over the number of jobs for the large instances.

Again, the algorithms applying WEDD outperform the algorithms based on random and priority rule based initial schedules. While, on average, $game^{WEDD}_{greedy}$ and $game^{WEDD}_{random}$ need less than m iterations for all considered n, the number of iterations of Step 3 seems to increase linear in n for the other considered variants of the algorithm. For the case of 30 machines (Figure 6c) and random generation of initial schedules, this results in more than 120 iterations for large n, and substantiates our believe that the game mechanism performs best when carefully generating an initial schedule that is an EQ.

Summing up, even though the social value of an EQ determined by WEDD may theoretically be arbitrarily far away from F^{opt}, $game^{WEDD}_{greedy}$ and $game^{WEDD}_{random}$ result in high quality solutions. Moreover, integrating WEDD into the game mechanism is well suited from a practical perspective because it results in an appropriate level of communication between operator and clients.



(a) 2 machines

(b) 15 machines

(c) 30 machines

Figure 6: Large test instances - number of iterations of Step 3

7. Summary and Suggestions for Future Research

This paper has suggested a game decision mechanism for fully risk averse clients who compete for the processing of their jobs not later than by their respective due dates in a parallel machine environment. For any agent, due date violation implies a cost. The mechanism aims at minimizing social cost, which is the sum of due date violation costs of all clients. It includes compensations to clients who suffer from a decision of another client. We have shown that the clients have no incentive to claim false due dates and costs when the suggested game decision mechanism is applied. Furthermore, algorithmic aspects of the mechanism have been analyzed. The price of stability is equal to one, the price of anarchy is infinitely large. An $O(n^2)$ algorithm to find an EQ has been suggested. While the social value of this EQ can be infinitely greater than the social optimum, computer experiments have demonstrated that the game mechanism results in high quality solutions when applying this algorithm. Furthermore, from a practical perspective, the resulting game mechanism features an appropriate level of communication between operator and clients.

For future research, it is interesting to find financial rules ensuring that the clients are truthful in case of accepting risky decisions. Furthermore, precedence relations on the set of jobs are an interesting extension of the model, which requires additional investigation. Finally, note that some of the results of this article carry over to other objective functions that may be appropriate for other applications and may deserve a detailed investigation.

References

References

[1] N. Nisan, A. Ronen, Computationally feasible VCG mechanisms, in: Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00), ACM, New York, 2000, pp. 242–252.

[2] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani (Eds.), Algorithmic Game Theory, Cambridge University Press, Cambridge, 2007.

[3] V. Krishna, Auction Theory, Academic Press, Amsterdam, 2010.

[4] D. Kress, S. Meiswinkel, E. Pesch, Mechanism design for machine scheduling problems: Classification and literature overview, OR Spectrum 40 (3) (2018) 583–611.

[5] N. Nisan, A. Ronen, Algorithmic mechanism design (extended abstract), in: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, STOC '99, ACM, 1999, pp. 129–140.

[6] N. Nisan, A. Ronen, Algorithmic mechanism design, Games and Economic Behavior 35 (1–2) (2001) 166–196.

[7] E. Angel, E. Bampis, F. Pascual, Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science 369 (1–3) (2006) 157–168.

[8] H. Hamers, F. Klijn, J. Suijs, On the balancedness of multiple machine sequencing games, European Journal of Operational Research 119 (3) (1999) 678–691.

[9] M. Mitra, Mechanism design in queueing problems, Economic Theory 17 (2) (2001) 277–305.

[10] J. Suijs, On incentive compatibility and budget balancedness in public decision making, Economic Design 2 (1) (1996) 193–209.

[11] B. Heydenreich, R. Müller, M. Uetz, Games and mechanism design in machine scheduling - an introduction, Production and Operations Management 16 (4) (2007) 437–454.

[12] G. Christodoulou, E. Koutsoupias, Mechanism design for scheduling, Bulletin of the EATCS 97 (2009) 40–59.

[13] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Węglarz, Handbook on Scheduling: from Theory to Applications, Springer, Berlin, 2007.

[14] J. Y.-T. Leung (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton, 2004.

[15] D. Fudenberg, J. Tirole, Game Theory, MIT Press, Cambridge, 1991.

[16] M. J. Osborne, An Introduction to Game Theory, Oxford University Press, New York, 2004.

[17] J. von Neumann, O. Morgenstern, Theory of Games and Economic Behavior, Princeton University Press, Princeton, 1944.

[18] N. Boysen, M. Fliedner, F. Jaehn, E. Pesch, A survey on container processing in railway yards, Transportation Science 47 (3) (2013) 312–329.

[19] M. Y. Kovalyov, E. Pesch, A game mechanism for single machine sequencing with zero risk, Omega 44 (2014) 104–110.

[20] D. Kahneman, A. Tversky, Prospect theory: An analysis of decision under risk, Econometrica 47 (2) (1979) 263–292.

[21] A. Tversky, D. Kahneman, Loss aversion in riskless choice: A reference-dependent model, The Quarterly Journal of Economics 106 (4) (1991) 1039–1061.

[22] S. Du, Y. Zhu, T. Nie, H. Yu, Loss-averse preferences in a two-echelon supply chain with yield risk and demand uncertainty, Operational Research 18 (2) (2018) 361–388.

[23] R. G. Fryer Jr., S. D. Levitt, J. List, S. Sadoff, Enhancing the efficacy of teacher incentives through loss aversion: A field experiment, National Bureau of Economic Research, Working Paper No. 18237 (2012).

[24] A. Imas, S. Sadoff, A. Samek, Do people anticipate loss aversion?, Management Science 63 (5) (2017) 1271–1284.

[25] D. Ellsberg, Risk, ambiguity, and the Savage axioms, The Quarterly Journal of Economics 75 (4) (1961) 643–669.

[26] M. J. Machina, M. Siniscalchi, Ambiguity and ambiguity aversion, in: M. J. Machina, W. K. Viscusi (Eds.), Handbook of the Economics of Risk and Uncertainty, Vol. 1, Elsevier, Amsterdam, 2014, pp. 729–807.

[27] R. K. Sarin, M. Weber, Effects of ambiguity in market experiments, Management Science 39 (5) (1993) 602–615.

[28] H. W. Chesson, W. K. Viscusi, Commonalities in time and ambiguity aversion for long-term risks, Theory and Decision 54 (1) (2003) 57–71.

[29] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann Discrete Math 5 (1979) 287–326.

[30] M. R. Garey, D. S. Johnson, "strong" NP-completeness results: Motivation, examples, and implications, Journal of the ACM 25 (3) (1978) 499–508.

[31] P. Brucker, S. Knust, Complexity results for scheduling problems, `http://www2.informatik.uni-osnabrueck.de/knust/class/` (2009).

[32] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, Sequencing and scheduling: algorithms and complexity, in: S. C. Graves, A. H. G. Rinnooy Kan, P. H. Zipkin (Eds.), Logistics of Production and Inventory, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 1993, pp. 445–522.

[33] E. Koutsoupias, C. Papadimitriou, Worst-case equilibria, in: Proceedings of STACS 1999, LNCS 1563, Springer, Berlin, 1999, pp. 404–413.

[34] C. Papadimitriou, Algorithms, games, and the internet, in: Proceedings of STOC/ICALP, ACM, 2001, pp. 749–753.

[35] A. S. Schulz, N. S. Moses, On the performance of user equilibria in traffic networks, in: Proceedings of SODA, ACM-SIAM, New York, Philadeplphia, 2003, pp. 86–87.

[36] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, T. Roughgarden, The price of stability for network design with fair cost allocation, in: Proceedings of FOCS 2004, IEEE, 2004, pp. 295–304.

[37] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, T. Roughgarden, The price of stability for network design with fair cost allocation, SIAM Journal on Computing 38 (4) (2008) 1602–1623.

[38] D. Monderer, L. S. Shapley, Potential games, Games and Economic Behavior 14 (1) (1996) 124–143.

[39] D. Kress, S. Meiswinkel, E. Pesch, Incentive compatible mechanisms for scheduling two-parameter job agents on parallel identical machines to minimize the weighted number of late jobs, Discrete Applied Mathematics 242 (2018) 89–101.

[40] E. L. Lawler, J. M. Moore, A functional equation and its application to resource allocation and sequencing problems, Management Science 16 (1) (1969) 77–84.

[41] R. Haupt, A survey of priority rule-based scheduling, OR Spektrum 11 (1) (1989) 3–16.