Incentive Compatible Mechanisms for Scheduling Two-Parameter Job Agents on Parallel Identical Machines to Minimize the Weighted Number of Late Jobs[†]

Dominik Kress[a,*], Sebastian Meiswinkel[a], Erwin Pesch[a,b]

[a] *University of Siegen, Management Information Science, Kohlbettstr. 15, 57068 Siegen, Germany*
[b] *Center for Advanced Studies in Management, HHL Leipzig, Jahnallee 59, 04109 Leipzig, Germany*

Abstract

We consider the problem of designing polynomial time truthful mechanisms for machine scheduling problems with parallel identical machines where some of the jobs' characteristics are private information of their respective owners and a central decision maker is in charge of computing the schedule. We study a two-parameter setting, where weights and due dates are private information while processing times are publicly known. The global objective is to minimize the sum of the weights of those jobs that are completed after their due dates. We derive a set of properties that is equivalent to the well known condition of cycle monotonicity, which is a general condition for truthful mechanisms in non-convex valuation function domains. Our results utilize knowledge about the underlying scheduling problem, so that the resulting properties are easier to implement and verify than the general condition of cycle monotonicity. We illustrate the use of our results by analyzing an example algorithm that has recently been proposed in the literature for the case of one machine.

Keywords: Algorithmic mechanism design, Machine scheduling, Truthfulness, Game theory, Logistics

1. Introduction and Contribution

When analyzing exact and heuristic methods for solving scheduling problems, we often assume that a *central decision maker* is equipped with all relevant data related to the problem. However, there exist many real world applications where this is not the case because part of the relevant data is private information of selfish players who aim to influence the solution determined by the scheduling algorithm by submitting false information to the decision maker. In some cases, however, the decision maker can extract the true information by designing an appropriate algorithm that sets the right incentives for these players. This in turn enables the

*Corresponding author
Email addresses: 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 **Discrete Applied Mathematics**, available online: https://doi.org/10.1016/j.dam.2017.08.026
© 2018. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/

decision maker to generate "fair" solutions with respect to some *social criterion* that considers the interests of all players. The design of such algorithms is subject of a field of research that is usually referred to as *algorithmic mechanism design* [1].

1.1. Basic Problem Setting and Applications

In this paper we will consider scheduling problems with parallel (identical) machines and publicly known processing times. These problems will be considered in the context of algorithmic mechanism design, with the job-owners being strategic players or *agents*. The agents are assumed to be risk-neutral. Each job-owner reports a *valuation function* to the mechanism. This valuation function may deviate from the true valuation function, which is private information of the job-owner. The mechanism itself is composed of a *social choice function* and *payment functions*. In the context of scheduling problems, the social choice function (or *allocation function*) determines a feasible schedule based on the valuation functions reported to the mechanism. A typical objective of the social choice function is maximizing social welfare, which corresponds to maximizing the sum of all valuation functions. However, each job-owner selfishly aims to maximize her utility function, which corresponds to the sum of her valuation of the schedule and a corresponding (potentially negative) payment from the mechanism to the job-owner. Thus, it is likely that the job-owner lies about the valuation function in order to achieve a greater utility function value than in the case of reporting the true valuation function. Obviously, if the job-owners do not report their true valuation functions to the mechanism, it is impossible to design a mechanism that maximizes social welfare. To overcome this problem, it is necessary to design the mechanism to be (dominant strategy) *incentive compatible* or *truthful*. That is, the mechanism must guarantee that reporting the true valuation function always maximizes the utility function of a rationally acting agent.

As described by Kovalyov and Pesch [2], applications of this problem setting can, for instance, be found in the field of intermodal transport, where some kind of service provider operates cranes in a container terminal (e.g. at a sea port or a rail-road terminal) to load and unload trains. The service provider has service contracts with its customers. Each contract is related to a specific customer of the service provider and vice versa. Each customer requests a train to be loaded or unloaded in a given planning period. These requests correspond to jobs to be processed by the service provider. A similar setting may arise when considering the problem of determining an execution sequence for computer tasks that have been accepted by a computing service provider who operates computing devices. In both examples, the customers compete for quick execution of their jobs in the schedule determined by the service provider for the planning period. The processing time for each job is publicly known. Jobs may incur additional costs to their owners if their completion time is too large, for example because of strict deadlines and corresponding contractual penalties. The related parameters are private information of the customers. The service provider's revenue for executing a job is fixed. Hence, the service provider seeks to determine a "fair" schedule that takes into account the interests of all customers. To generate this schedule, the service provider must know the private information of the players. Hence, the service provider must design an incentive compatible algorithm for scheduling the execution of the jobs.

1.2. Related Literature

A general introduction to the field of algorithmic mechanism design can be found in Nisan et al. [3]. Additionally, there is a fairly large amount of publications dealing with mechanism design in the context of machine scheduling. An excellent overview is given by Heydenreich et al. [4]. A literature overview and a classification scheme is presented by Kress et al. [5].

One of the most important general results in the field of mechanism design is the *Vickrey-Clarke-Groves mechanism* (VCG mechanism), that was suggested by Vickrey [6] and generalized by Clarke [7] and Groves [8]. A mechanism is called a VCG mechanism, if the social choice function maximizes social welfare, i.e. the sum of all valuation functions, and if the payment functions are of some special structure. A VCG mechanism is incentive compatible, but a major drawback is the need for finding optimal solutions to the underlying problem of maximizing social welfare, which may be NP-hard (see, for instance, Nisan [9]). Hence, in the context of scheduling problems, VCG mechanisms are oftentimes not appropriate even if the objective function of the specific scheduling problem corresponds to maximizing social welfare. One must therefore make use of other theoretical results related to incentive compatibility that are suitable for approximate and heuristic algorithms. These results oftentimes turn out to "boil down to a certain algorithmic condition of *monotonicity*" [10].

One can identify two streams of literature dealing with mechanism design in the context of scheduling problems. The first group of publications presumes that the machines are selfish agents (*machine agents*); see Christodoulou and Koutsoupias [11] and Kress et al. [5] for an overview. These papers follow the seminal work of Nisan and Ronen [12] and include Lavi and Swamy [10], Archer and Tardos [13], Christodoulou et al. [14], Koutsoupias and Vidali [15]. The second stream of literature assumes that the jobs are selfish agents (*job agents*), which is the perspective taken in this paper.

Angel et al. [16, 17, 18], Auletta et al. [19], and Christodoulou et al. [20], for example, consider the design of incentive compatible mechanisms in different settings with parallel identical machines, parallel related machines, and parallel unrelated machines. The job agents may manipulate the schedule by providing false information regarding the processing times. Angel et al. [16] also consider online settings, where the existence of jobs is unknown until their release dates. The global objective in all settings is to minimize the makespan.

Other authors analyze the global objective of minimizing the total weighted completion time. Duives et al. [21] and Hoeksma and Uetz [22], for instance, assume that there is only one machine and restrict themselves to considering discrete valuation function domains. They consider a one-parameter setting, where the processing time of each job is public knowledge and the job's weights are private information (see also Hain and Mitra [23] for a related model with processing times being private information), and a two-parameter setting, where both processing times and weights are private information. They derive *optimal mechanisms* that are not only truthful, but at the same time minimize the total (expected) payments that are made to the job-owners. In some applications, however, one may want to achieve different properties of the payments. For example, Suijs [24] presents ("*budget balanced*") VCG payment functions for the one-parameter case in continuous valuation function domains such that the clients, on average, neither win nor lose money (see also [25, 26, 27] for related and more general results).

Some papers deal with mechanism design in *online machine scheduling.* For example, Heydenreich et al. [28] study an online version of parallel machine scheduling to minimize total weighted completion time, where processing times, weights and release dates are private information of the job-owners. Porter [29] additionally considers private information on due dates in the one machine case where the global objective is to minimize the sum of the weights of those jobs that are completed after their respective due dates.

1.3. Contribution and Outline of this Paper

Scheduling problems with the global objective of minimizing the sum of the weights of those jobs that are completed after their respective due dates have received little attention in the mechanism design literature (an example is [29]). We will contribute to closing this gap by considering a setting with job-owners controlling two private parameters, namely the weights and due dates of their jobs, and are therefore referred to as *two-parameter agents.* Processing times are assumed to be publicly known. As mentioned above, one will typically have to make use of monotonicity conditions when constructing incentive compatible mechanisms for NP-hard scheduling problems. In case of non-convex valuation function domains, the monotonicity condition of interest is usually referred to as *cycle monotonicity.* However, due to the general nature of this condition, it is not always easy to design an algorithm that is cycle monotone. We therefore contribute to the literature by deriving a set of conditions that is equivalent to cycle monotonicity for the scheduling problems under consideration in this paper. Our results utilize knowledge about the underlying optimization problem, so that the resulting conditions are easier to implement and verify than the general condition of cycle monotonicity. We will illustrate the use of our results by analyzing an example algorithm that has recently been proposed by Kovalyov and Pesch [2], who consider a similar setting for risk-averse agents and one machine. Our results therefore also extend these authors' prior findings.

The remainder of this paper is organized in four sections. Section 2 introduces the notation, the scheduling problems, as well as the mechanism design setting under consideration in this paper in detail. Furthermore, it presents the monotonicity condition of interest. Section 3 then presents our main result, i.e. the derivation of conditions that guarantee cycle monotonicity for our scheduling domain, which is then illustrated based on an example algorithm for the case of one machine in Section 4. Finally, Section 5 summarizes the paper.

2. Preliminaries

We will start our deliberations by introducing the relevant scheduling problems and the related mechanism design setting in detail in Sections 2.1 and 2.2. We will then introduce some additional notation in Section 2.3 and define the condition of cycle monotonicity and its implications in Section 2.4.

The number sets used throughout this paper are defined in Table 1.

Table 1: Number sets used throughout the paper

$\mathbb{R}$, $\mathbb{Q}$	real (rational) numbers
$\mathbb{R}_{>0}$, $\mathbb{Q}_{>0}$	positive real (rational) numbers
$\mathbb{R}_{\geq 0}$, $\mathbb{Q}_{\geq 0}$	non-negative real (rational) numbers

2.1. Scheduling Domain

The scheduling problems under consideration in this paper presume that there is a set $J = \{1, \dots, n\}$ of $n \geq 2$ jobs and a set $M = \{m_1, \dots, m_m\}$ of m parallel (identical) machines. Each job $i \in J$ must be processed exactly once by an arbitrary machine $m_j \in M$. Each machine processes jobs one after another, starting from time zero, without idle time. Preemption of jobs is not allowed. We denote the processing time of job $i \in J$ by $t_i \in \mathbb{Q}_{>0}$. The set of all feasible solutions of the problem of scheduling all jobs $i \in J$ on machines $m_j \in M$ is denoted by O. A given schedule $o \in O$ determines the completion time $C_i(o)$ of all jobs $i \in J$. In addition to its processing time t_i, each job $i \in J$ is associated with a weight $w_i \in \mathbb{Q}_{\geq 0}$ and a due date $d_i \in \mathbb{Q}_{\geq 0}$. If i is on time, that is, if it completes before or at d_i in a given schedule $o \in O$, then it incurs no cost to the owner of the job. Otherwise, the job is late and it incurs cost w_i.

We define a corresponding cost indicator function $U_{d_i} : \mathbb{R} \to \{0, 1\}$, which is equal to one if its argument is larger than d_i and zero otherwise. The problem is to find a schedule $o \in O$ that minimizes the total scheduling cost of all job-owners, $\sum_{i=1}^{n} w_i U_{d_i}(C_i(o))$, i.e. that maximizes social welfare. This problem is denoted by $P||\sum w_i U_i$ in the literature [30] and is known to be NP-hard in the strong sense for m arbitrary [30, 31, 32]. When considering only one machine, i.e. $m = 1$, the problem is weakly NP-hard [33, 34] and it is denoted by $1||\sum w_i U_i$ [30].

Table 2 summarizes the scheduling notation used throughout the paper.

Table 2: Scheduling notation used throughout the paper

J	set of jobs	$J = \{1, \dots, n\}$
M	set of parallel (identical) machines	$M = \{m_1, \dots, m_m\}$
O	set of feasible solutions of scheduling problem	
C_i	completion time of job $i \in J$	$C_i : O \to \mathbb{R}_{>0}$
t_i	processing time of job $i \in J$	$t_i \in \mathbb{Q}_{>0}$
d_i	due date of job $i \in J$	$d_i \in \mathbb{Q}_{\geq 0}$
w_i	weight of job $i \in J$	$w_i \in \mathbb{Q}_{\geq 0}$

2.2. Mechanism Design Domain

We assume that each job $i \in J$ has an associated *owner* (also referred to as *customer* or *client*). Moreover, each owner possesses exactly one job. Thus, in the remainder of this paper we can identify a job by its owner and vice versa. Each job-owner $i \in J$ has a (true) *valuation function* $v_i^t : O \to \mathbb{R}$ that maps every feasible schedule of the considered scheduling domain to a real value. Negative values can, for example, relate to costs incurred to the job-owner due to waiting for the associated job to be completed.

The mechanism design setting described in Section 1.1 is depicted in more detail in Figure 1. As mentioned above, each job-owner $i \in J$ reports a valuation function v_i, which may

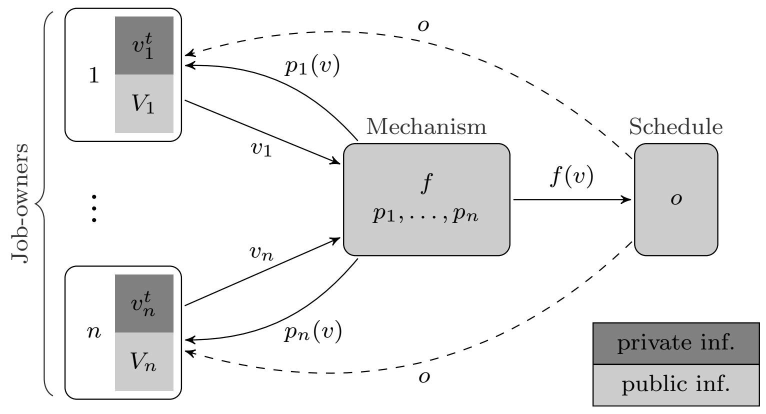

Figure 1: Problem setting

deviate from the true valuation function v_i^t, to the mechanism. v_i^t is private information of the job-owner and is thus sometimes referred to as the job-owner's *type*. Each valuation function v_i, $i \in J$, is element of a publicly known set $V_i \subseteq \mathbb{R}^{|O|}$, which is referred to as the valuation function domain. We define $V := V_1 \times \cdots \times V_n$. Furthermore, we denote the vector of all valuation functions reported to the mechanism by $v = (v_1, \ldots, v_n)$ and the vector of all valuation functions reported to the mechanism except of v_i by $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$. For the sake of notational convenience, we will use v and (v_i, v_{-i}) interchangeably.

In the context of the scheduling domain described in Section 2.1, we will consider w_i and d_i as private information of job-owner $i \in J$. Thus, the valuation functions v_i take the form $v_i(o) = -w_i U_{d_i}(C_i(o))$ with $o \in O$ for all $i \in J$. Hence, $V_i \neq \mathbb{R}^{|O|}$. The processing times of all jobs are assumed to be public information. This setting is denoted by $P|priv\{w_i, d_i\}, U_i| \sum w_i U_i$ when using the classification scheme of Kress et al. [5]. Basically, this scheme extends the well known three field notation of Graham et al. [30], in our case $P|| \sum w_i U_i$, by including mechanism design related information: *priv*$\{w_i, d_i\}$ refers to the fact that job-owners posses private information on weights and due dates, and U_i indicates that each job-owner aims for having her job completed on time.

The *mechanism* $(f, p_1, \ldots, p_n)$ itself is composed of a *social choice function* $f : V \to O$ and *payment functions* $p_1, \ldots, p_n$, with $p_i : V \to \mathbb{R}$ for all $i \in J$. A feasible schedule $o \in O$ determined by the social choice function determines the completion time $C_i(f(v))$ of all jobs $i \in J$.

The *utility function* $u_i : V \to \mathbb{R}$ of player $i \in J$ is (in the considered case of risk-neutral agents) defined as $u_i(v_i, v_{-i}) := v_i^t(f(v_i, v_{-i})) + p_i(v_i, v_{-i})$. Hence, in order to be (dominant strategy) *incentive compatible* or *truthful*, the mechanism must guarantee that $u_i(v_i^t, v_{-i}) \geq u_i(v_i, v_{-i})$ for all $i \in J$, all $v_i \in V_i$, and all $v_{-i} \in V_{-i}$.

Table 3 summarizes the mechanism design notation used throughout the paper.

Table 3: Mechanism design notation used throughout the paper

V_i	valuation function domain for client $i \in J$	
V	Cartesian product of sets V_i, $i \in J$	$V = V_1 \times \cdots \times V_n$
V_{-i}	Cartesian product of sets V_j, $j \in J \setminus \{i\}$	$V = V_1 \times \cdots \times V_{i-1} \times V_{i+1} \times \cdots \times V_n$
f	social choice function	$f : V \to O$
v_i^t	true valuation function of client $i \in J$	$V_i \ni v_i^t : O \to \mathbb{R}$
v_i	claimed valuation function of client $i \in J$	$V_i \ni v_i : O \to \mathbb{R}$
p_i	payment function for client $i \in J$	$p_i : V \to \mathbb{R}$
u_i	utility function of client $i \in J$	$u_i(v) = v_i^t(f(v)) + p_i(v)$
v_{-i}	vector of claimed valuation functions except v_i, $i \in J$	$v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$
v	vector of claimed valuation functions	$v = (v_1, \ldots, v_n)$ $v = (v_i, v_{-i})$, $i \in J$

2.3. Additional notation

In order to ease the notation throughout the remainder of this paper, we will sometimes denote v_i, $i \in J$, by $v_i^{w,d}$ for a given weight $w \in \mathbb{Q}_{\geq 0}$ and due date $d \in \mathbb{Q}_{\geq 0}$ that is committed to the mechanism by client $i \in J$.

Obviously, social choice functions in the context of scheduling problems will have to be established (implemented) by scheduling algorithms. With respect to these algorithms, we will restrict ourselves to deterministic algorithms. In order to simplify the proofs throughout this paper, we introduce some related additional notation, which is summarized in Table 4.

Table 4: Additional notation for $v_{-i} \in V_{-i}$ fixed

c_i^{late}	Smallest completion time of job $i \in J$ if it is scheduled as a late job by the scheduling algorithm.
W_i^d	Set of all w such that $C_i(f(v_i^{w,d}, v_{-i})) \leq d$, with $i \in J$ and d fixed.
$w_i^{min}(d)$	$:= \min W_i^d$, $i \in J$
$w_i^{inf}(d)$	$:= \inf W_i^d$, $i \in J$

Assume $i \in J$ to be an arbitrary job-owner and let $v_{-i} \in V_{-i}$ be fixed. First, consider all $v_i \in V_i$ that result in a schedule with job i being late with respect to the due date committed to the mechanism by the respective job-owner (in the following, we will say that job i is "scheduled as a late job") and denote the smallest of the corresponding completion times of job i by c_i^{late}. c_i^{late} is well defined because $t_i \in \mathbb{Q}_{>0}$ and $d_i \in \mathbb{Q}_{\geq 0}$. Second, we define $W_i^d := \{w \in \mathbb{Q}_{\geq 0} | v_i^{w,d} \in V_i,\ C_i(f(v_i^{w,d}, v_{-i})) \leq d\}$ and $w_i^{min}(d) := \min W_i^d$ for a given $d \in \mathbb{Q}_{\geq 0}$. We are aware that $w_i^{min}(d)$ does not always exist. Nevertheless, for the sake of brevity of the proofs in the following sections, we will use $w_i^{min}(d)$. In case of its non-existence, it will always be easy to bring forward analogous arguments by using $w_i^{inf}(d) := \inf W_i^d$ instead of $w_i^{min}(d)$ and replacing some $\leq$-signs by $<$-signs and vice versa. If the argumentation is independent of using $w_i^{min}(d)$ or $w_i^{inf}(d)$, we will use the latter. Furthermore, we define $w_i^{inf}(d) := \infty$ if $W_i^d = \emptyset$, which may, for example, be the case if $d < t_i$. We will refer to $w_i^{inf}(d)$ or $w_i^{min}(d)$ as a "threshold value" with respect to d.

2.4. Cycle Monotonicity

Monotonicity conditions for incentive compatible mechanisms can become fairly complex when the sets V_i, $i \in J$, are restricted, i.e. when the valuation function domains are proper subsets of $\mathbb{R}^{|O|}$ (see, for example, Krishna [35], Lavi et al. [36]). As mentioned above, the basic condition of interest in our setting is usually referred to as *cycle monotonicity* (see, for example, Lavi and Swamy [10]).

Definition 1 (**Cycle monotonicity**). *A social choice function f is cycle monotone if for every player $i \in J$, every $v_{-i} \in V_{-i}$, every integer K, and every $v_i^1, \ldots, v_i^K \in V_i$,*

$$\sum_{k=1}^{K} \left(v_i^k(a_k) - v_i^k(a_{k+1}) \right) \geq 0, \tag{1}$$

where $a_k := f(v_i^k, v_{-i})$ for $1 \leq k \leq K$, and $a_{K+1} := a_1$.

Cycle monotonicity is both necessary and sufficient for truthful mechanisms (again, see Lavi and Swamy [10]).

Theorem 1. *There exist payment functions $p_1, \ldots, p_n$ such that the mechanism $(f, p_1, \ldots, p_n)$ is truthful iff f is cycle monotone.*

It is possible to consider a simpler version of the cycle monotonicity condition, usually called *weak monotonicity*, if the valuation function domains are convex. This has been observed by Saks and Yu [37] (related results are due to [38, 36, 39]). However, we are concerned with non-convex valuation function domains. To see this, observe that the above valuation functions can be represented by step functions that map completion times in $\mathbb{R}_{>0}$ to $\mathbb{Q}_{\geq 0}$ with at most one jump discontinuity. A convex combination of two or more of these step functions may have more than one jump discontinuity, which implies non-convexity of the valuation function domains.

3. Incentive Compatible Mechanisms for $\mathbf{P|priv\{w_i, d_i\}, U_i|\sum w_i U_i}$

We will now consider the mechanism design setting $P|priv\{w_i, d_i\}, U_i|\sum w_i U_i$ described in Section 2. We will introduce properties of social choice functions (Section 3.1) and show that a subset of these properties is sufficient and each of them if necessary for truthful mechanisms (Section 3.2), i.e. these properties are equivalent to cycle monotonicity. Our motivation stems from the fact that, being a general condition for truthful mechanisms in non-convex valuation function domains, cycle monotonicity is not always easy to prove when aiming to construct a truthful mechanism for a specific problem setting. Our results, however, utilize knowledge about $P||\sum w_i U_i$ and are therefore easier to implement and verify. We will close this section by deriving the payment functions of the related mechanisms in Section 3.3.

3.1. Properties of Social Choice Functions

We refer to the first relevant property as *threshold monotonicity.* It requires the functions $w_i^{inf}(d)$ to be monotonically decreasing in d for all $i \in J$ and $v_{-i} \in V_{-i}$.

Definition 2 (Threshold monotonicity). *A social choice function f is called threshold monotone if*

$$\forall\, d < d' : \, w_i^{inf}(d) \geq w_i^{inf}(d')$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

We will refer to a social choice function as *due-date stable* if the following is true for all $i \in J$ and $v_{-i} \in V_{-i}$: If there exists a combination of weight w and due date d with a given threshold value that, when committed to the mechanism by agent i, results in job i being scheduled on time with respect to d with completion time C_i, then this threshold value remains constant for all due dates between C_i and d.

Definition 3 (Due-date stability). *A social choice function f is called due-date stable if*

$$\forall\, w, d, d' \in \mathbb{Q}_{\geq 0} \text{ with } d \geq d' \geq C_i(f(v_i^{w,d}, v_{-i})) :\, w_i^{inf}(d) = w_i^{inf}(d')$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

Hence, for any social choice function that is threshold monotone and due-date stable, the functions $w_i^{inf}(d)$ are step functions that are monotonically decreasing in d for all $i \in J$ and $v_{-i} \in V_{-i}$.

The third property, referred to as *weight monotonicity*, requires the sets $W_i^{d_i}$ to be connected for all $i \in J$ and $v_{-i} \in V_{-i}$. Moreover, the interval $W_i^{d_i}$ must not be bounded from above.

Definition 4 (Weight monotonicity). *A social choice function f is called weight monotone if*

$$\forall\, v_i^{w,d}, v_i^{w',d} \in V_i \text{ with } w < w' :\, C_i(f(v_i^{w,d}, v_{-i})) \leq d \Rightarrow C_i(f(v_i^{w',d}, v_{-i})) \leq d$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

Next, a social choice function is *late-on-time separating* if any due date that can be committed to the mechanism and that allows job i to be scheduled as a late job by the scheduling algorithm is strictly smaller than c_i^{late} for all $i \in J$ and $v_{-i} \in V_{-i}$.

Definition 5 (Late-on-time separability). *A social choice function f is called late-on-time separating if*

$$\forall\, d \in \mathbb{Q}_{\geq 0} \text{ with } w_i^{inf}(d) > 0 :\, d < c_i^{late}$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

Figure 2 illustrates some implications of Definitions 2–5. Consider a job $i \in J$. Because of Definitions 2 and 3, $w_i^{inf}(d_i)$ is a monotonically decreasing step function. Note that $w_i^{inf}(d_i)$ is equal to ∞ for all d_i smaller than t_i. Definition 4 is represented by shaded areas in the figure. The light gray area above the graph of $w_i^{inf}(d_i)$ represents all pairs (d_i, w_i) that result in a schedule with job i being scheduled as an on-time job if the corresponding valuation function

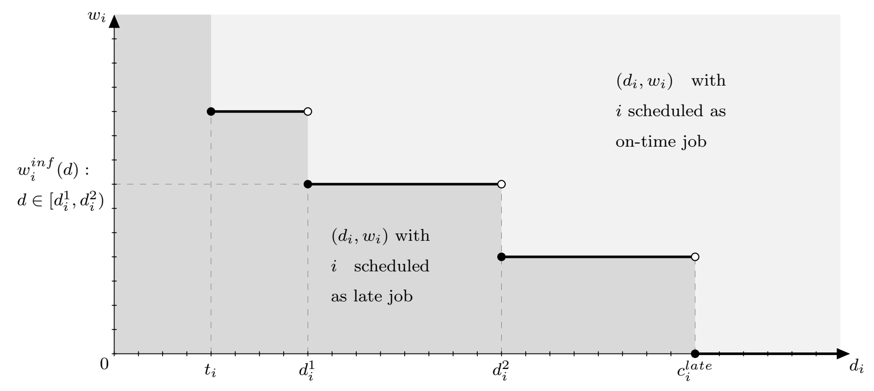

Figure 2: Plot of $w_i^{inf}(d_i)$

$v_i^{w_i,d_i}$ is committed to the mechanism by job-owner i. All pairs (d_i, w_i) in the dark gray area result in schedules with job i being scheduled as a late job. For a given d_i, the interval $W_i^{d_i}$ is defined by the connected light gray area above $w_i^{inf}(d_i)$. As a result of Definition 5, $w_i^{inf}(d_i)$ is equal to zero for $d_i \geq c_i^{late}$.

As we will prove in Section 3.2, each of the properties related to Definitions 2–5 is necessary and together they are sufficient for a social choice function to be cycle monotone. However, since there is an unlimited number of potential scheduling algorithms, we introduce an additional property, referred to as *threshold stability*, that is equivalent to Definitions 2 and 3 and may be easier to handle for specific algorithms. A simplified interpretation of this property is as follows: Let $i \in J$ and $v_{-i} \in V_{-i}$ be fixed. If there exist two due dates that result in different threshold values, then the completion time of the selected player i's job will always (i.e. independent of the weight) exceed the smaller one of the due dates when committing the larger due date.

Definition 6 (Threshold stability). *A social choice function f is called threshold stable if*

$$w_i^{inf}(d) > w_i^{inf}(d') \Rightarrow d < d' \wedge \forall\, w \in W_i^{d'} \,:\, C_i(f(v_i^{w,d'}, v_{-i})) > d \tag{2}$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

As indicated above, a social choice function is threshold monotone and due-date stable iff it is threshold stable.

Lemma 1. *Let f be a social choice function. Then*

$$f \textit{ is threshold monotone and due-date stable} \Leftrightarrow f \textit{ is threshold stable.}$$

Proof. The proof is rather simple, so that we restrict ourselves to a sketch.

"$\Rightarrow$": Assuming f to be threshold monotone but not threshold stable can easily be seen to contradict f being due-date stable.

"$\Leftarrow$": The fact that f is threshold monotone can easily be proven by contraposition of (2). Additionally, if f is not due-date stable but threshold monotone, this can easily be shown to contradict f being threshold stable. $\square$

3.2. Incentive Compatibility

We will prove that satisfying Definitions 2–5 at the same time is necessary and sufficient for incentive compatibility after having introduced a lemma that directly relates to the definition of cycle monotonicity, i.e. Definition 1. The message of this lemma is that there exists no sequence of valuation functions of a given player, such that (1) has two succeeding negative summands.

Lemma 2. *Let f be a social choice function satisfying Definitions 4 and 5. Furthermore, let $i \in J$ and $v_{-i} \in V_{-i}$ be fixed. There is no sequence $v_i^k, v_i^{k+1}, v_i^{k+2} \in V_i$ with*

$$v_i^k(a_k) - v_i^k(a_{k+1}) < 0 \; \wedge \; v_i^{k+1}(a_{k+1}) - v_i^{k+1}(a_{k+2}) < 0, \tag{3}$$

where $a_k := f(v_i^k, v_{-i})$.

Proof. Let $i \in J$ and $v_{-i} \in V_{-i}$ be arbitrary but fixed. For a given k, denote the due date and the weight that correspond to v_i^k by d_i^k and w_i^k, respectively. First, note that the value of a valuation function $v_i \in V_i$ is non-positive by definition. Additionally, if $d_i^k < C_i(a_k)$ and $d_i^k < C_i(a_{k+1})$, then $v_i^k(a_k) - v_i^k(a_{k+1}) = -w_i^k - (-w_i^k) = 0$. Hence, $v_i^k(a_k) - v_i^k(a_{k+1})$ is negative iff $d_i^k < C_i(a_k)$ and $d_i^k \geq C_i(a_{k+1})$ with $w_i^{inf}(d_i^k) > 0$ (because of Definition 4) and we must have $C_i(a_k) \geq c_i^{late} > d_i^k \geq C_i(a_{k+1})$, because f satisfies Definition 5. Hence, job i is scheduled as an on-time job if client i reports v_i^{k+1}, i.e. $v_i^{k+1}(a_{k+1}) = 0$, as otherwise $C_i(a_{k+1}) \geq c_i^{late}$. Thus, we must have $v_i^{k+1}(a_{k+2}) > 0$ in order for (3) to be true. This, however, is not possible by definition of v_i^{k+1}, which concludes the proof. $\square$

We can now present our main result.

Theorem 2. *A social choice function f is cycle monotone iff it satisfies Definitions 2–5, i.e. iff f is threshold monotone, weight monotone, late-on-time separating, and due-date stable.*

Proof. We will first show that f is cycle monotone if it satisfies Definitions 2–5. To do so, let $i \in J$ and $v_{-i} \in V_{-i}$ be arbitrary but fixed. Furthermore, consider $K \in \mathbb{N}$ valuation functions $v_i^1, \dots, v_i^K \in V_i$. Denote the schedule returned by f in case of reporting v_i^k by a_k, i.e. $a_k := f(v_i^k, v_{-i})$, for $k \in \{1, \dots, K\}$. Moreover, for all $k \in \{1, \dots, K\}$, denote the due date and the weight that correspond to v_i^k by d_i^k and w_i^k, respectively.

Obviously, inequality (1) holds if all summands are non-negative. If at least one summand is negative, we can partition the indices $1, \ldots, K$ into disjoint subsequences, such that each subsequence consists of consecutive indices $k', \ldots, k''-1$ (with $K + x \equiv x$ for $1 \leq x < K$) with

$$v_i^{k'}(a_{k'}) - v_i^{k'}(a_{k'+1}) < 0 \tag{4}$$

and such that

$$\sum_{j=k'}^{k''-1} \Big(v_i^j(a_j) - v_i^j(a_{j+1}) \Big) \tag{5}$$

has exactly one negative summand. Using Lemma 2, we can deduce that each subsequence consists of at least two consecutive indices.

Let $k', \ldots, k''-1$ be an arbitrary of the above subsequences as illustrated in Figure 3. Recall that the value of a valuation function is non-positive by definition. Thus, the only

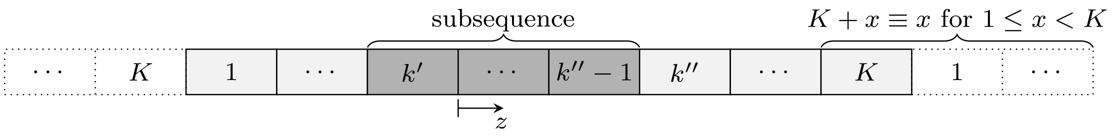

Figure 3: An arbitrary subsequence

negative summand in (5) has value $-w_i^{k'} < 0$.

We will prove that there is at least one summand in (5) with a positive value greater than $w_i^{k'}$. To do so, we will identify an element z of the subsequence (see Figure 3) that features the corresponding difference $v_i^{z-1}(a_{z-1}) - v_i^{z-1}(a_z)$ to be larger than $w_i^{k'}$. Our argumentation is structured as follows. First, we provide some auxiliary statements. Next, we present a characterization of the index z and show that $z \neq k'+1$. Hereafter, we prove that $w_i^{z-1} > w_i^{k'}$. Finally, we conclude by making a case differentiation (job i is scheduled as a late job or as an on-time job in a_z) and proving that $v_i^{z-1}(a_{z-1}) - v_i^{z-1}(a_z) = w_i^{z-1} > w_i^{k'}$ for both cases.

Auxiliary statements: Because of (4) we have $C_i(a_{k'+1}) \leq d_i^{k'}$. Furthermore, $w_i^{inf}(d_i^{k'}) > 0$, because otherwise $v_i^{k'}(a_{k'}) = 0$ in contradiction to (4) because of Definition 4. Define $D_i(d) := \{d' \in \mathbb{Q}_{\geq 0} \mid w_i^{inf}(d') = w_i^{inf}(d)\}$. Hence, $w_i^{inf}(d) > 0$ for all $d \in D_i(d_i^{k'})$. Moreover, job i is scheduled as an on-time job in $a_{k'+1}$, i.e. $C_i(a_{k'+1}) \leq d_i^{k'+1}$, because Definition 5 requires $C_i(a_{k'+1}) \leq d_i^{k'} < c_i^{late}$.

Characterization of z: Denote by z the first ("minimal") index occurring within $k'+1, \ldots, k''$ (see Figure 3) such that job i is either scheduled as a late job in a_z or as an on-time job in a_z with $d_i^z > d$ for all $d \in D_i(d_i^{k'})$. Note that $z \neq k'+1$ because i is not late in $a_{k'+1}$ and, therefore, $z = k'+1$ would require $d_i^{k'+1} > d$ for all $d \in D_i(d_i^{k'})$. Particularly, we would have $d_i^{k'+1} > d_i^{k'}$ and $w_i^{inf}(d_i^{k'}) \neq w_i^{inf}(d_i^{k'+1})$, which results in $w_i^{inf}(d_i^{k'}) > w_i^{inf}(d_i^{k'+1})$ by using Definition 2 and thus induces $C_i(a_{k'+1}) > d_i^{k'}$ due to Lemma 1 and Definition 6, which contradicts (4) as mentioned above.

$w_i^{z-1} > w_i^{k'}$: Hence, job i is scheduled as an on-time job in a_{z-1} with a completion time less or equal to $\sup D_i(d_i^{k'})$ and there exists a $d \in D_i(d_i^{k'})$ with $d_i^{z-1} \leq d$, because otherwise z would not be minimal. Thus, we have $w_i^{inf}(d_i^{z-1}) > 0$ because of Definition 2 and $w_i^{z-1} \geq w_i^{min}(d)$ for all $d \in D_i(d_i^{k'})$. Furthermore, since Definition 4 is satisfied, we have $w_i^{k'} < w_i^{min}(d)$ for all $d \in D_i(d_i^{k'})$. Therefore, $w_i^{z-1} > w_i^{k'}$.

Case differentiation: We now have to consider two cases. First, assume that job i is scheduled as a late job in a_z. Then $C_i(a_z) \geq c_i^{late}$. Moreover, as shown above, $w_i^{inf}(d_i^{z-1}) > 0$. Therefore, together with Definition 5, $v_i^{z-1}(a_{z-1}) - v_i^{z-1}(a_z) = 0 - (-w_i^{z-1}) > w_i^{k'}$. Second, assume that job i is scheduled as an on-time job in a_z and $d_i^z > d$ for all $d \in D_i(d_i^{k'})$. Then, in analogy to the above deliberations based on Definition 2, we have $w_i^{inf}(d_i^z) < w_i^{inf}(d_i^{k'})$. Similarly, since there is at least one $d \in D_i(d_i^{k'})$ with $d_i^{z-1} \leq d$, we know that $w_i^{inf}(d_i^{z-1}) \geq w_i^{inf}(d_i^{k'})$. It follows from Lemma 1 and Definition 6 that $C_i(a_z) > d_i^{z-1}$ and therefore $v_i^{z-1}(a_{z-1}) - v_i^{z-1}(a_z) = 0 - (-w_i^{z-1}) > w_i^{k'}$. Hence - as claimed above - in both cases there exists a summand with a positive value greater than $w_i^{k'}$ in (5).

We will now prove that cycle monotonicity implies the following four properties: threshold monotonicity, weight monotonicity, late-on-time separability, and due-date stability. In order to do so, we will show that, in terms of Theorem 1, there exists a cycle of negative length if at least one of these properties is negated. For each example, we will assume $i \in J$ and $v_{-i} \in V_{-i}$ to be fixed and consider a sequence of $K \in \{2, 3\}$ valuation functions $v_i^1, \ldots, v_i^K$. As above, for all $k \in \{1, \ldots, K\}$, we denote the corresponding weights and due dates by w_i^k and d_i^k, respectively. Moreover, we define $a_k := f(v_i^k, v_{-i})$, $k \in \{1, \ldots, K\}$.

First, assume that f is not threshold monotone. Then there exist due dates $d < d'$ with $w_i^{inf}(d) < w_i^{inf}(d')$ and a weight $w \in W_i^d$ with $w_i^{inf}(d) \leq w < w_i^{inf}(d')$.

Chose $w' \in \mathbb{Q}_{\geq 0}$, such that $w_i^{inf}(d) \leq w < w' < w_i^{inf}(d')$. Furthermore, set $K = 2$,

$$\begin{aligned} w_i^1 &= w, & d_i^1 &= d, \\ w_i^2 &= w', \text{ and} & d_i^2 &= d'. \end{aligned}$$

Then the resulting summands of (1) are

$$\begin{aligned} v_i^1(a_1) - v_i^1(a_2) &= 0 - (-w) = w, \text{ and} \\ v_i^2(a_2) - v_i^2(a_1) &= -w' - 0 = -w'. \end{aligned}$$

Therefore, (1) is violated and there exists a cycle of negative length.

Now, assume that f is not weight monotone. Then there exist $v_i^{w,d}, v_i^{w',d} \in V_i$ with $w < w'$ and $C_i(f(v_i^{w,d}, v_{-i})) \leq d$, such that $C_i(f(v_i^{w',d}, v_{-i})) > d$.

Set $K = 2$,

$$\begin{aligned} w_i^1 &= w, & d_i^1 &= d, \\ w_i^2 &= w', \text{ and} & d_i^2 &= d. \end{aligned}$$

Then

$$\begin{aligned} v_i^1(a_1) - v_i^1(a_2) &= 0 - (-w) = w, \text{ and} \\ v_i^2(a_2) - v_i^2(a_1) &= -w' - 0 = -w', \end{aligned}$$

so that there exists a cycle of negative length.

Next, assume that f is not late-on-time separating. Then there exist values $w, d, d' \in \mathbb{Q}_{\geq 0}$, such that $w_i^{inf}(d') > 0$, $C_i(f(v_i^{w,d}, v_{-i})) > d$, and $C_i(f(v_i^{w,d}, v_{-i})) \leq d'$.

Set $K = 2$,

$$\begin{aligned} w_i^1 &= w, & d_i^1 &= d, \\ w_i^2 &= \frac{1}{2} w_i^{inf}(d'), \text{ and} & d_i^2 &= d'. \end{aligned}$$

Then

$$\begin{aligned} v_i^1(a_1) - v_i^1(a_2) &= -w - \overbrace{v_i^1(a_2)}^{\in\{-w,0\}} \leq 0, \text{ and} \\ v_i^2(a_2) - v_i^2(a_1) &= -w_i^2 - 0 = -\frac{1}{2} w_i^{inf}(d') < 0. \end{aligned}$$

Hence, there exists a cycle of negative length.

Finally, assume that f is not due-date stable. Let $w, d \in \mathbb{Q}_{\geq 0}$ with $v_i^{w,d} \in V_i$ and $C_i(f(v_i^{w,d}, v_{-i})) \leq d$. If f is not due-date stable, there exists a d' with $d \geq d' \geq C_i(f(v_i^{w,d}, v_{-i}))$ and $w_i^{inf}(d) \neq w_i^{inf}(d')$.

Assume $w_i^{inf}(d) < w_i^{inf}(d')$. Chose a weight $\hat{w} \in W_i^d$ with $w_i^{inf}(d) < \hat{w} < w_i^{inf}(d')$ and a weight $w' \in \mathbb{Q}_{\geq 0}$, such that $w_i^{inf}(d') > w' > \hat{w} > w_i^{inf}(d)$. Set $K = 3$,

$$\begin{aligned} w_i^1 &= \hat{w}, & d_i^1 &= d, \\ w_i^2 &= w', & d_i^2 &= d', \\ w_i^3 &= w, \text{ and} & d_i^3 &= d. \end{aligned}$$

Then the resulting summands of (1) are

$$\begin{aligned} v_i^1(a_1) - v_i^1(a_2) &= 0 - \overbrace{v_i^1(a_2)}^{\in\{-\hat{w},0\}} \leq \hat{w}, \\ v_i^2(a_2) - v_i^2(a_3) &= -w' - 0 = -w', \text{ and} \\ v_i^3(a_3) - v_i^3(a_1) &= 0 - 0 = 0, \end{aligned}$$

so that there exists a cycle of negative length.

If $w_i^{inf}(d) > w_i^{inf}(d')$, f is not threshold monotone, so that we refer to the corresponding part of this proof for a cycle of negative length. □

3.3. Payment Functions

If a social choice function f is cycle monotone, Theorem 1 guarantees the existence of payment functions $p_1, \ldots, p_n$, such that the mechanism $(f, p_1, \ldots, p_n)$ is incentive compatible. Nisan [9] proves that payment functions of a client i are unique up to an additive constant. However, the construction of payment functions in the proof of Theorem 1 is based on finding shortest paths in a graph with exponentially many nodes, so that it is not clear if it is possible to efficiently compute the payments in that way. Theorem 3 therefore presents a representation of payment functions that utilize knowledge about $P||\sum w_i U_i$ and that are easier to analyze with respect to the complexity of their computation.

Theorem 3. *Let f be a social choice function satisfying Definitions 2–5 and let $i \in J$, $v_i^{w,d} \in V_i$, and $v_{-i} \in V_{-i}$. Define*

$$p_i(v_i^{w,d}, v_{-i}) := \begin{cases} -w_i^{inf}(d) & \textit{if } C_i(f(v_i^{w,d}, v_{-i})) \leq d, \\ 0 & \textit{else.} \end{cases}$$

The mechanism $(f, p_1, \ldots, p_n)$ is incentive compatible.

Proof. Denote the due date and weight that correspond to the true valuation function v_i^t of client $i \in J$ by d_i^t and w_i^t, respectively. We will show that $u_i(v_i^t, v_{-i}) = v_i^t(f(v_i^t, v_{-i})) +$

$p_i(v_i^t, v_{-i}) \geq v_i^t(f(v_i, v_{-i})) + p_i(v_i, v_{-i}) = u_i(v_i, v_{-i})$ for all $i \in J$, all $v_{-i} \in V_{-i}$, and all $v_i \in V_i$. To do so, we will consider two cases.

First, assume that $C_i(f(v_i^t, v_{-i})) > d_i^t$, i.e. job $i \in J$ is scheduled as a late job by the scheduling algorithm if the corresponding job-owner reports the true valuation function. Hence, $u_i(v_i^t, v_{-i}) = -w_i^t$. Obviously, there exists no $v_i \in V_i$ with $C_i(f(v_i, v_{-i})) > d_i^t$ that results in a better utility function value for client i, because the value of the client's true valuation function does not increase when compared to $u_i(v_i^t, v_{-i})$ (job i remains late with respect to d_i^t) and the value of the payment function is non-positive. Hence, we will now consider an arbitrary $v_i \in V_i$ with $C_i(f(v_i, v_{-i})) \leq d_i^t$. By applying Lemma 1 as well as Definitions 2 and 6 we can deduce that client i must report a valuation function with a due date d such that there exists a $\hat{d} \in D_i(d_i^t)$ with $d \leq \hat{d}$, where $D_i(d) := \{d' \in \mathbb{Q}_{\geq 0} \mid w_i^{inf}(d') = w_i^{inf}(d)\}$. Hence, $w_i^{inf}(d_i^t) \leq w_i^{inf}(d)$. Furthermore, due to Definition 4 and because $C_i(f(v_i^t, v_{-i})) > d_i^t$, we have $w_i^{inf}(d_i^t) > 0$. Hence, by Definition 5, $d_i^t < c_i^{late}$, and therefore $C_i(f(v_i, v_{-i})) < c_i^{late}$, so that the weight w of the reported valuation function v_i must be at least $w_i^{min}(d)$. Additionally, note that $w_i^t \leq w_i^{inf}(d_i^t)$ because of Definition 4 and the fact that job i is scheduled as a late job if client i reports v_i^t. Hence, by using Definition 2 once more, we get $u_i(v_i^{w,d}, v_{-i}) = v_i^t(f(v_i^{w,d}, v_{-i})) + p_i(v_i^{w,d}, v_{-i}) = 0 - w_i^{inf}(d) \leq -w_i^t = u_i(v_i^t, v_{-i})$.

Second, consider the case $C_i(f(v_i^t, v_{-i})) \leq d_i^t$. Then $u_i(v_i^t, v_{-i}) = v_i^t(f(v_i^t, v_{-i})) + p_i(v_i^t, v_{-i}) = 0 - w_i^{inf}(d_i^t)$. There exist two potential options for client i to try to increase her utility function value. First, client i can report a larger due date in order to obtain a better payment function value while still being scheduled as an on-time job. Then, however, Lemma 1, Definition 2 and Definition 6 guarantee that the completion time of job i is no longer less or equal to d_i^t and the utility function value decreases when compared to $u_i(v_i^t, v_{-i})$. Second, client i can report a valuation function $v_i \in V_i$, such that job i is scheduled as a late job. Then, we get $u_i(v_i, v_{-i}) = v_i^t(f(v_i, v_{-i})) + p_i(v_i, v_{-i}) = -w_i^t + 0 \leq -w_i^{inf}(d_i^t) = u_i(v_i^t, v_{-i})$ if $d_i^t < c_i^{late}$ and, as a consequence of Definition 5, $u_i(v_i, v_{-i}) = v_i^t(f(v_i, v_{-i})) + p_i(v_i, v_{-i}) \leq 0 = -w_i^{inf}(d_i^t) = u_i(v_i^t, v_{-i})$ if $d_i^t \geq c_i^{late}$. Hence, in any case, $u_i(v_i, v_{-i}) \leq u_i(v_i^t, v_{-i})$. □

4. Applying Our Results to an Example Algorithm for $\mathbf{1|priv\{w_i, d_i\}, U_i|\sum w_i U_i}$

In this Section, we will illustrate the use of Theorem 2 by analyzing an example algorithm that has recently been proposed in the literature. More specifically, we will consider a heuristic approach introduced by Kovalyov and Pesch [2], who consider a mechanism design setting with risk averse job agents and who restrict themselves to the case of one machine, i.e. to the scheduling problem $1||\sum w_i U_i$. Their algorithm has some degrees of freedom, that we constrain by using concrete sub-algorithms. The resulting heuristic is presented in Algorithm 1. It is referred to as GreedyWeight [2]. It makes use of the Earliest Due Date (EDD) order of jobs,

Algorithm 1 GreedyWeight

```
Step 1  Initialize a list $L$ by sorting all jobs according to their weights in non-increasing order, breaking ties
        with respect to their indices (smaller indices first). Let $L[j]$ denote the $j$-th element of $L$. Initialize an
        empty set of on-time jobs, $S^{early}$, an empty set of late jobs, $S^{late}$, and an empty set $S^{temp}$. Set $k := 1$.
Step 2  If $k = n + 1$, then go to Step 3. Else, set $S^{temp} := S^{early} \cup \{L[k]\}$. Construct an EDD sequence $S^{EDD}$
        of the jobs in the set $S^{temp}$, breaking ties according to the job-indices (smaller indices first). If all jobs
        of sequence $S^{EDD}$ are on time, then set $S^{early} := S^{temp}$. If at least one job of the sequence $S^{EDD}$ is
        late, then set $S^{late} := S^{late} \cup \{L[k]\}$. Set $k := k + 1$ and repeat Step 2.
Step 3  Output a schedule $S^*$, which is constructed as follows. On-time jobs are scheduled according to the
        EDD sequence of $S^{early}$, breaking ties as described in Step 2. Afterwards, late jobs of the set $S^{late}$ are
        scheduled according to their SPT sequence.
```

which refers to an ordering of jobs such that a job with smaller due date appears earlier than a job with larger due date. Moreover, it uses the Shortest Processing Time (SPT) order of jobs, which is such that a job with smaller processing time appears earlier than a job with larger processing time. Kovalyov and Pesch [2] prove GreedyWeight to be an $(n-1)$-approximation algorithm and show that it can be implemented to run in $\mathcal{O}(n^2)$.

In order to apply Theorem 2, we will prove that the social choice function that is established by Algorithm 1, which we will refer to by f^1 throughout the remainder of this section, satisfies Definitions 4, 5 and 6. We will begin with Definition 4.

Proposition 1. *f^1 is weight monotone because*

$$C_i(f(v_i^{w,d}, v_{-i})) = C_i(f(v_i^{w',d}, v_{-i})) \quad \forall\, v_i^{w,d}, v_i^{w',d} \in V_i,\ w, w' \geq w_i^{min}(d),\ w \neq w'$$

for all $i \in J$ and $v_{-i} \in V_{-i}$.

Proof. Let $i \in J$ and $v_{-i} \in V_{-i}$ be arbitrary but fixed. Assume there are weights $w, w' \geq w_i^{min}(d)$ with $C_i(f(v_i^{w,d}, v_{-i})) \neq C_i(f(v_i^{w',d}, v_{-i}))$ and w.l.o.g. $w < w'$. Note that these weights are solely considered in Step 1 of Algorithm 1. Define $a_w := f(v_i^{w,d}, v_{-i})$ and $a_{w'} := f(v_i^{w',d}, v_{-i})$. In the following, we will consider two cases.

We will show that all jobs that are on time in a_w are on time in $a_{w'}$ as well, and that all jobs that are late in a_w are also late in $a_{w'}$. Hence, the EDD sequence of on-time jobs is identical in a_w and $a_{w'}$, so that $C_i(a_w) = C_i(a_{w'})$, which is a contradiction. To do so, denote the sorted lists of jobs resulting from Step 1 of Algorithm 1 with respect to the weight of job i by $L(w)$ and $L(w')$, respectively (see Figure 4). Obviously, the set of jobs Γ_1 that precede i in $L(w')$ are considered in the same iteration of Step 2 of Algorithm 1 in the process of determining a_w and $a_{w'}$. Hence, these jobs cannot be on time in a_w and late in $a_{w'}$ or vice versa. Now, consider the set of jobs Γ_2 that precede i in $L(w)$ and succeed i in $L(w')$. Assume the non-trivial case $\Gamma_2 \neq \emptyset$. Obviously, if one of these jobs is late in a_w, it must also be late in $a_{w'}$, because i is

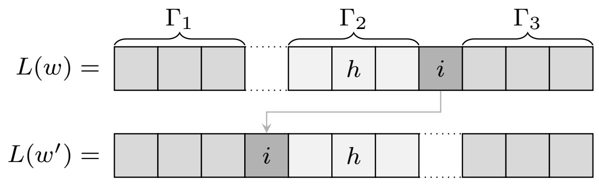

Figure 4: Sorted lists of jobs $L(w)$ and $L(w')$ with $w < w'$

considered in an earlier iteration of Step 2 of Algorithm 1 when determining $a_{w'}$ than when determining a_w. Similarly, if one of these jobs, say h, is on time in a_w, it is possible to finish both i and h on time when applying Algorithm 1. Hence, the fact that job i is considered in an earlier iteration of Step 2 when determining $a_{w'}$ cannot prevent h from being scheduled on time in $a_{w'}$ as well. We are left with the set of jobs Γ_3 that succeed i in $L(w)$. As none of the jobs of Γ_1 and Γ_2 have switched their status (from late to on-time or vice versa) in a_w and $a_{w'}$, the same must hold for all jobs in Γ_3. Hence, as mentioned before, the EDD sequence of on-time jobs is identical in a_w and $a_{w'}$, so that $C_i(a_w) = C_i(a_{w'})$, which is a contradiction.

□

We will now turn our attention to Definition 5.

Proposition 2. *f^1 is late-on-time separating.*

Proof. Let $i \in J$ and $v_{-i} \in V_{-i}$ be arbitrary but fixed. Note that all $v_i \in V_i$ that result in a schedule with job i being scheduled as a late job feature the same completion time c_i^{late}. This is easy to see, because Algorithm 1 schedules all late jobs after the on-time jobs and the order of late jobs is independent of their weights and due dates. Furthermore, as specified in Proposition 1, all $w \in \mathbb{Q}_{\geq 0}$ that result in job i being scheduled as an on-time job for a given $d \in \mathbb{Q}_{\geq 0}$ feature the same completion time of job i. Denote this completion time by $c_i^{on}(d)$. Obviously, if $w_i^{inf}(d) > 0$ for a given $d \in \mathbb{Q}_{\geq 0}$, we must have $c_i^{late} > c_i^{on}(d)$, and there exists no $w \in \mathbb{Q}_{\geq 0}$ that results in i being scheduled as an on-time job with completion time c_i^{late}. Thus, for all $w \in \mathbb{Q}_{\geq 0}$ with $C_i(f(v_i^{w,d}, v_{-i})) = c_i^{late}$, we have $c_i^{late} > d$ and, therefore, f^1 is late-on-time separating. □

We are left with having to analyze f^1 in the context of Definition 6.

Proposition 3. *f^1 is threshold stable.*

Proof. Let $i \in J$ and $v_{-i} \in V_{-i}$ be arbitrary but fixed. Furthermore, let $w_i^{inf}(d) > w_i^{inf}(d')$ and choose $w' \in W_i^{d'}$ such that $w' < w_i^{inf}(d)$. Denote the iteration in which job i is considered in Step 2 of Algorithm 1 if client i reports weight w' by k, and denote the temporary EDD schedules that result from this iteration by $a_k^{d'}$ if i additionally reports d' and a_k^d in case of reporting d. Denote the set of jobs that are known to be on time after iteration $k-1$ of Step 2 by A. Moreover, given a temporary EDD schedule a^{temp}, denote the completion time of a job j included in this schedule by $C_j(a^{temp})$.

Obviously, job i is scheduled as a late job if client i reports $v_i^{w',d}$ and as an on-time job if i reports $v_i^{w',d'}$.

Assume $d > d'$ as illustrated in Figure 5. Then the only job that can potentially be late

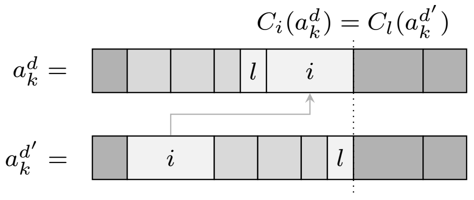

Figure 5: Temporary schedules a_k^d and $a_k^{d'}$ with $d > d'$

in a_k^d is job i, because every other job is completed at the same time or earlier than it is completed in $a_k^{d'}$, where all jobs are on time. Denote the job that directly precedes i in a_k^d by l. Furthermore, denote l's reported valuation function by $v_l^{\hat{w},\hat{d}}$. Obviously, we have $d \geq \hat{d}$ (EDD sequence of jobs). Moreover, $C_l(a_k^{d'}) \leq \hat{d}$ and $C_i(a_k^d) = C_l(a_k^{d'})$. Summing up, we have $C_i(a_k^d) = C_l(a_k^{d'}) \leq \hat{d} \leq d$. Therefore, i is on time when reporting $v_i^{w',d}$, which is a contradiction. Hence, we must have $d < d'$.

Now assume there is at least one job $j \in A$ with reported valuation function $v_j^{\bar{w},\bar{d}}$ and $d \leq \bar{d} < C_j(a_k^d)$ as illustrated in Figure 6. Since all jobs of the set $A \cup \{i\}$ must be on time in $a_k^{d'}$ and because the due date of job i is the only one that differs when generating the schedules, job i must succeed job j in $a_k^{d'}$. Therefore, $C_i(a_k^{d'}) \geq C_j(a_k^{d'}) + t_i$. Now note that the jobs of the set A that proceed j in a_k^d proceed j in $a_k^{d'}$ as well. Hence $C_j(a_k^d) = C_j(a_k^{d'}) + t_i$. Therefore, $C_i(f(v_i^{w',d'}, v_{-i})) \geq C_i(a_k^{d'}) \geq C_j(a_k^{d'}) + t_i = C_j(a_k^d) > \bar{d} \geq d$.

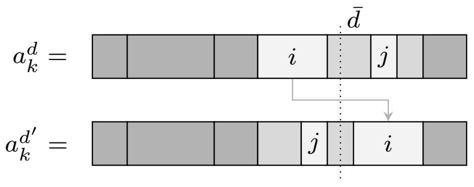

Figure 6: Temporary schedules a_k^d and $a_k^{d'}$ with $d < d'$

Similarly, if there is no job $j \in A$ with $d \leq \bar{d} < C_j(a_k^d)$, we must have $C_i(a_k^d) > d$. Additionally, increasing the due date to d' does not result in an earlier position in the schedule. Therefore, $C_i(f(v_i^{w',d'}, v_{-i})) \geq C_i(a_k^{d'}) \geq C_i(a_k^d) > d$.

From the above deliberations we may conclude: $w_i^{inf}(d) > w_i^{inf}(d') \Rightarrow d < d' \wedge \forall w \in W_i^{d'}, w < w_i^{inf}(d) : C_i(f(v_i^{w,d'}, v_{-i})) > d$. When additionally considering Proposition 1, we immediately get (2). $\square$

By applying Lemma 1 and Theorem 2, we can now conclude:

Corollary 1. *f^1 is cycle monotone.*

In order to obtain a truthful polynomial time mechanism based on Algorithm 1, we need to show how to compute the payments of Theorem 3 in polynomial time, i.e. we need to analyze the time complexity of having to compute $w_i^{inf}(d)$ for a client $i \in J$, who has committed a valuation function $v_i^{w,d} \in V_i$ to the mechanism based on f^1. Note that, given the vector of valuation functions committed to the mechanism, it is easy to check in polynomial time if $w_i^{inf}(d) = 0$ or $w_i^{inf}(d) = \infty$. If neither is the case, then there must exist a player $j \in J$, $j \neq i$, who has committed the valuation function $v_j^{\hat{w},\hat{d}} \in V_j$, such that $w_i^{inf}(d) = \hat{w}$. Hence, we can perform a binary search on all reported weights to find $w_i^{inf}(d)$. This results in $\mathcal{O}(\log n)$ calls of Algorithm 1, which obviously is polynomial.

5. Summary and Future Research

This paper has studied the problem of designing polynomial time truthful mechanisms for scheduling two-parameter job agents on parallel identical machines to minimize the weighted number of late jobs. The agents are assumed to have private information on their weights and due dates, while processing times are publicly known. We have contributed to the literature by deriving a set of conditions that is equivalent to cycle monotonicity, which is a general condition for incentive compatible mechanisms in non-convex valuation function domains. Our results have utilized knowledge about the relevant scheduling problems, so that the resulting conditions are easier to implement and verify than the general condition of cycle monotonicity. We have illustrated this fact by making use of our results to prove incentive compatibility of a mechanism that is established by an example algorithm that has recently been proposed by Kovalyov and Pesch [2] for the case of one machine.

Future research may focus on several issues. With respect to the scheduling problems considered in this paper, it may be interesting to investigate if other heuristics that have been proposed in the literature are suitable for the construction of truthful polynomial time mechanisms by making use of our results. Further work may also analyze the implications of assuming the agents to be risk averse or risk seeking. Furthermore, there remain plenty of interesting scheduling problems that have not yet been analyzed in the context of algorithmic mechanism design.

Acknowledgement

This work has been supported by the German Science Foundation (DFG) through the grant "Scheduling mechanisms for rail mounted gantries with regard to crane interdependencies" (PE 514/22-1).

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, 242–252, 2000.

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

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

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

[5] D. Kress, S. Meiswinkel, E. Pesch, Mechanism Design for Machine Scheduling Problems – Classification and Literature Overview, Working Paper, 2017.

[6] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, The Journal of Finance 16 (1) (1961) 8–37.

[7] E. H. Clarke, Multipart Pricing of Public Goods, Public Choice 11 (1) (1971) 17–33.

[8] T. Groves, Incentives in teams, Econometrica 41 (4) (1973) 617–631.

[9] N. Nisan, Introduction to mechanism design (for computer scientists), in: N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani (Eds.), Algorithmic Game Theory, Cambridge University Press, Cambridge, 209–241, 2007.

[10] R. Lavi, C. Swamy, Truthful mechanism design for multidimensional scheduling via cycle monotonicity, Games and Economic Behavior 67 (1) (2009) 99–124.

[11] G. Christodoulou, E. Koutsoupias, Mechanism Design for Scheduling, Bulletin of the EATCS 97 (2009) 40–59.

[12] N. Nisan, A. Ronen, Algorithmic mechanism design, in: Proceedings of the 31st annual ACM Symposium on Theory of Computing (STOC'99), ACM, New York, 129–140, 1999.

[13] A. Archer, É.. Tardos, Truthful mechanisms for one-parameter agents, in: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01), IEEE, 482–491, 2001.

[14] G. Christodoulou, E. Koutsoupias, A. Vidali, A lower bound for scheduling mechanisms, Algorithmica 55 (4) (2009) 729–740.

[15] E. Koutsoupias, A. Vidali, A lower bound of 1+ φ for truthful scheduling mechanisms, Algorithmica 66 (1) (2013) 211–223.

[16] E. Angel, E. Bampis, F. Pascual, A.-A. Tchetgnia, On truthfulness and approximation for scheduling selfish tasks, Journal of Scheduling 12 (5) (2009) 437–445.

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

[18] E. Angel, E. Bampis, N. Thibault, Randomized truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science 414 (1) (2012) 1–8.

[19] V. Auletta, R. De Prisco, P. Penna, P. Persiano, How to route and tax selfish unsplittable traffic, in: Proceedings of the 16th annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'04), ACM, New York, 196–205, 2004.

[20] G. Christodoulou, L. Gourvès, F. Pascual, Scheduling selfish tasks: about the performance of truthful algorithms, in: G. Lin (Ed.), Computing and Combinatorics, Springer, Berlin, 187–197, 2007.

[21] J. Duives, B. Heydenreich, D. Mishra, R. Müller, M. Uetz, On optimal mechanism design for a sequencing problem, Journal of Scheduling 18 (1) (2015) 45–59.

[22] R. Hoeksma, M. Uetz, Two dimensional optimal mechanism design for a sequencing problem, in: M. Goemans, J. Correa (Eds.), Integer Programming and Combinatorial Optimization, Springer, Berlin, 242–253, 2013.

[23] R. Hain, M. Mitra, Simple sequencing problems with interdependent costs, Games and Economic Behavior 48 (2) (2004) 271–291.

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

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

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

[27] M. Mitra, Incomplete information and multiple machine queueing problems, European Journal of Operational Research 165 (1) (2005) 251–266.

[28] B. Heydenreich, R. Müller, M. Uetz, Mechanism design for decentralized online machine scheduling, Operations Research 58 (2) (2010) 445–457.

[29] R. Porter, Mechanism design for online real-time scheduling, in: Proceedings of the 5th ACM Conference on Electronic Commerce (EC'04), ACM, New York, 61–70, 2004.

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

[31] M. R. Garey, D. S. Johnson, Computers and Intractability - A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

[32] M. R. Garey, D. S. Johnson, "Strong" NP-Completeness Results: Motivation, Examples, and Implications, Journal of the ACM 25 (3) (1978) 499–508.

[33] R. M. Karp, Reducibility among combinatorial problems, in: R. E. Miller, J. W. Thatcher, J. D. Bohlinger (Eds.), Complexity of Computer Computations, Plenum Press, New York, 85–103, 1972.

[34] 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.

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

[36] R. Lavi, A. Mu'alem, N. Nisan, Towards a characterization of truthful combinatorial auctions, in: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03), IEEE, 574–583, 2003.

[37] M. Saks, L. Yu, Weak monotonicity suffices for truthfulness on convex domains, in: Proceedings of the 6th ACM conference on Electronic Commerce (EC'05), ACM, New York, 286–293, 2005.

[38] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, A. Sen, Weak monotonicity characterizes deterministic dominant-strategy implementation, Econometrica 74 (4) (2006) 1109–1132.

[39] K. Roberts, The characterization of implementable choice rules, in: J.-J. Laffont (Ed.), Aggregation and Revelation of Preferences, North-Holland, Amsterdam, 321–348, 1979.