Proceedings of the 16th Conference on Computer Science and Intelligence Systems pp. 201–209
DOI: 10.15439/2021F36
ISSN 2300-5963 ACSIS, Vol. 25

# Minimizing Tardiness in a Scheduling Environment with Jobs' Hierarchy

Michal Sinai and Tami Tamir
School of Computer Science
The Interdisciplinary Center
Herzliya, Israel
Emails: michal.sinai@post.idc.ac.il, tami@idc.ac.il

***Abstract*—In many scheduling environments, some jobs have higher priority than others. Such scenarios are theoretically modelled by associating jobs with weights, or by having precedence constraints that limit jobs' processing order. In this paper we define and consider a new model, motivated by real-life behaviour, in which the priority among jobs is defined by a *dominance hierarchy*. Specifically, the jobs are arranged in hierarchy levels, and high ranking jobs are ready to accept only outcomes in which the service they receive is better than the service of subordinate jobs. We first define the model and the set of feasible schedules formally. We then consider two classical problems: minimizing the maximal tardiness and minimizing the number of tardy jobs. We provide optimal algorithms or hardness proofs for these problems, distinguishing between a global objective function and a multi-criteria objective.**

## I. Introduction

JOB Scheduling problems are considered to be a fundamental and well studied field in theoretical computer science. The study of the combinatorial optimization problems induced by various scheduling environments is motivated by numerous real-life applications arising in production planning, traffic control, cloud computing services, and many more. A typical scheduling problem instance involves assigning a set of $n$ independent jobs on $m$ parallel machines in a way that optimally utilizes the machines and achieves high quality of service for the jobs. These objectives are mathematically modelled by minimizing a predefined objective function, such as the makespan (maximal completion time of some job), total completion time, lateness, etc. We refer to [18] for a comprehensive survey of various models of scheduling problems.

In many scheduling environments, the jobs are not treated in a fair way. Naturally, some jobs have higher priority than others. Such scenarios are theoretically modelled in two ways: $(i)$ jobs are associated with weights that reflect their priority. The jobs' performance measure is scaled by the weight, thus jobs with higher weight get better quality of service. $(ii)$ the scheduling instance includes a directed acyclic graph describing precedence constraints among jobs. A directed edge from job $j_1$ to job $j_2$ implies that the processing of $j_2$ can start only after the processing of $j_1$ is completed.

In this paper we study a scheduling setting, motivated by real-life behaviour, in which the priority among jobs is defined in a different way. Our model reflects real-life environments in which the schedule is not determined completely by the system. Traditionally, scheduling problems have been studied from a centralized point of view, that is, a centralized authority, 'the scheduler' determines the assignment. Many modern systems provide service to multiple strategic users, who may influence the possible outcomes. As a result, non-cooperative game theory has become an essential tool in the analysis of job-scheduling applications [21], [5]. Our model studies a natural setting, in which the users are arranged in a *dominance hierarchy*, and high ranking users are ready to accept only outcomes in which the service they receive is better than the service of subordinate users.

In behavioral sciences, the study of dominance hierarchy is based on the fact that different organisms have different aggressiveness levels. Aggression is defined as a behavior which is intended to increase the social dominance of the organism relative to the dominance position of other organisms [6]. Different levels of aggressiveness lead to a dominance hierarchy - a type of social hierarchy that arises when members of a social group interact [4], [9]. Highly rank members of the society have better access to valuable resources such as mates and food. Our model is inspired by such environments. Specifically, in our setting, the jobs are partitioned into $c$ hierarchical levels. High-ranking jobs can bypass subordinate jobs if this improves their performance. Moreover, all the jobs, from all hierarchy levels cooperate and are ready to modify their assignment if this modification does not harm their performance, and may help other high-ranking jobs get an advantage over subordinate ones.

In Section II we define the model and the set of feasible schedules formally. We also present algorithms for testing the feasibility of a schedule with respect to jobs' tardiness and with respect to the lateness indicator. In Section III we present optimal algorithms for the problem of minimizing the maximal tardiness of a job. In Section IV we consider the problem of minimizing the number of tardy jobs. We distinguish between the global objective in which the goal is to find a feasible schedule that minimizes the total number of tardy jobs, independent of their hierarchy level, and the multi-criteria objective, in which the primary goal is to minimize the number of tardy highly ranked jobs, the secondary goal is to minimize the number of tardy jobs from the 2nd rank, and so on. For a constant number of hierarchy levels we present an optimal algorithm for the multi-criteria objective, and an NP-hardness proof for the global objective.

## II. Preliminaries

Let $\mathcal{J}$ be a set of jobs. Every job $J_j \in \mathcal{J}$ has a processing time $p_j$, as well as a due-date $d_j$, denoting the time in which it should be completed. The set $\mathcal{J}$ consists of $c$ sets; $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2 \cup \ldots \cup \mathcal{J}_c$, where $\mathcal{J}_k$ is a set of jobs in the $\ell$-th hierarchy level. That is, the jobs of $\mathcal{J}_1$ have the highest rank, and the jobs of $\mathcal{J}_c$ are the most subordinate. Let $n = \sum_{\ell=1}^{c} |\mathcal{J}_\ell|$. A schedule $\pi$ on a single machine determines a non-preemptive assignment of the jobs on the machine. For a schedule $\pi$ and a job $J_j$, let $S_j(\pi), C_j(\pi)$ denote the start time and the completion time of $J_j$ in $\pi$. In our setting, all the jobs are available at time 0 (no release times), and no preemptions are allowed, therefore, for every job $j$, $C_j(\pi) - S_j(\pi) = p_j$. Also, w.l.o.g., we only consider schedules with no intended idle. Clearly, idle segments can be removed by shifting some jobs to start earlier. By the above, a schedule $\pi$ can be described by specifying an order of the jobs, and $C_j(\pi) = \sum_{j':S_{j'}(\pi) \leq S_j(\pi)} p_{j'}$.

For a given schedule $\pi$, let $L_j(\pi) = C_j(\pi) - d_j$ denote the lateness of job $J_j$ in $\pi$. The jobs need to be ready by their due-date; early completion of a job has no effect on the quality of service, thus, the study of scheduling environments in which jobs are associated with due-dates, considers mostly the two following measurements:

1) $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ is the *tardiness* of job $J_j$.
2) $U_j(\pi) \in \{0, 1\}$ is a binary *lateness indicator* indicating whether $J_j$ is *tardy*, that is, $U_j(\pi) = 1$ if and only if $C_j(\pi) > d_j$.

For a set $\mathcal{J}_\ell$, let $T_{\mathcal{J}_\ell}(\pi) = \max_{j \in \mathcal{J}_\ell} T_j(\pi)$ be the maximal tardiness of a job in $\mathcal{J}_\ell$. For the lateness indicator we measure the performance of a set of jobs by the number of tardy jobs in the set, in particular, $U_{\mathcal{J}_\ell}(\pi) = \sum_{j \in \mathcal{J}_\ell} U_j(\pi)$ is the number of tardy jobs in $\mathcal{J}_\ell$.

We will analyze two objective functions. The first is minimizing the maximal tardiness, and the second is minimizing the number of tardy jobs. Using the common three-fields notation for theoretic scheduling problems [10], we denote the corresponding problems in the presence of hierarchy levels by $1|hierarchy|T_{max}$ and $1|hierarchy|\sum U_j$.

High rank jobs can bypass and push subordinate jobs. They also cooperate with each other. Formally, a schedule $\pi$ is considered *feasible* if for every hierarchy level $1 \leq \ell \leq c$, and every job $J_i \in \mathcal{J}_\ell$ it holds that $J_i$ cannot improve its objective value by bypassing less dominant jobs, even if all the jobs having rank at least $\ell$ are ready to modify their assignment as long as they are not harmed. This general definition has a different practical meaning depending on the objective function. Specifically:

*Definition 2.1:* A schedule $\pi$ is *feasible with respect to tardiness* if for for every rank $1 \leq \ell \leq c$ and every tardy job $J_i \in \mathcal{J}_\ell$ it holds that there is no schedule $\pi'$ such that $C_i(\pi') < C_i(\pi)$ and for every job $J_j \in \cup_{1 \leq k \leq \ell} \mathcal{J}_k$ it holds that $T_j(\pi') \leq T_j(\pi)$. In other words, there is not schedule in which $J_i$ has a reduced tardiness, and no job from a higher or equal hierarchy level has a higher tardiness.

*Definition 2.2:* A schedule $\pi$ is *feasible with respect to the number of tardy jobs* if for every rank $1 \leq \ell \leq c$ and every tardy job $J_i \in \mathcal{J}_\ell$ it holds that there is no schedule $\pi'$ such that $C_i(\pi_i) \leq d_i$ and for every job $J_j \in \cup_{1 \leq k \leq \ell} \mathcal{J}_k$ it holds that $U_j(\pi') \leq U_j(\pi)$. Thus, $J_i$ completes on time and if a same or higher hank job $J_j$ is not tardy in $\pi$ it must complete in time also in $\pi'$.

Note that if the objective of a job is merely to minimize its completion time, then the hierarchy induces an order according to which jobs of different levels must be processed, and finding an optimal solution on a single machines is an easy task. Objective functions that depend on jobs' tardiness are more challenging since a job may have a high completion time and still perform perfectly as long as it is not tardy. Thus, the order of jobs in an optimal schedule does not necessarily agree with their ranks. This observation is crucial in understanding the model and the involved challenges.

The general problem we consider is finding a feasible schedule that optimizes the objective function, that is, minimize the maximal tardiness of a job, or minimizes the number of tardy jobs. A different goal that we consider is a multi-criteria one. Specifically, the primary goal is to optimize the schedule for $\mathcal{J}_1$. Out of all feasible schedules achieving the best for $\mathcal{J}_1$, the goal is to optimize the schedule for the jobs in $\mathcal{J}_2$, and so on. We use the notation $1|hierarchy|(\gamma_1, \ldots, \gamma_c)$ the denote the problem with the multi-criteria objective function $\gamma$. E.g., for $c = 2$, in the problem $1|hierarchy|(U_\mathcal{A}, U_\mathcal{B})$, the primary goal is to minimize the number of tardy dominant jobs, and among all the feasible schedules achieving this objective, minimize the number of tardy subordinate jobs.

We conclude the introduction with an example that demonstrates the optimality with respect to the general and the multi-criteria objective function. Consider the problem of minimizing the number of late jobs. That is, $1|hierarchy|\sum U_j$. Assume $c = 2$. Let $\mathcal{A} = \{a_1, a_2, a_3\}$ be the set of dominant jobs, where $p_1 = p_2 = L$ and $p_3 = L + 1$, for some constant $L > 2$. The set $\mathcal{B}$ of subordinate jobs includes $L-1$ unit-length jobs. Note that $n = |\mathcal{A}| + |\mathcal{B}| = L + 2$. Assume further that all the jobs in the instance have the same due-date $d_j = 2L$. An optimal schedule for $\sum U_j$ is the schedule $\pi_1$, presented in the top of Figure 1. There are 2 tardy jobs. The longer dominant job, $a_3$, and $L - 1$ subordinate jobs complete on time. The schedule $\pi_1$ is feasible, even though $a_1$ and $a_2$ are late. None of these jobs can benefit from bypassing subordinate jobs, as their total processing time is less than $L$. The schedule $\pi_2$ in Figure 1 is optimal for the problem $1|hierarchy|(U_\mathcal{A}, U_\mathcal{B})$. The two dominant jobs $a_1$ and $a_2$ are not late, and the other $L$ jobs are late. The above example illustrates some of the challenges in scheduling jobs with different hierarchy levels, and the difference from the global objective function and the multi-criteria one.

**Related work:** Job scheduling on a single machine has been widely studied. When there are no precedence constraints or

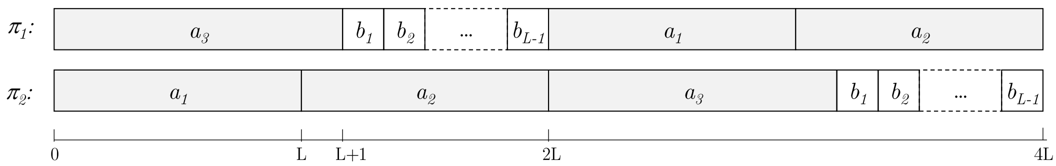


Fig. 1. $\pi_1$ is optimal for $1|hierarchy|\sum U_j$, while $\pi_2$ is optimal for $1|hierarchy|(U_1, U_1)$.

weights, the problem $1||T_{max}$ is solved optimally by Earliest Due-Date first (EDD) rule, that schedule the jobs in non-decreasing order of due-date [15]. The problem $1||\sum U_j$ is solved by Moore's algorithm [17].

When jobs are associated with weights, the problem $1||\sum w_j Uj$ becomes NP-hard even when the jobs all have common due-date [13]. Pseudo-polynomial time algorithms are given in [16] and [19]. If the number of different job weights is a constant, then $1||\sum w_j Uj$ is solvable in polynomial time [11]. The unweighted problem of minimizing the number of tardy jobs is strongly NP-hard when the jobs' processing order must obey some precedence constraints. This is true even if the precedence constraints are limited to chains and all jobs have unit length, that is, $1|chains; p_j = 1|\sum U_j$. See [1] for a survey on algorithms for single machine scheduling to minimize weighted number of tardy jobs.

For the maximum tardiness problem, the addition of weights does not change the complexity of the problem, that is, the weighted problem, $1||\max_j w_j T_j$, is solvable in polynomial time [12], [7]. Moreover, the problem remains tractable even in the addition of arbitrary precedence constraints [15]. On the other hand, when a machine can process several jobs simultaneously (batch-scheduling), the problem becomes NP-hard [3].

Aggressiveness is a lighter notion of priority. Dominant jobs can be processed after less dominant ones, if their performance is not harmed. An environment in which some jobs are aggressive is studied in [20], where a new notion of selfish precedence constraint is defined. The paper presents algorithms for scheduling jobs on parallel machines, where some of the jobs are aggressive. An aggressive job do not let non-aggressive jobs start processing before it. Additional relaxed models of precedence constraints are studied in [14], [2].

### A. Feasibility Tests

In this section we present algorithms for testing whether a given schedule is feasible with respect to some objective. For a schedule $\pi$, the algorithms returns *True* if $\pi$ is feasible, or *False due to* $J_i$, if $\pi$ is not feasible since some job $J_i$ can benefit from rearranging the jobs.

Algorithm 1 performs a feasibility test of a given schedule with respect to the jobs' tardiness. It proceeds by verifying, for every tardy job $J_i$, that there is no schedule in which $J_i$ has a reduced tardiness and the non-tardy jobs from hierarchy levels at least as high as $J_i$ are not harmed, as required by Definition 2.1.

**Algorithm 1** - Feasibility test of a schedule $\pi$ w.r.t $T_i$

1: Let $\mathcal{J}_{tardy}$ and $\mathcal{J}_{in.time}$ be, respectively, the set of tardy and non-tardy jobs in $\pi$.
2: **for** each job $J_i \in \mathcal{J}_{tardy}$ **do**
3: Assume $J_i \in \mathcal{J}_\ell$.
4: Let $S_1 = \cup_{1\le k\le \ell}\mathcal{J}_k \cap \mathcal{J}_{tardy}$.
5: Let $S_2 = \cup_{1\le k\le \ell}\mathcal{J}_k \cap \mathcal{J}_{in.time}$.
6: Let $\pi_i'$ be a schedule of $S_1 \cup S_2 \setminus \{J_i\}$ produced in the following way:
7: Assign the jobs in $S_1 \setminus \{J_i\}$ as in $\pi$.
8: Add the jobs in $S_2$ in non-increasing order of due-date. Every job $J_j$ is assigned, possibly with preemptions, in the latest available slots in $[0, d_j]$.
9: If $\pi_i'$ includes more than $p_i$ idle slots in $[0, C_i(\pi)]$ then return *False due to* $J_i$.
10: **end for**
11: return *True*.

*Lemma 2.1:* Algorithm 1 returns *True* if and only if $\pi$ is feasible with respect to $T_i$.

**Proof:** The algorithm proceeds by checking feasibility for every tardy job separately. Clearly, if a job $J_i$ is not late, then the schedule is feasible for it. If $J_i$ is late, then $S_1$ and $S_2$ are the sets of tardy and non-tardy jobs that are ranked in the hierarchy at least as high as $J_i$. We check whether there exists a schedule in which these jobs are not harmed, and the tardiness of $J_i$ is reduced.

Assume that the algorithm returns *False due to* $J_i$. Assume $J_i \in \mathcal{J}_\ell$. We show that there exists a schedule $\pi'$ such that $C_i(\pi') < C_i(\pi)$ and for every job $J_j \in \cup_{1\le k\le \ell}\mathcal{J}_k$ it holds that $T_j(\pi') \le T_j(\pi)$.

The schedule $\pi'$ is produced from the schedule $\pi_i'$ build in steps 6–8. First, preemptions are removed: if job $J_j$ is preempted in $\pi_i'$, then in $\pi'$ it is processed non preemptively in $[C_j(\pi_i') - p_j, C_j(\pi_i')]$. Jobs that were processed in this interval are shifted to start earlier. The tardiness of $J_j$ does not change, as its completion time remains $C_j(\pi_i')$. The tardiness of the shifted job could only decrease. After the preemption removal, we add $J_i$ in the earliest idle slots, possibly with preemptions. Since the condition in step 9 is met, $T_i(\pi') < T_i(\pi)$. Next, if $J_i$ is scheduled with preemptions, then preemptions are removed, without harming any of the completion times, as described above. Finally, the jobs from lower hierarchy levels $\cup_{\ell<k\le c}\mathcal{J}_k$ are added in arbitrary way.

Note that it is always possible to add the jobs of $S_2$ as required in Step 8 of the algorithm, since they are not late in $\pi$, thus, there is clearly sufficient space for them on the machine when the jobs of $\cup_{\ell<k\leq c}\mathcal{J}_k$ are removed.

By the condition in Step 9, the lateness of $J_i$ in $\pi'$ is lower than its lateness in $\pi$. Since all other jobs with at least the same rank are not harmed, we get a contradiction to the stability of $\pi$.

Assume that the algorithm returns *True*. It means that for every tardy job $j_i$ in $\pi$, there are at most $p_i$ idle slots in $[0, C_i(\pi)]$ in a schedule in which jobs of lower rank are removed, and each of the remaining jobs is scheduled as late as possible. This implies that it is not possible to rearrange the jobs in $\pi$ such that $J_i$ reduces its tardiness, without harming the performance of at least one job with rank higher or equal to rank of $J_i$. Thus, $\pi$ is feasible. ■

We turn to consider objectives that refer to the lateness indicator. Algorithm 2 performs a feasibility test of a given schedule with respect to the lateness indicator. The algorithm proceeds by verifying, for every tardy job, $J_i$, that there is no schedule in which $J_i$ completes on time and the jobs from hierarchy levels at least as high as $J_i$ are not harmed, as required by Definition 2.2.

**Algorithm 2** - Feasibility test of a schedule $\pi$ w.r.t $U_i$

1: Let $\mathcal{J}_{tardy}$ and $\mathcal{J}_{in.time}$ be, respectively, the set of tardy and non-tardy jobs in $\pi$.
2: **for** each job $J_i \in \mathcal{J}_{tardy}$ **do**
3: Assume $J_i \in \mathcal{J}_\ell$.
4: Let $S_2 = \cup_{1\leq k\leq \ell}\mathcal{J}_k \cap \mathcal{J}_{in.time}$.
5: Let $\pi'_i$ be a schedule in EDD order of the jobs in $S_2 \cup \{J_i\}$.
6: If no job is late in $\pi'_i$ then return *False due to* $J_i$.
7: **end for**
8: return *True*.

*Lemma 2.2:* Algorithm 2 returns *True* if and only if $\pi$ is feasible with respect to $U_i$.

**Proof:** The problem $1||T_{max}$ is known to be solvable optimally by EDD rule. In particular, if for some instance of $1||T_{max}$, there exists a schedule in which no job is late, that is, $T_{max} = 0$, then no job is late if the jobs are processed in EDD order. Algorithm 2 is based on the above fact.

Assume that the algorithm returns *False*. This implies that for some late job, there exists a schedule of $S_2 \cup \{J_i\}$ in which no job is late. Thus, $\pi$ can be replaced by the schedule $\pi'_i$ built in step 5, followed by a schedule in arbitrary order of the jobs that are late in $\pi$. This modified schedule is better for $J_i$ and does not harm the objective value of any job in $\cup_{1\leq k\leq \ell}\mathcal{J}_k$, as required. Thus, $\pi$ is not feasible.

Assume that the algorithm returns *True*. It means that for every late job in $\pi$, at least one job would be late in a schedule in which the jobs of $S_2 \cup \{J_i\}$ are processed in EDD order. Since EDD is optimal for $1||T_{max}$, there is no schedule in which none of these jobs is late. This implies that it is not possible to rearrange the jobs in $\pi$ such that $J_i$ is not late, without harming the performance of at least one job with rank at least as high as $J_i$. Thus, $\pi$ is feasible. ■

## III. Minimizing Maximal Tardiness

### *A. The multi-criteria objective function: $1|hierarchy|(T_{\mathcal{J}_1}, \ldots, T_{\mathcal{J}_c})$*

In this section we consider the multi-criteria objective function of minimizing the maximal tardiness. Formally, recall that for every $1 \leq \ell \leq c$, $T_{\mathcal{J}_\ell}(\pi) = \max_{j\in\mathcal{J}_\ell} T_j(\pi)$ denotes the maximal tardiness of a job in $\mathcal{J}_\ell$ in a schedule $\pi$. An optimal schedule achieves the minimal possible $T_{\mathcal{J}_1}$ and for all $\ell > 1$ it achieves the minimal possible $T_{\mathcal{J}_\ell}$ among all schedules that achieve the minimal $T_{\mathcal{J}_k}$ for every $1 \leq k < \ell$.

We present an optimal algorithm for the problem. Recall that algorithm EDD, that schedule the jobs in non-decreasing due-date order is optimal for the problem when there are no hierarchy levels. The algorithm is presented for $c = 2$, that is, $\mathcal{J} = \mathcal{A} \cup \mathcal{B}$, where $\mathcal{A}$ is a set of dominant jobs, and $\mathcal{B}$ a set of subordinate jobs. At the end of this section we explain how to generalize it for $c > 2$ hierarchy levels.

Algorithm 3 constructs an optimal schedule $\pi$ in two phases. First, all the jobs are assigned according to EDD order, and then the schedule is turned into a feasible one, by letting some dominant jobs pass some subordinate jobs. Recall that for a schedule $\pi$ and a job $J_i$, we denote by $S_i(\pi)$ and $C_i(\pi)$ the start time and the completion time of $J_i$ in $\pi$.

**Algorithm 3** - An optimal algorithm for $1|hierarchy|(T_{\mathcal{A}}, T_{\mathcal{B}})$

1: Schedule all jobs according to EDD order, that is, $d_1 \leq d_2 \leq \cdots \leq d_n$.
2: Let $\pi$ be the schedule produced by EDD.
3: **for** each job $J_i \in \mathcal{A}$ according to their order in $\pi$ **do**
4: **while** $J_i$ is late and at least one job from $\mathcal{B}$ precedes it **do**
5: Let $J_k$ be the job in $\mathcal{B}$ for which $S_k(\pi) < S_i(\pi)$, and $S_k(\pi)$ is maximal.
6: Shift the jobs scheduled in $[C_k(\pi), C_i(\pi)]$ earlier by $p_k$ units.
7: Schedule $J_k$ right after $J_i$.
8: **end while**
9: **end for**

*Theorem 3.1:* Algorithm 3 produces a feasible schedule, optimal for the bi-criteria problem $(T_{\mathcal{A}}, T_{\mathcal{B}})$.

**Proof:** Let $\pi$ be the schedule produced by the algorithm for an input $\mathcal{A} \cup \mathcal{B}$. Every dominant job $J_i \in \mathcal{A}$ is considered in the while loop. Note that in the shifts performed in step 6, the jobs that are shifted forward are all dominant, since $J_k$ is the last $\mathcal{B}$-job before $J_i$. Combining this with the initial EDD order, we get,

*Observation 3.2:* In $\pi$, the jobs in $\mathcal{A}$ are processed in EDD order, and the jobs in $\mathcal{B}$ are processed in EDD order.

The while loop terminates if $J_i$ is not late or if it is preceded only by $\mathcal{A}$-jobs with lower or equal due-date. Also, by Observation 3.2, in the final schedule, every $\mathcal{B}$-job is preceded by $\mathcal{B}$-jobs with lower or equal due-date, or $\mathcal{A}$-jobs that bypassed it in order to reduce their tardiness. Thus, $\pi$ is feasible.

Next note that no $\mathcal{A}$-job that is processed after a $\mathcal{B}$-job is late. Thus, as illustrated in Figure 2, the schedule $\pi$ begins with a sequence of $\mathcal{A}$-jobs that are processed in a row, and are possibly late, followed a mixture of $\mathcal{B}$-jobs and non-late $\mathcal{A}$-jobs. Let $J_z$ be the last late $\mathcal{A}$-job in $\pi$. Let $\mathcal{A}_1$ be the subset of $\mathcal{A}$-jobs that are processed sequentially in $[0, C_z(\pi)]$, and let $\mathcal{A}_2$ be the set of remaining $\mathcal{A}$-jobs, that are not late and are processed interleaved with $\mathcal{B}$-jobs after $C_z(\pi)$. We prove the optimality of $\pi$ by considering separately the prefix in which the jobs of $\mathcal{A}_1$ are processed and the suffix in which the jobs of $\mathcal{A}_2 \cup B$ are processed.

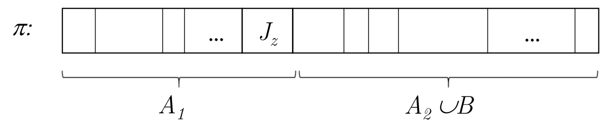


Fig. 2. The structure of the optimal schedule $\pi$

*Lemma 3.3:* Every optimal schedule $\pi^\star$ can be modified such that it agrees with $\pi$ on the assignment of $\mathcal{A}_1$, without harming its feasibility nor the objective function.

**Proof:** Since $J_z$ is the last late $\mathcal{A}$-job in $\pi$, which is a feasible schedule, any solution that schedules some $\mathcal{B}$-job, $J_b$, in the interval $[0, C_z(\pi)]$ is not feasible, as $J_z$ may reduce its tardiness by bypassing $J_b$. Thus, in any feasible schedule, only $\mathcal{A}$-jobs are processed in the interval $[0, C_z(\pi)]$.

Assume that $\pi^*$ does not agree with $\pi$ on the assignment of $\mathcal{A}_1$, and let $J_i \in \mathcal{A}_1$ be the first job in $\pi$ that has a higher starting time in $\pi^\star$. We use an exchange argument to show that $\pi^*$ can be converted to agree with $\pi$ on the assignment of $J_i$ without harming its feasibility nor increasing the maximal tardiness of jobs in $\mathcal{A}$ or $\mathcal{B}$. Let $H$ be the set of jobs that are scheduled in $\pi^\star$ during the interval $[S_i(\pi), S_i(\pi^\star)]$. Let $\pi'$ be the schedule obtained from $\pi^\star$ by moving $J_i$ before $H$ in $\pi^\star$ (see Figure 3). By Observation 3.2, each of these jobs has higher due-date than $d_i$.

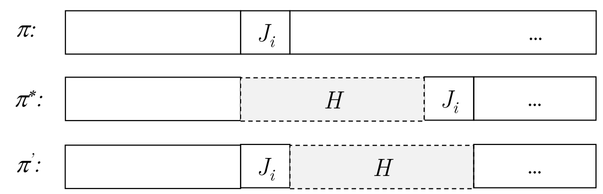


Fig. 3. Converting $\pi^\star$ to a profile $\pi'$ that agrees with $\pi$ on the assignment of job $J_i \in \mathcal{A}$.

The schedule $\pi'$ is feasible: By the feasibility of $\pi^*$, the set $H$ includes only $\mathcal{A}$-jobs, as otherwise, $J_i$ or another job from $\mathcal{A}_1$ is tardy and can reduce its tardiness by bypassing the $\mathcal{B}$-jobs in $H$.

Since in $\pi$, the jobs of $\mathcal{A}_1$ are processed in EDD order, for every $J_k \in H$, it holds that $d_i \leq d_k$ and $C_k(\pi') \leq C_i(\pi^\star)$. Therefore, the lateness of $J_k$ in $\pi'$ is not higher than the lateness of $J_i$ in $\pi^\star$, and the maximal tardiness among the jobs in $\mathcal{A}$ is not harmed. The jobs that are processed after $C_i(\pi^\star)$ are not affected by the exchange, and their tardiness does not change.

By repeating the above exchange argument as long as $\pi^\star$ does not agree with $\pi$ on the assignment of $\mathcal{A}_1$, we get the statement of the lemma. ∎

We turn to consider the jobs of $\mathcal{B} \cup \mathcal{A}_2$. These jobs are processed after time $C_z(\pi)$.

*Claim 3.4:* Every optimal schedule $\pi^\star$ can be modified such that no job in $\mathcal{A}_2$ is late, without delaying any job in $\mathcal{B}$.

**Proof:** Let $J_i \in \mathcal{A}_2$ be a late job in $\pi^\star$. Since $\pi^\star$ is feasible, $J_i$ is precedes only by $\mathcal{A}$-jobs. Also, we can assume that the machine is not idle between these jobs, as otherwise, idles can be removed by shifting the jobs to start earlier, without harming the feasibility or the quality of the solution. Modify $\pi^\star$ be rearranging in EDD order $J_i$ and the $\mathcal{A}$-jobs from $\mathcal{A}_2$ that precedes it. From the optimally of EDD, the maximal tardiness of the $\mathcal{A}$-jobs in the resulting schedule is equal to or lower than their maximal tardiness before the modification. After the reorder, the order of the jobs agrees with $\pi$, and since in $\pi$ no job from $\mathcal{A}_2$ is late, this is true for $\pi^\star$ as well. ∎

Based on Claim 3.4, we can assume w.l.o.g., that no jobs in $\mathcal{A}_2$ is late in $\pi^\star$.

*Lemma 3.5:* Every optimal schedule $\pi^\star$ can be modified such that it agrees with the assignment of $\mathcal{A}_2 \cup \mathcal{B}$ in $\pi$ without harming its feasibility nor the objective function.

**Proof:** We show that $\pi^\star$ can be converted to agree with $\pi$. Specifically, we use an exchange argument for handling the leftmost disagreement. The same argument can be applied as long as the schedules are not identical.

Let $J_i \in \mathcal{A}_2 \cup \mathcal{B}$ be the first job in $\pi$ that has a different starting time in $\pi^\star$. Let $H$ be the set of jobs that are scheduled in $\pi^\star$ during the interval $[S_i(\pi), S_i(\pi^\star)]$. We distinguish between two cases depending on the hierarchy level of $J_i$.

Assume first that $J_i \in \mathcal{A}_2$. As in the case $J_i \in \mathcal{A}_1$, let $\pi'$ be the schedule obtained from $\pi^\star$ by moving $i$ before $H$ in $\pi^\star$ (see Figure 3). We show that $\pi'$ is feasible and has the same objective value. Consider an $\mathcal{A}$-job $J_k \in H$. Since Algorithm 3 schedules $\mathcal{A}$-jobs by EDD order, for every $\mathcal{A}$-job $J_k \in H$, it holds that $d_k \geq d_i$. By Claim 3.4, $J_i$ is not late in $\pi^\star$, hence $C_i(\pi^\star) \leq d_i$. For every $\mathcal{A}$-job $J_k \in H$, we have that $C_k(\pi') \leq C_i(\pi^\star) \leq d_i \leq d_k$, that is, $J_k$ is not late in $\pi'$.

We conclude that all $\mathcal{A}$-jobs in $H$ will not be late after the modification, therefore the maximal tardiness of the $\mathcal{A}$-jobs is not affected.

Consider now a $\mathcal{B}$-job $J_k \in H$. Algorithm 3 schedules $J_i$ before $J_k$ in two cases:

1) $d_i \leq d_k$. By the feasibility of $\pi^*$, $J_i$ is not late in $\pi^\star$. Since $d_i \leq d_k$, $J_k$ is not late in $\pi^\star$ as well. In this case, $J_k$ will not be late after the modification, since $C_k(\pi') \leq C_i(\pi^\star) \leq d_i \leq d_k$.
2) $d_i > d_k$. We show that this case never happens. $J_i$ is scheduled in $\pi$ before $J_k$ even-though $d_i > d_k$ since $J_k$ was delayed when some $J_\ell \in \mathcal{A}$ is considered in the

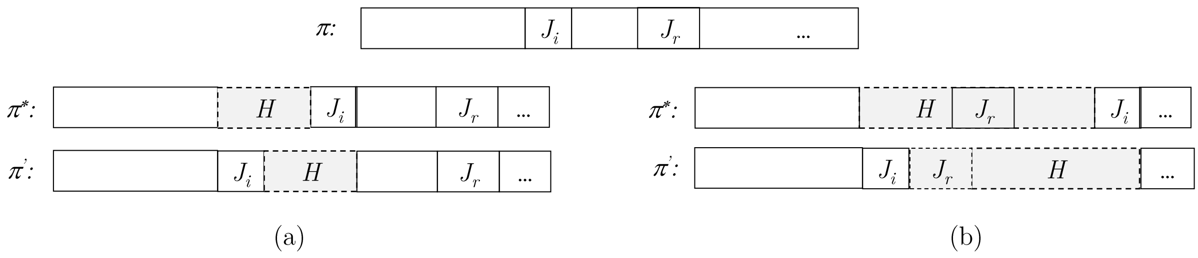


Fig. 4. $J_i \in \mathcal{B}$. (a) $J_r$ is scheduled after $J_i$ in $\pi^\star$, (b) $J_r$ is scheduled before $J_i$ in $\pi^\star$

while loop (possibly $\ell = i$). By the algorithm, $d_i \leq d_\ell$, and a consequent set of jobs from $\mathcal{A}$ are processed in $\pi$ in EDD order in $[S_i(\pi), C_\ell(\pi)]$. Moreover, since $J_k$ must be delayed in order to prevent $J_\ell$ from being late, at least one of these jobs will be late if $p_k$ precedes one of them, contradicting the feasibility of $\pi^\star$.

Therefore, the maximal tardiness of the $\mathcal{B}$-jobs in $H$ is not affected by the modification.

We turn to consider the case $J_i \in \mathcal{B}$.

Let $H$ be the set of jobs that are scheduled in $\pi^\star$ during the interval $[S_i(\pi), S_i(\pi^\star)]$, and let $J_r$ be the first $\mathcal{A}$-job scheduled after $J_i$ in $\pi$. Note that $J_r$ has the minimal due date among all $\mathcal{A}$-jobs that follow $J_i$ in $\pi$. We distinguish between two cases:

1) $J_r$ is scheduled after $J_i$ in $\pi^\star$. Let $\pi'$ be the schedule obtained from $\pi^\star$ by moving $J_i$ before $H$ in $\pi^\star$ (see Figure 4(a)). We show that $\pi'$ is feasible and has the same objective value. First, we show that the maximal tardiness of the jobs in $\mathcal{B} \cap H$ in $\pi'$ is not higher than the maximal tardiness of these jobs in $\pi^\star$.
For $J_i$, the exchange is clearly beneficial, as it is moved to start earlier. For every other $\mathcal{B}$-job $J_k \in H$, the EDD order applied in Algorithm 3 implies that $d_i \leq d_k$, thus,

$$\begin{aligned} L_k(\pi') = C_k(\pi') - d_k &\leq C_i(\pi^\star) - d_k \\ &\leq C_i(\pi^\star) - d_i = L_i(\pi^\star) \end{aligned}$$

We conclude that

$$max(L_k(\pi'), L_i(\pi')) \leq max(L_k(\pi^\star), L_i(\pi^\star)).$$

Since $T_j = max(L_j, 0)$ we get,

$$max(T_k(\pi'), T_i(\pi')) \leq max(T_k(\pi^\star), T_i(\pi^\star)).$$

Therefore, the maximal tardiness of a job in $\mathcal{B} \cap H$ is not harmed.
Next, we consider the jobs in $\mathcal{A}_2 \cap H$. Since $J_r$ has the minimal due date among all $\mathcal{A}$-jobs that follow $J_i$ in $\pi$, every $\mathcal{A}$-job $J_k \in H$ satisfies $d_k \geq d_r$. In addition, by Claim 3.4, $J_r$ is not late in $\pi^\star$. Thus,

$$C_k(\pi') \leq C_i(\pi^\star) \leq C_r(\pi^\star) \leq d_r \leq d_k.$$

Therefore, no $\mathcal{A}$-job in $\mathcal{A}_2 \cap H$ is late in $\pi'$, and the maximal tardiness is not affected.

2) $J_r$ is scheduled before $J_i$ in $\pi^\star$ (see Figure 4(b)). In this case, let $\pi'$ be the schedule obtained by moving $J_i$ to precede $H$ in $\pi^\star$, and reorder the jobs in $H$, such that $\mathcal{A}$-jobs in $H$ appear first, in EDD order, and are followed by the $\mathcal{B}$-jobs in $H$, also in EDD order. Note that $J_r \in H$ and since it has the minimal due date among the $\mathcal{A}$-jobs that follow $J_i$ in $\pi$, it is now processed right after $J_i$.
Clearly, $J_i$ is not late in $\pi'$. $J_r$ is not late in $\pi^\star$ and in $\pi$. Since it is processed after $J_i$ in $\pi$, it is not late in $\pi'$ either. The remaining $\mathcal{A}$-jobs in $H$ are processed in both $\pi$ and $\pi'$ in EDD order and are preceded by both $J_i$ and $J_r$. Since they are not late in $\pi$, they are not late in $\pi'$ either. The $\mathcal{B}$-jobs in $H$ are processed last, in EDD order. For each such job $J_k$, by Algorithm 3, $d_i \leq d_k$, therefore, $L_k(\pi') = C_k(\pi') - d_k \leq C_i(\pi^\star) - d_i = L_i(\pi^\star)$. We conclude that the maximal tardiness of the $\mathcal{B}$-jobs in $\pi'$ is not be higher than the tardiness of $J_i$ in $\pi^\star$. Therefore, the modification does not increase the maximal tardiness of a job in $\mathcal{B}$.

By repeating the above exchange argument, as long as $\pi^\star$ does not agree with $\pi$, we conclude that every optimal schedule can be modified such that it agrees with $\pi$, and its objective value is not harmed. Also, the initial EDD order implies the feasibility for the $\mathcal{B}$-jobs. That is, no $\mathcal{B}$-job can benefit from rearranging other jobs in a way that reduces its tardiness and does not harm any of the other jobs. ■

Combining Lemmas 3.3 and 3.5, we conclude that Algorithm 3 produces a feasible schedule that is optimal for the bi-criteria objective $(T_\mathcal{A}, T_\mathcal{B})$. ■

**Extension for $c > 2$ hierarchy levels:** Algorithm 4 extends Algorithm 3 for more than two hierarchy levels. The idea is to consider the levels one after the other. When the set $\mathcal{J}_\ell$ is considered, all the sets $J_\ell + 1, \ldots, \mathcal{J}_c$, can be viewed as a single subordinate level, and is therefore treated as the class $\mathcal{B}$ in the case of $c = 2$.

**Algorithm 4** - An optimal algorithm for $1|hierarchy|(T_{\mathcal{J}_1}, \ldots, T_{\mathcal{J}_c})$

```
Schedule all jobs according to EDD order, that is, d_1 ≤ d_2 ≤ ··· ≤ d_n.
Let π be the schedule produced by EDD.
for ℓ = 1 to c − 1 do
    Let B = ∪_{j=ℓ+1}^{c} J_j
    for each job J_i ∈ J_ℓ according to their order in π do
        while J_i is late and at least one job from B precedes it do
            Let J_k be the job in B for which S_k(π) < S_i(π), and S_k(π) is maximal.
            Shift the jobs scheduled in [C_k(π), C_i(π)] earlier by p_k units.
            Schedule J_k right after J_i.
        end while
    end for
end for
```

The proof of the algorithm follows the structure of the proof of Algorithm 3. We show by induction that for every $1 \leq \ell < c$, the schedule after $\ell$ iterations is optimal with respect to the multi-criteria objective of the $\ell$ high hierarchy levels. The initial EDD order implies the optimality and feasibility for the subordinate class, $\mathcal{J}_c$.

### B. The global objective function: $1|hierarchy|T_{max}$

We turn to consider the global objective function of minimizing the maximal tardiness of a job. Unlike the multi-criteria objective, here we do not give priority to the objective achieved by highly ranked jobs, and only care about the maximal tardiness of any job, independent of its hierarchy level.

We show that the problem is optimally solvable. In particular, Algorithm 4, which was shown to be optimal for $1|hierarchy|(T_{\mathcal{J}_1}, \ldots, T_{\mathcal{J}_c})$, produces a schedule that achieves the minimal tardiness of any job.

*Theorem 3.6:* Algorithm 4 is optimal also for $1|hierarchy|T_{max}$.

**Proof:** The proof of Algorithm 3 for $c = 2$, as well as its extension for $c > 2$ (Algorithm 4), are based on exchange arguments. Specifically, every optimal schedule can be modified to a one that agrees with the schedule produced by the algorithm. A close look at the exchange arguments reveals that none of them harms the maximal tardiness of any job in the instance. Thus, if $\pi^*$ is an optimal schedule with respect to $T_{max}$ it can be converted to agree with the schedule $\pi$ produced by the algorithm, without harming its feasibility, nor the maximal tardiness. ■

## IV. Minimizing Number of Tardy Jobs

In this section we consider the objective function of minimizing the number of tardy jobs. We assume that the number $c$ of different hierarchy levels is a constant. Our results show an interesting distinction between the multi-criteria objective, for which we present an optimal algorithm, and the global objective function, for which we present a hardness proof.

Without a dominance hierarchy, Moore's algorithm is an optimal greedy algorithm for $1||\sum U_j$. A naive approach for the problem with jobs' hierarchy can be based on scheduling the highly rank jobs according to Moore's algorithm, then create spaces between the jobs, such that each job is shifted to complete as close as possible to its due date, and add the jobs of the next level. This approach fails because Moore's algorithm does not take into account the due-dates of the jobs as long as they are not late. The following example demonstrates this issue and highlight the challenges in solving the problem. Let $c = 2$. The dominant set is $\mathcal{A} = \{a_1, a_2\}$, where $p_1 = 2, d_1 = 2$ and $p_2 = 3, d_2 = 4$. The subordinate set consists of a single job $\mathcal{B} = \{b\}$, where $p_b = 1, d_b = 1$. Executing Moore's Algorithm on $\mathcal{A}$ gives the schedule $[a_1, a_2]$. No spacing is possible, thus, $b$ must be late when added. The resulting schedule, $\pi_1$, is shown in Figure 5. $U_{\mathcal{A}}(\pi_1) = 1, U_{\mathcal{B}}(\pi_1) = 1$. Note that the same number of tardy jobs is achieved if $b$ is assigned before $a_2$. The optimal solution for this instance is the schedule $\pi_2$, shown in Figure 5. We have $U_{\mathcal{A}}(\pi_2) = 1, U_{\mathcal{B}}(\pi_2) = 0$. Note that the jobs of $\mathcal{A}$ are not processed in EDD order.

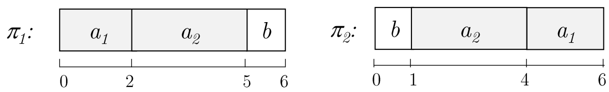


Fig. 5. $\pi_1$ : A schedule based on Moore's algorithm. Both $a_2$ and $b$ are tardy. $\pi_2$: An optimal schedule. $a_1$ is the only tardy job.

### A. The multi-criteria objective function: $1|hierarchy|(U_{\mathcal{J}_1}, \ldots, U_{\mathcal{J}_c})$

In the problem $1|hierarchy|(U_{\mathcal{J}_1}, \ldots, U_{\mathcal{J}_c})$, the goal is to find a feasible schedule that fulfills the following conditions:

1) The number of tardy jobs from the top hierarchy level, that is, $U_{\mathcal{J}_1}$, is minimal.
2) For every $2 \leq \ell \leq c$, the number of tardy jobs from the $\ell$-th hierarchy level, that is $U_{\mathcal{J}_\ell}$ is the minimal possible among all the schedules that achieve the minimal values of $U_{\mathcal{J}_1}, \ldots, U_{\mathcal{J}_{\ell-1}}$.

We present an optimal algorithm for a constant number of hierarchy levels. Specifically, we reduce the problem to the problem $1||\sum w_j U_j$, for which an optimal algorithm, based on dynamic programming, is presented in [11].

**Algorithm 5** - An optimal algorithm for $1|hierarchy|(U_{\mathcal{J}_1}, \ldots, U_{\mathcal{J}_c})$, with constant $c$.

```
Assign every job a weight:
For each J_i ∈ J_c, let w_i = 1.
for ℓ = c − 1 down to 1 do
    C_ℓ = 1 + Σ_{k=ℓ+1}^{c} |J_k| · C_k
    For each J_i ∈ J_ℓ, let w_i = 1 + C_ℓ
end for
Ignore the hierarchy levels and find an optimal solution for 1||Σ w_j U_j [11].
```

*Theorem 4.1:* Algorithm 5 is optimal for $1|hierarchy|(U_{\mathcal{J}_1},\ldots,U_{\mathcal{J}_c})$ with a constant number of levels.

**Proof:** The algorithm assign the jobs weight, such that $(i)$ jobs from the same rank have the same weight, and $(ii)$ the weight of each job in hierarchy level $\ell$ is equal to one plus the total weight of jobs in lower levels. The above weights imply that a single non-tardy job from hierarchy level $\ell$ contributes to the objective function more than all the jobs in lower levels. Thus, the multi-criteria objective function is achieved by minimizing $\sum w_jU_j$ in the resulting weighted instance.

We show that every optimal solution for $\sum w_jU_j$ corresponds to a feasible solution. Assume by contradiction that a schedule $\pi$ is optimal for $\sum w_jU_j$, but is not feasible due to job $J_i \in \mathcal{J}_\ell$. This means that $J_i$ can complete on time by delaying jobs from lower ranks. Since the weight of $J_i$ is higher than the total weight of lower rank jobs, we get a contradiction to the $\pi$'s optimality. ∎

The algorithm in [11] assumes that the instance is given as a list of jobs, every job, $J_i$, is represented by a triplet $(p_i,w_i,d_i)$. In the full version we show how to extend the algorithm to handle a compact representation of the input in which for every weight, $w_k$ we are either given a list of jobs having weight $w_k$, in which each job is represented by a pair $(p_i,d_i)$; or we are given the amount $n_k$ of jobs having weight $w_k$, and a single pair $(p_k,d_k)$ such that all $n_k$ jobs having weight $w_k$ have the same processing time $p_k$, and due-date, $d_k$.

The fact that an optimal poly-time algorithm exists for instance in the above compact representation gives a nice distinction between the multi-criteria objective and the global objective for which we who a hardness proof already for 4 hierarchy levels.

### B. The Global Objective function: $1|hierarchy|\sum U_j$

The goal in the problem $1|hierarchy|\sum U_j$ is to minimize the total number of tardy jobs, independent of their rank. Clearly, the dominance hierarchy plays a significant role also in the global objective problem since it induces the set of feasible schedules.

*Theorem 4.2:* The problem $1|hierarchy|\sum U_j$ is NP-complete for four or more hierarchy levels.

**Proof:** Given a schedule, it is possible to calculate the number of non-tardy jobs. Also, Algorithm 2 is a poly-time algorithm for verifying the feasibility of a given schedule, thus, the problem is in NP.

The hardness proof is by a reduction from the subset-sum problem. Given a set of integers $A=\{a_1,a_2,\ldots,a_{n_A}\}$ and a target value $T$, the goal is to decide whether $A$ has a subset $A'\subseteq A$ such that $\sum_{j\in A'}a_j=T$. The subset-sum problem is known to be NP-hard [8].

Given an instance of subset-sum, $(A,T)$, we build an instance of $1|hierarchy|\sum U_j$, consisting of four hierarchy levels, $\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D}$ as follows.

1) The set $\mathcal{A}$ includes the jobs at the top of the hierarchy. It consists of $n_A$ jobs induced by the subset-sum instance. Specifically, every element $a_j\in A$ contributes to $\mathcal{A}$ one jobs of length $a_j$ and due-date $d_j=T$.
2) The set $\mathcal{B}$ includes a single job, to be denoted $J_b$, for which $p_b=2$ and $d_b=T+1$.
3) The set $\mathcal{C}$ includes $T$ jobs of length 1, all having due-date $d_j=T$.
4) The set $\mathcal{D}$ of lowest level includes $T$ jobs of length $1/T$. All these jobs have due-date $d_j=T+1$.

We note that the reduction is polynomial assuming a compact representation of the input, that is, the value of $T$ may not be polynomial in $n$, but we do not list the jobs in $\mathcal{B}$ and $\mathcal{C}$, only specify their amount. Similarly, a schedule can be presented in a compact way - if there are $x$ jobs of length $p$ assigned one after the other, the schedule is presented by specifying $x$ and $p$, rather than listing all these jobs.

We turn to show the validity of the reduction. The idea is that the jobs of $\mathcal{D}$ whose total length is 1 can be assigned before their due-date only if $A$ is a YES-instance of the subset-sum problem. Intuitively, the job $J_b\in\mathcal{B}$ can benefit from bypassing the jobs of $\mathcal{C}\cup\mathcal{D}$ if and only if there is one more available slot for it in $[0,T]$, and such a slot exists only if the jobs of $\mathcal{A}$ do not use exactly all $T$ slots in $[0,T]$.

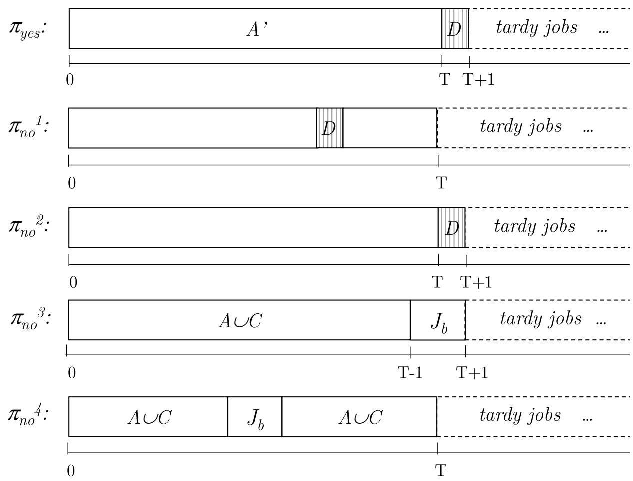


Fig. 6. $\pi_{yes}$ : An optimal feasible schedule of a YES-instance. $\pi^1_{no}$ and $\pi^2_{no}$ : Non-feasible schedules of a NO-instance. $\pi^3_{no}$ and $\pi^3_{no}$ : Feasible schedules of a NO-instance.

*Claim 4.3:* There exists a feasible schedule with more than $T$ non-tardy jobs if and only if $A$ has a subset that sums up to $T$.

**Proof:** Assume that $A$ has a subset $A'$ such that $\sum_{j\in A'}a_j=T$. Consider the schedule $\pi_{yes}$, depicted in Figure 6, in which the jobs corresponding to the elements of $A'$ are processed in arbitrary order during the interval $[0,T]$ and the $T$ jobs of $\mathcal{D}$ are processed in $[T,T+1]$. All other jobs are late and their schedule is arbitrary. We show that $\pi_{yes}$ is feasible. The tardy jobs of $\mathcal{A}\cup\mathcal{C}$ all have due-date $T$. Since the machine processes only jobs from $\mathcal{A}$ in $[0,T]$, and the non-tardy jobs of $A'$ complete their processing exactly at their due-date, it is not possible to add any tardy job to complete on time without harming a non-tardy job from $A'$. The job $J_b$, whose due-date is $T+1$ cannot benefit from bypassing the jobs of $\mathcal{D}$, since their total length is 1, and $p_b = 2$. Thus, $\pi_{yes}$ is a feasible schedule. The number of non-tardy jobs in $\pi_{yes}$ is $T + |A'|$.

Assume next that $A$ does not have a subset of total sum $T$. We show that the number of non-tardy jobs in an optimal feasible schedule is at most $T$. Specifically, we show that the jobs of $\mathcal{D}$ are not processed in any feasible schedule.

Let $\pi_{no}$ be a schedule of a NO-instance. Consider the interval $[0, T]$. Assume by contradiction that there are more than $T$ non-tardy jobs in $\pi_{no}$. Since the jobs of $\mathcal{J} \setminus \mathcal{D}$ all have length at least 1, at most $T$ jobs from $\mathcal{J} \setminus \mathcal{D}$ are processed in $[0, T]$. Also, since $p_b = 2$, if $J_b$ is not tardy, then at most $T$ jobs are processed in $[0, T+1]$. All the jobs that complete after time $T+1$ are clearly tardy. We conclude that if there are more than $T$ non-tardy jobs in $\pi_{no}$, then some jobs from $\mathcal{D}$ are non-tardy. Moreover, the jobs of $\mathcal{D}$ are the only jobs whose length is not integral and their total length is 1, therefore, in every optimal feasible solution with some non-tardy jobs from $\mathcal{D}$, all the jobs from $\mathcal{D}$ are non-tardy. Moreover, w.l.o.g., we assume that the jobs of $\mathcal{D}$ are processed sequentially in one time slot in $[0, T+1]$, as otherwise, they can be shifted to be processed sequentially; some other jobs may be shifted to start earlier, which is clearly beneficial for them and therefore does not harm the feasibility of the schedule.

We show that no schedule in which the jobs of $\mathcal{D}$ are allocated one time slot in $[0, T+1]$ is feasible. Assume first that the $\mathcal{D}$-jobs are assigned before time $T$ (see $\pi^1_{no}$ in Figure 6). If $J_b$ is tardy, then it can remove the jobs of $\mathcal{D}$ and be assigned in $[T-1, T+1]$, resulting in $\pi^3_{no}$. If $J_b$ is not tardy, then the jobs of $\mathcal{D}$ will be removed by a tardy job from $\mathcal{C}$, that can assign itself in their slot, resulting in $\pi^4_{no}$. This contradicts the feasibility of $\pi^1_{no}$.

Assume next that the $\mathcal{D}$-jobs are assigned in $[T, T+1]$ ($\pi^2_{no}$ in Figure 6). If $J_b$ is tardy, then since a subset of $A$ of total sum $T$ does not exist, at least one job from $\mathcal{C}$ is processed in $[0, T]$. Job $J_b$ can remove this job and the jobs of $\mathcal{D}$ and be assigned in $[T-1, T+1]$, resulting in $\pi^3_{no}$. If $J_b$ is not-tardy, then by removing the $\mathcal{D}$-jobs and be processed in $[T-1, T+1]$, $J_b$ can help some tardy job from $\mathcal{C}$ be processed before time $T$. Again, we get a contradiction to the feasibility of $\pi_{no}$.

The above analysis implies that the only possible feasible profiles, have the structure depicted in profiles $\pi^3_{no}$ or $\pi^4_{no}$ in Figure 6. The number of non-tardy jobs in these schedules is at most $T$. ∎

The above claim, together with the fact that subset-sum is NP-hard, implies that $1|hierarchy|\sum U_j$ is NP-hard, already for 4 hierarchy levels. ∎

## V. Conclusions

In this paper we analyzed a natural situation in real life scenarios, where some users are more dominant than others, and as a result they should receive a better quality of service. We considered the effect of having such dominance hierarchy on two classical scheduling problems.

We first provided efficient algorithms for testing the feasibility of a schedule, and then considered the problems of $(i)$ minimizing the maximal tardiness of a job, and $(ii)$ minimizing the number of tardy jobs. For the first problem we provided an efficient algorithm for both the bi-criteria objective and the global objective. For the second problem we provided an optimal solution for the bi-criteria objective, and presented a hardness proof for the global objective, when the number of different hierarchy levels in the input set is at least four.

Our work demonstrates the challenges arising in the analysis of systems with users' dominance hierarchy. We believe that this setting represents a natural phenomenon, which be studied further, in additional resource allocation environments.

## References


[1] M. Adamu and A. Adewumi. A survey of single machine scheduling to minimize weighted number of tardy jobs. *Journal of Industrial and Management Optimization*, 10:219–241, 2014.

[2] A. Agnetis, F. Rossi, and S. Smriglio. Some results on shop scheduling with s-precedence constraints among job tasks. *Algorithms*, 12(12), 2019.

[3] P. Brucker, A. Gladky, H. Hoogeveen, M. Y. Kovalyov, Chris N. Potts, T. Tautenhahn, and S. L. van de Velde. Scheduling a batching machine. *Journal of Scheduling*, 1(1):31–54, 1998.

[4] I.D. Chase, C. Tovey, and P. Murch, Two's Company, Three's a Crowd: Differences in Dominance Relationships in Isolated versus Socially Embedded Pairs of Fish. *Behaviour*. 140(10):1193–21, 2003.

[5] G. Christodoulou, E. Koutsoupias and A. Nanavati, Coordination mechanisms, *Theor. Comput. Sci.*, 410(36), 2009.

[6] C. Ferguson and K. M. Beaver. Natural born killers: The genetic origins of extreme violence. *Aggression and Violent Behavior*. 14(5):286-–294, 2009.

[7] M. C. Fields and G. N. Frederickson. A faster algorithm for the maximum weighted tardiness problem. *Information Processing Letters*, 36(1):39–44, 1990.

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

[9] L. R. Gesquiere, N. H .Learn, M. C.Simao, P. O. Onyango, S. C. Alberts, and J. Altmann. Life at the top: rank and stress in wild male baboons. *Science (New York, N.Y.)*, 333(6040), 357–360, 2011.

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

[11] D. Hermelin, S. Karhi, M. Pinedo, and D. Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research, Springer*, 298(1):271–287, 2012.

[12] D. Hochbaum and R. Shamir. An $O(nlog^2 n)$ algorithm for the maximum weigthed tardiness problem. *Information Processing Letters*, 31(4):215–219, 1989.

[13] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103, 1972.

[14] E.S. Kim and M. E. Posner. Parallel machine scheduling with s-precedence constraints. *IIE Transactions*, 42(7):525–537, 2010.

[15] E.L. Lawler. Optimal sequencing of a single machine subject to precedence constraints. *Management Sci.*, 19:544–546, 1973.

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

[17] J.M. Moore. An $n$ job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.*, 15:102–109, 1968.

[18] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2008.

[19] S. K. Sahni. Algorithms for scheduling independent tasks. *Journal of Assoc. Comput. Mach,*, 23:116–127, 1976.

[20] T. Tamir. Scheduling with bully selfish jobs. *Theory Comput. Syst.*, 50(1):124–146, 2012.

[21] B. Vöcking. *Algorithmic Game Theory*, chapter 20: Selfish Load Balancing. Cambridge University Press, 2007.