Collective Schedules: Scheduling Meets Computational Social Choice

Fanny Pascual
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6
Paris, France
fanny.pascual@lip6.fr

Krzysztof Rzadca
Institute of Informatics
University of Warsaw
Warsaw, Poland
krz@mimuw.edu.pl

Piotr Skowron
Institute of Informatics
University of Warsaw
Warsaw, Poland
p.skowron@mimuw.edu.pl

ABSTRACT

When scheduling public works or events in a shared facility one needs to accommodate preferences of a population. We formalize this problem by introducing the notion of a *collective schedule*. We show how to extend fundamental tools from social choice theory—positional scoring rules, the Kemeny rule and the Condorcet principle—to collective scheduling. We study the computational complexity of finding collective schedules. We also experimentally demonstrate that optimal collective schedules can be found for instances with realistic sizes.

KEYWORDS

scheduling; computational social choice; participatory scheduling

ACM Reference Format:

Fanny Pascual, Krzysztof Rzadca, and Piotr Skowron. 2018. Collective Schedules: Scheduling Meets Computational Social Choice. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Major public infrastructure projects, such as extending the city subway system, are often phased. As workforce, machines and yearly budgets are limited, phases have to be developed one by one. Some phases are inherently longer-lasting than others. Moreover, individual citizens have different preferred orders of phases. Should the construction start with a long phase with a strong support, or rather a less popular phase, that, however, will be finished faster? If the long phase starts first, the citizens supporting the short phase would have to wait significantly longer. Consider another example: planning events in a single lecture theater for a large, varied audience. The theater needs to be shared among different groups. Some events last just a few hours, while others multiple days. What is the optimal schedule? We formalize these and similar questions by introducing the notion of a *collective schedule*, a plan that takes into account both jobs' durations and their societal support. The central idea stems from the observation that the problem of finding a socially optimal collective schedule is closely related to the problem of aggregating agents' preferences, one of the central problems studied in social choice theory [2]. However, differences in jobs' lengths have to be explicitly considered. Let us illustrate these similarities through the following example.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

Consider a collection of jobs all having the same duration. The jobs have to be processed sequentially (one by one). Different agents might have different preferred schedules of processing these jobs. Since each agent would like all the jobs to be executed as soon as possible, the preferred schedule of each agent does not contain "gaps" (idle times), and so, such a preferred schedule can be viewed as an order over the set of jobs, and can be interpreted as a preference relation. Similarly, the resulting collective schedule can be viewed as an aggregated preference relation. From this perspective, it is natural to apply tools from social choice theory to find a socially desired collective schedule.

Yet, the tools of social choice cannot be always applied directly. The scheduling model is typically much richer, and contains additional elements. In particular, when jobs' durations vastly differ, these differences must be taken into account when constructing a collective schedule. For instance, imagine that we are dealing with two jobs—one very short, J_s, and one very long, J_l. Further, imagine that 55% of the population prefers the long job to be executed first and that the remaining 45% has exactly opposite preferences. If we disregard the jobs' durations, then perhaps every decision maker would schedule J_l before J_s. However, starting with J_s affects 55% of population just slightly (as J_l is just slightly delayed compared to their preferred schedules). In contrast, starting with J_l affects 45% of population significantly (as J_s is severely delayed).

1.1 Overview of Our Contributions

We explore the following question: How can we meaningfully apply the classic tools from social choice theory to find a collective schedule? The key idea behind this work is to use fundamental concepts from both fields to highlight the new perspectives.

Scheduling offers an impressive collection of models, tools and algorithms which can be applied to a broad class of problems. It is impossible to cover all of them in a single work. We use perhaps the most fundamental (although still non-trivial) scheduling model: a single processor executing a set of independent jobs. This model is already rich enough to describe significant real-world problems (such as the public works or the lecture theater introduced earlier). At the same time, such a model, fundamental, well-studied and stripped from orthogonal issues, enables us to highlight the new elements brought by social choice.

Similarly, we focus on three well-known and extensively studied tools from social choice theory: positional scoring rules, the Kemeny rule and the Condorcet principle. Under *a positional scoring rule* the score that an object receives from an agent is derived only on the basis of the position of this object in the agent's preference ranking; the objects are then ranked in the descending order of their total scores received from all the agents. The *Kemeny rule* uses the concept of distances between rankings. It selects a ranking which minimizes the sum of the swap distances to the preference rankings of all the agents. The *Condorcet principle* states that if there exists an object that is preferred to any other object by the majority of agents, then this object should be put on the top of the aggregated ranking. The Condorcet principle can be generalized to the remaining ranking positions. Assume that the graph of the preferences of the majority of agents is acyclic, i.e., there exists no such a sequence of objects $o_1, \ldots, o_\ell$ that o_1 is preferred by the majority of agents to o_2, o_2 to o_3, ..., $o_{\ell-1}$ to o_ℓ and o_ℓ to o_1. Whenever an object o is preferred by the majority of agents to another object q, o should be put before q in the aggregated ranking.

Naturally, these three notions can be directly applied to find a collective schedule. Yet, as we argued in our example with a long and a short job, this can lead to intuitively suboptimal schedules, because they do not consider significantly different processing times. We propose extensions of these tools to take into account lengths of the jobs. We also analyze their computational complexity.

Some of the proofs have been omitted due to space constraints. They can be found in the full version of this paper [23].

1.2 Related Work

Scheduling: The two most related scheduling models apply concepts from game theory and multiagent optimization. The selfish job model [17, 27] assumes that each job has a single owner trying to minimize its completion time and that the jobs compete for processors. The multi-organizational model [10] assumes that a single organization owns and cares about multiple jobs. Our work complements these with a third perspective: not only each job has multiple "owners", but also they care about all jobs (albeit to a different degree).

In multiagent scheduling [1], agents have different optimization goals (e.g., different functions or weights). The system's objective is to find all Pareto-optimal schedules, or a single Pareto-optimal schedule (optimizing one agent's goal with constraints on admissible values for other goals). In contrast, our aim is to propose rules allowing to construct a single, compromise schedule. This compromise stems from social choice methods and tools. Moreover, our setting is motivated by problems in which the number of agents is large. To the best of our knowledge, the existing literature on multiagent scheduling focuses on cases with a few (e.g. two) agents.

Computational social choice: For an overview of tools and methods for aggregating agents' preferences see the book of Arrow et al. [2]. Fischer et al. [14] overview the computational complexity of finding Kemeny rankings. Caragiannis et al. [6] discuss computational complexity of finding winners according to a number of Condorcet-consistent methods.

Typically in social choice, an aggregated ranking is created to establish the collective preference relation, and to eventually select a single best alternative (sometimes with a few runner-ups). Thus, the agents usually do not care what is the order of the candidates in the further part of the collective ranking. In our model the agents are interested in the whole output rankings. We can thus implement fairness—the agents who are dissatisfied with an order in the beginning of a collective schedule might be compensated in the further part of the schedule. Thus, our approach is closer to the recent works of Skowron et al. [26] and Celis et al. [7] analyzing fairness of collective rankings.

In participatory budgeting [3, 5, 12, 15, 24] agents express preferences over projects which have different costs. The goal is to choose a socially-optimal set of items with a total cost not exceeding the budget. Thus, in a way, participatory budgeting extends the knapsack problem similarly to how we extend scheduling.

2 THE COLLECTIVE SCHEDULING MODEL

We use standard scheduling notations and definitions from the book of Brucker [4], unless otherwise stated. For each integer t, by $[t]$ we denote the set $\{1, \ldots, t\}$. Let $N = [n]$ be the set of n agents (voters) and let $\mathcal{J} = \{J_1, \ldots, J_m\}$ be the set of m jobs (note that in scheduling m is typically used to denote the number of machines; we deliberately abuse this notation as our results are for a single machine). For a job J_i by $p_i \in \mathbb{N}$ we denote its processing time (also called duration or size), i.e., the number of time units J_i requires to be completed. We consider an off-line problem, i.e., jobs $\mathcal{J}$ are known in advance. Jobs are ready to be processed (there are no release dates). For each job J_i its processing time p_i is known in advance (*clairvoyance*, a standard assumption in the scheduling theory). Once started, a job cannot be interrupted until it completes (we do not allow for *preemption* of the jobs).

There is a single machine that executes all the jobs. A schedule $\sigma\colon \mathcal{J} \to \mathbb{N}$ is a function that assigns to each job J_i its start time $\sigma(J_i)$, such that no two jobs J_k, J_ℓ execute simultaneously. Thus, either $\sigma(J_k) \geq \sigma(J_\ell)+p_\ell$ or $\sigma(J_\ell) \geq \sigma(J_k)+p_k$. By $C_i(\sigma)$ we denote the completion time of job J_i: $C_i(\sigma) = \sigma(J_i) + p_i$. We assume that a schedule has no gaps: for each job i, except the job that completes as the last one, there exists job j such that $C_i(\sigma) = \sigma(J_j)$. Let $\mathscr{S}$ denote the set of all possible schedules for the set of jobs $\mathcal{J}$.

Each agent wants all jobs to be completed as soon as possible, yet agents differ in their views on the relative importance of the jobs. We assume that each agent a has a certain preferred schedule $\sigma_a \in \mathcal{J}$, and when building σ_a, an agent is aware of the processing times of the jobs. In particular, σ_a does not have to directly correspond to the relative importance of jobs. For instance, if in σ_a a short job J_s precedes a long job J_ℓ, then this does not necessarily mean that a considers J_s more important than J_ℓ. a might consider J_ℓ more important, but she might prefer a marginally less important job J_s to be completed sooner as it would delay J_ℓ only a bit.

A schedule can be encoded as a (transitive, asymmetric) binary relation: $J_i \ \sigma_a \ J_k \Leftrightarrow \sigma_a(J_i) < \sigma_a(J_k)$. E.g., $J_1 \ \sigma_a \ J_2 \ \sigma_a \ \ldots \ \sigma_a \ J_m$ means that agent a wants J_1 to be processed first, J_2 second, and so on. We will denote such a schedule as $(J_1, J_2, \ldots, J_m)$.

We call a vector of preferred schedules, one for each agent, a *preference profile*. By $\mathscr{P}$ we denote the set of all preference profiles of the agents. A *scheduling rule* $\mathcal{R}\colon \mathscr{P} \to \mathscr{S}$ is a function which takes a preference profile as an input and returns a collective schedule.

In the remaining part of this section we propose different methods in which the preference profile is used to evaluate a proposed collective schedule σ (and thus, to construct a scheduling rule $\mathcal{R}$). All the proposed methods extrapolate information from σ_a (a preferred schedule) to evaluate σ. Such an extrapolation is common in social choice: in participatory budgeting it is typical to ask each agent to provide a single set of items [3, 5, 15, 24] (instead of preferences over sets of items); similarly in multiwinner elections, each agent provides separable preferences of candidates [13, 25]. Alternatively, we could ask an agent to express her preferences over all possible schedules. This approach is also common in other areas of social choice (e.g., in voting in combinatorial domains model [18]), yet it requires eliciting exponential information from the agents. There exist also middle ground approaches, using specifically designed languages, such as CP-nets, for expressing preferences.

2.1 Scheduling by Positional Scoring Rules

In the classic social choice, positional scoring rules are perhaps the most straightforward, and the most commonly used in practice, tools to aggregate agents' preferences. Informally, under a positional scoring rule each agent a assigns a score to each candidate c (a job, in our case), which depends only on the position of c in a's preference ranking. For each candidate the scores that she receives from all the agents are summed up, and the candidates are ranked in the descending order of their total scores.

There is a natural way to adapt this concept. For an increasing function $h\colon \mathbb{N} \to \mathbb{R}$ and a job J we define the h-score of J as the total duration of jobs scheduled after J in all preferred schedules:

$$h\text{-score}(J) = \sum_{a \in N} f\left(\sum_{J_i\colon J \;\sigma_a\; J_i} p_i\right).$$

The h-psf-rule (psf for positional scoring function) schedules the jobs by their descending h-scores. If jobs are unit-size ($p_i = 1$), then h-score(J) is simply the score that J would get from the classic positional scoring rule induced by h. For an identity function $h_{\mathrm{id}}(x) = x$, the h_{id}-psf-rule corresponds to the Borda voting method adapted to collective scheduling.

The so-defined scheduling methods differ from traditional positional scoring rules, by taking into account the processing times of the jobs:

(1) A score that a job J receives from an agent a depends on the total processing time rather than on the number of jobs that J precedes in schedule σ_a.
(2) When scoring a job J we sum the duration of jobs scheduled *after* J, rather than before it. This implicitly favors jobs with lower processing times. Indeed, consider two preferred schedules, σ and τ identical until time t, at which a long job J_ℓ is scheduled in σ, and a short job J_s is scheduled in τ. Since J_s is shorter, the total size of the jobs succeeding J_s in τ is larger than the total size of the jobs succeeding J_ℓ in σ. Consequently, J_s gets a higher score from τ than J_ℓ gets from σ.

However, this implicit preference for short jobs seems insufficient, as illustrated by the following example.

Example 2.1. Consider three jobs, $J_{\ell,1}, J_{\ell,2}, J_s$, with the processing times ℓ, ℓ, and 1, respectively. Assume that $\ell \gg 1$, and consider the following preferred schedules of agents:

$$\begin{array}{lccccc}
3n/8 + \epsilon \text{ of agents}: & J_{\ell,1} & \sigma & J_{\ell,2} & \sigma & J_s \\
3n/8 + \epsilon \text{ of agents}: & J_{\ell,2} & \sigma & J_{\ell,1} & \sigma & J_s \\
n/8 - \epsilon \text{ of agents}: & J_s & \sigma & J_{\ell,1} & \sigma & J_{\ell,2} \\
n/8 - \epsilon \text{ of agents}: & J_s & \sigma & J_{\ell,2} & \sigma & J_{\ell,1}
\end{array}$$

By h_{id}-psf-rule, $J_{\ell,1}$ and $J_{\ell,2}$ are scheduled before J_s. However, starting with J_s would delay $J_{\ell,1}$ and $J_{\ell,2}$ by only one time unit, while starting with $J_{\ell,1}$ and $J_{\ell,2}$ delays J_s by 2ℓ, an arbitrarily large value. Moreover, J_s is put first by roughly 1/4 of agents, a significant fraction.

Example 2.1 demonstrates that the pure social choice theory does not offer tools appropriate for collective scheduling (we will provide more arguments to support this statement throughout the text). To address such issues we propose an approach that builds upon social choice *and* the scheduling theory.

2.2 Scheduling Based on Cost Functions

A cost function quantifies how a given schedule τ differs from an agent's preferred schedule σ. In this section, we adapt to our model classic costs used in scheduling and in social choice. We then show how to aggregate these costs among agents in order to produce a single measure of a quality of a schedule. This approach allows us to construct a family of scheduling methods that, in some sense, extend the classic Kemeny rule.

Formally, a cost function f maps a pair of schedules, τ and σ, to a non-negative real value. We analyze the following cost functions. Below, τ denotes a collective schedule the quality of which we want to assess; while σ denotes the preferred schedule of a single agent.

2.2.1 Swap Costs. These functions take into account only the orders of jobs in the two schedules (ignoring the processing times), thus directly correspond to costs from social choice.

(1) The Kendall [16] tau (or swap) distance (**K**), measures the number of swaps of adjacent jobs to turn one schedule into another one. We use an equivalent definition that counts all pairs of jobs executed in a non-preferred order:
$$K(\tau,\sigma) = \left|\left\{(k,\ell)\colon J_k \;\tau\; J_\ell \text{ and } J_\ell \;\sigma\; J_k\right\}\right|.$$
(2) Spearman distance (**S**). Let $\mathrm{pos}(J,\pi)$ denote the position of job J in a schedule π, i.e., the number of jobs scheduled before J in π. The Spearman distance is defined as:
$$S(\tau,\sigma) = \textstyle\sum_{J \in \mathcal{J}} \left|\mathrm{pos}(J,\sigma) - \mathrm{pos}(J,\tau)\right|.$$

2.2.2 Delay Costs. These functions use the *completion times* $\{C_i(\sigma)\colon J_i \in \mathcal{J}\}$ of jobs in the preferred schedule σ (and thus, indirectly, jobs' lengths). The completion times form jobs' due dates, $d_i = C_i(\sigma)$. A delay cost then quantifies how far are the proposed completion times $\{c_i = C_i(\tau)\colon J_i \in \mathcal{J}\}$ from their due dates $\{d_i\}$ by one of the six classic criteria defined in Brucker [4]:

Tardiness (T) $T(c_i,d_i) = \max(0, c_i - d_i)$.
Unit penalties (U) how many jobs are late:
$$U(c_i,d_i) = \begin{cases} 1 & \text{if } c_i > d_i \\ 0 & \text{otherwise.} \end{cases}$$
Lateness (L) is similar to tardiness, but includes a bonus for being early: $L(c_i,d_i) = c_i - d_i$.
Earliness (E) $E(c_i,d_i) = \max(0, d_i - c_i)$.
Absolute deviation (D) $D(c_i,d_i) = |c_i - d_i|$.
Squared deviation (SD) $SD(c_i,d_i) = (c_i - d_i)^2$.

Each such a criterion $f \in \{T, U, L, E, D, SD\}$ naturally induces the corresponding delay cost of an agent, $f(\tau, \sigma)$:

$$f(\tau, \sigma) = \sum_{J_i \in \mathcal{J}} f\Big(C_i(\tau), C_i(\sigma)\Big).$$

In this work, we mostly focus on the tardiness T, which is both easy to interpret for our motivating examples and the most extensively studied in scheduling. However, there is interest to study the remaining functions as well. U and L are similar to T—the sooner a task is completed, the better. The remaining three measures (E, S, and SD) penalize the jobs which are executed before their "preferred times". However, each job when executed earlier makes other jobs executed later (e.g., after their due times). Thus, these penalties quantify the unnecessary (wasted) promotion of jobs executed too early (causing other jobs being executed too late).[1]

By restricting the instances to unit-size jobs, we can relate delay and swap costs. The Spearman distance S has the same value as the absolute deviation D (by definition), and twice that of T:

Proposition 2.2. *For unit-size jobs it holds that* $S(\sigma, \tau) = 2T(\sigma, \tau)$, *for all schedules* σ, τ.

Since different agents can have different preferred schedules, in order to score a proposed schedule τ we need to aggregate the costs across all agents. We will consider three classic aggregations:

The sum (Σ): $\sum_{a \in N} f(\tau, \sigma_a)$, a utilitarian aggregation.

The max: $\max_{a \in N} f(\tau, \sigma_a)$, an egalitarian aggregation.

The L_p norm (L_p): $\sqrt[p]{\sum_{a \in N} \big(f(\tau, \sigma_a)\big)^p}$, with a parameter $p \geq 1$. The L_p norms form a spectrum of aggregations between the sum (L_1) and the max (L_∞).

For a cost function $f \in \{K, S, T, U, L, E, D, SD\}$ and an aggregation $\alpha \in \{\Sigma, \max, L_p\}$, by α-f we denote a scheduling rule returning a schedule that minimizes the α-aggregation of the f-costs of the agents. In particular, for unit-size jobs the Σ-T rule is equivalent to Σ-S and to Σ-D, and Σ-K is simply the Kemeny rule.

Scheduling based on cost functions avoids the problems exposed by Example 2.1 (indeed for that instance, e.g., the Σ-T rule starts with the short job J_s). Additionally, these methods satisfy some naturally-appealing axiomatic properties, such as reinforcement, which is a particularly natural requirement in our case.

Definition 2.3 (Reinforcement). A scheduling rule $\mathcal{R}$ satisfies reinforcement iff for any two groups of agents N_1 and N_2, a schedule σ is selected by $\mathcal{R}$ both for N_1 and for N_2, then it should be also selected for the joint instance $N_1 \cup N_2$.

Proposition 2.4. *All* Σ-f *scheduling rules satisfy reinforcement.*

2.3 Beyond Positional Scoring Rules and Cost Functions: the Condorcet Principle

In the previous section we introduced several scheduling rules, all based on the notion of a distance between schedules. Thus, these scheduling rules are closely related to the Kemeny voting system. We now take a different approach. We start from desired properties of a collective schedule and design scheduling rules satisfying them.

Pareto efficiency is one of the most accepted axioms in social choice theory. Below we use a formulation analogous to the one used in voting theory (based on swaps in preferred schedules).

Definition 2.5 (Pareto efficiency). A scheduling rule $\mathcal{R}$ satisfies Pareto efficiency iff for each pair of jobs, J_k and J_ℓ, and for each preference profile $\sigma = (\sigma_1, \dots, \sigma_n) \in \mathscr{P}$ such that for each $a \in N$ we have $J_k \ \sigma_a \ J_\ell$, it holds that $J_k \ \mathcal{R}(\sigma) \ J_\ell$.

In other words, if all agents prefer J_k to be scheduled before J_ℓ, then in the collective schedule J_k should be before J_ℓ. Curiously, the total tardiness Σ-T rule does not satisfy Pareto efficiency:

Example 2.6. Consider an instance with 3 jobs J_1, J_2, J_3 with lengths 20, 5, and 1, respectively, and with two agents having preferred schedules $\sigma_a = (J_1, J_3, J_2)$ and $\sigma_b = (J_2, J_1, J_3)$. Both agents prefer J_1 to be scheduled before J_3. If our scheduling rule satisfied Pareto efficiency, then it would pick one of the following three schedules: (J_1, J_3, J_2), (J_1, J_2, J_3), or (J_2, J_1, J_3). The total tardinesses of these schedules are equal to: 21, 25, and 10, respectively. Yet, the total tardiness of the schedule (J_2, J_3, J_1) is equal to 7.

This example can be generalized to inapproximability:

Proposition 2.7. *For any* $\alpha > 1$, *there is no scheduling rule that satisfies Pareto efficiency and is* α*-approximate for* max-T *or* Σ-T.

Proof. Let us assume, towards a contradiction, that there exists a scheduling rule $\mathcal{R}$ that satisfies Pareto efficiency and is α-approximate for minimizing Σ-T (the proof for max-T is analogous). Let $x = \lceil 3\alpha \rceil$. Consider an instance with $x + 2$ jobs: one job J_1 of length x^2, one job J_2 of length x, and x jobs $J_3, \dots, J_{x+2}$ of length 1. Let us consider two agents with preferred schedules $\sigma_1 = (J_1, J_3, \dots, J_{x+2}, J_2)$ and $\sigma_2 = (J_2, J_1, J_3, \dots, J_{x+2})$. For each $i \in \{3, \dots, x+2\}$, both agents prefer job J_1 to be scheduled before job J_i. Let τ be the schedule returned by $\mathcal{R}$. Since $\mathcal{R}$ satisfies Pareto efficiency, for each $i \in \{3, \dots, x+2\}$, J_1 is scheduled before job J_i in τ. Thus τ is either σ_2 , or a schedule where J_1 is scheduled first, followed by i jobs of length 1 ($i \in \{0, \dots, x\}$), followed by J_2, followed by the $x - i$ remaining jobs of length 1. Let S_i be such a schedule. In S_i, the tardiness of job J_2 is $x^2 + i$ (this job is in first position in σ_2), and the tardiness of the jobs of length 1 is $(x - i)x$ (the $x - i$ last jobs in S_i are scheduled before J_2 in σ_1). Thus the total tardiness of S_i is $(x^2 + i) + (x - i)x \geq x^2 + x$. The total tardiness of schedule σ_2 is $x^2 + x$ (each of the x jobs $J_1, J_3, \dots, J_{x+2}$ in σ_2 finishes x time units later than in σ_1). Thus, the total tardiness of τ is at least $x^2 + x$. Let us now consider schedule τ', which does not satisfy Pareto efficiency, and which is as follows: job J_2 is scheduled first, followed by the jobs of length 1, followed by job J_1. The total tardiness of this schedule is $3x$ (the only job which is delayed compared to σ_1 and σ_2 is job J_1). This schedule is optimal for Σ-T. Thus the approximation ratio of $\mathcal{R}$ is at least $\frac{x^2+x}{3x} = \frac{x+1}{3} > \alpha$. Therefore, $\mathcal{R}$ is not α-approximate for Σ-T, a contradiction. □

Proposition 2.8. *If all jobs are unit-size, the scheduling rule* $\sum$-T *is Pareto efficient.*

Pareto efficiency is one of the most fundamental properties in social choice. However, sometimes (especially in our setting) there

[1] The considered metrics have their natural interpretations also in other more specific settings. E.g., the earliness E is useful if each task represents a (collective) work to be done by the agents (workers) and when agents do not want to work before their preferred start times. Similarly, D and SD can be used when an agent wants each task to be executed exactly at the preferred time.

exist reasons for violating it. For instance, even if all the agents agree that J_x should be scheduled before J_y, the preferences of the agents with respect to other jobs might differ. Breaking Pareto efficiency can help to achieve a compromise with respect to these other jobs.

Nevertheless, Proposition 2.7 motivated us to formulate alternative scheduling rules based on axiomatic properties. We choose the Condorcet principle, a classic social choice property that is stronger than Pareto efficiency. We adapt it to consider the durations of jobs.

Definition 2.9 (Processing Time Aware (PTA) Condorcet principle). A schedule $\tau \in \mathscr{S}$ is PTA Condorcet consistent with a preference profile $\sigma = (\sigma_1, \dots, \sigma_n) \in \mathscr{P}$ if for each two jobs, J_k and J_ℓ, it holds that $J_k \ \tau \ J_\ell$ whenever at least $\frac{p_k}{p_k+p_\ell} \cdot n$ agents put J_k before J_ℓ in their preferred schedule. A scheduling rule $\mathcal{R}$ satisfies the PTA Condorcet principle if for each preference profile it returns a PTA Condorcet consistent schedule, whenever such exists.

Let us explain our motivation for ratio $\frac{p_k}{p_k+p_\ell}$. Consider a schedule τ and two jobs, J_k and J_ℓ, scheduled consecutively in τ. By N_k we denote the set of agents who rank J_k before J_ℓ in their preferred schedules, and let us assume that $|N_k| > \frac{p_k}{p_k+p_\ell}n$; we set $N_\ell = N - N_k$. Observe that if we swapped J_k and J_ℓ in τ, then each agent from N_k would be disappointed. Since such a swap makes J_k scheduled p_ℓ time units later than in τ, the level of dissatisfaction of each agent from N_k could be quantified by p_ℓ. Thus, their total (utilitarian) dissatisfaction $\mathrm{dis}(N_k)$ could be quantified by $|N_k| \cdot p_\ell$. By an analogous argument, if we started with a schedule where J_ℓ is put right before J_k, and swapped these jobs, then the total dissatisfaction of agents from N_ℓ could be quantified by:

$$\begin{aligned}\mathrm{dis}(N_\ell) &= |N_\ell| p_k < \left(n - \frac{p_k}{p_k + p_\ell} n\right) p_k \\ &= n \cdot \frac{p_k p_\ell}{p_k + p_\ell} < |N_k| \cdot p_\ell = \mathrm{dis}(N_k).\end{aligned}$$

Thus, the total dissatisfaction of all agents from scheduling J_k before J_ℓ is smaller than that from scheduling J_ℓ before J_k. Definition 2.9 requires that in such case J_k should be indeed scheduled before J_ℓ.

Proposition 2.10 below highlights the difference between scheduling based on the tardiness and on the PTA Condorcet principle.

Proposition 2.10. *Even if all jobs are unit-size, the $\sum$-T rule does not satisfy the PTA Condorcet principle.*

Proof. Consider an instance with three jobs and three agents with the following preferred schedules:

$$\begin{aligned}&\sigma_1 = (J_1, J_2, J_3); \qquad \sigma_2 = (J_1, J_3, J_2); \quad \sigma_3 = (J_1, J_3, J_2);\\ &\sigma_4 = (J_2, J_3, J_1); \qquad \sigma_5 = (J_2, J_3, J_1).\end{aligned}$$

The only PTA Condorcet consistent schedule is (J_1, J_2, J_3) with the total tardiness of 6. At the same time, the schedule (J_1, J_3, J_2) has the total tardiness equal to 5. □

To construct a PTA Condorcet consistent schedule, we propose to extend Condorcet consistent [8, 19] election rules to jobs with varying lengths. For example, we obtain:

PTA Copeland's method. For each job J_k we define the score of J_k as the number of jobs J_ℓ such that at least $\frac{p_k}{p_k+p_\ell} \cdot n$ agents put J_k before J_ℓ in their preferred schedule. The jobs are scheduled in the descending order of their scores.

Iterative PTA Minimax. For each pair of jobs, J_k and J_ℓ, we define the defeat score of J_k against J_ℓ as $\max(0, \frac{p_k}{p_k+p_\ell} n - n_k)$, where n_k is the number of agents who put J_k before J_ℓ in their preferred schedule. We define the defeat score of J_k as the highest defeat score of J_k against any other job. The job with the lowest defeat score is scheduled first. Next, we remove this job from the preferences of the agents, and repeat (until there are no jobs left).

Other Condorcet consistent election rules, such as the Dogdson's rule or the Tideman's ranked pairs method, can be adapted similarly. It is apparent that they satisfy the PTA Condorcet principle.

PTA Condorcet consistency comes at a cost: e.g., the two scheduling rules violate reinforcement, even if the jobs are unit-size. Indeed, by the classic result of Young and Levenglick [28] one can infer that any rule that satisfies PTA-Condorcet principle, neutrality, and reinforcement must be a generalization of the Kemeny rule (i.e., must be equivalent to the Kemeny rule if the processing times of the jobs are equal). We conjecture that rules satisfying neutrality and reinforcement fail the PTA-Condorcet principle; it is an interesting open question whether such an impossibility theorem holds.

3 COMPUTATIONAL RESULTS

In this section we study the computational complexity of finding collective schedules according to the previously defined rules. We start from the simple observation about the two PTA Condorcet consistent rules that we defined in the previous section.

Proposition 3.1. *The PTA Copeland's method and the iterative PTA minimax rule are computable in polynomial time.*

We further observe that computational complexity of the rules which ignore the lengths of the jobs (rules based on swap costs) can be directly inferred from the known results from computational social choice. For instance, the Σ-K rule is simply the well-known and extensively studied Kemeny rule. Thus, in the further part of this section we focus on the rules based on delay costs.

3.1 Sum of Delay Costs

First, observe that the problem of finding a collective schedule is computationally easy for the total lateness (Σ-L). In fact, Σ-L ignores the preferred schedules of the agents and arranges the jobs from the shortest to the longest one.

Proposition 3.2. *The rule Σ-L schedules the jobs in the ascending order of their lengths.*

Proof. Consider the total cost of the agents:

$$\begin{aligned}\sum_{a \in N} L(\tau, \sigma_a) &= \sum_{a \in N} \sum_{J_i \in \mathcal{J}} (C_i(\tau) - C_i(\sigma_a)) \\ &= |N| \sum_{J_i \in \mathcal{J}} C_i(\tau) - \sum_{a \in N} \sum_{J_i \in \mathcal{J}} C_i(\sigma_a).\end{aligned}$$

Thus, the total cost of the agents is minimized when $\sum_{J_i \in \mathcal{J}} C_i(\tau)$ is minimal. This value is minimal when the jobs are scheduled from the shortest to the longest one. □

J_s	$P_{s,i,1}$	X_1	$P_{s,i,2}$	X_2	$\cdots$	$P_{s,i,\mu}$	X_μ	$\mathcal{J_S} - \{J_s\}$	$\mathcal{P} - \{P_{s,i,j}\}$
s	s_T	s_T+1		$2s_T+2$			$s_\Sigma+\mu$		

$\mathcal{J}_1$	X_1	$\mathcal{J}_2$	X_2	$\cdots$	$\mathcal{J}_\mu$	X_μ	$\mathcal{P}$
	s_T+1		$2s_T+2$			$s_\Sigma+\mu$	

Figure 1: The preferred schedule $\sigma_{(s,i)}$ of agent $a_{s,i}$ (top) and the optimal schedule (bottom).

On the other hand, minimizing the total tardiness Σ-T is NP-hard even with the unary representation of the durations of jobs. Du and Leung [9] show that minimizing total tardiness with *arbitrary* due dates on a single processor ($1||\sum T_i$) is weakly NP-hard. We cannot use this result directly as the due dates in our problem Σ-T are structured and depend, among others, on jobs' durations.

Theorem 3.3. *The problem of finding a collective schedule minimizing the total tardiness (Σ-T) is strongly* NP-*hard.*

Proof. We reduce from the strongly NP-hard 3-Partition problem. Let I be an instance of 3-Partition. In I we are given a multiset of integers $S = \{s_1, \dots, s_{3\mu}\}$. We denote $s_\Sigma = \sum_{s \in S} s$. We ask if S can be partitioned into μ triples that all have the same sum, $s_T = s_\Sigma/\mu$. Without loss of generality, we can assume that $\mu \geq 2$ and that for each $s \in S$, $\mu < s < \frac{s_T}{2}$ (otherwise, we can add a large constant s_Σ to each integer from S, which does not change the optimal solution of the instance, but which ensures that $\mu < s < \frac{s_T}{2}$ in the new instance). We also assume that the integers from S are represented in unary encoding.

From I we construct an instance I' of the problem of finding a collective schedule that minimizes the total tardiness in the following way. For each number $s \in S$ we introduce $1 + s\mu$ jobs: J_s and $\left\{P_{s,i,j} \colon i \in [s], j \in [\mu]\right\}$. We set the processing time of J_s to s. Further, for each $i \in [s]$ we set the processing time of $P_{s,i,1}$ to $(s_T - s)$, and of the remaining $j \geq 2$ jobs $P_{s,i,j}$ to s_T. We denote the set of all such jobs as $\mathcal{J_S} = \{J_s \colon s \in S\}$ and $\mathcal{P} = \left\{P_{s,i,j} \colon s, i \in [s], j \in [\mu]\right\}$. Additionally, we introduce μ jobs, $\mathcal{X} = \{X_1, \dots, X_\mu\}$, each having a unit processing time.

There are s_Σ agents. For each integer $s \in S$ we introduce s agents. The i-th agent corresponding to number s, denoted by $a_{s,i}$, has the following preferred schedule (in the notation below a set, e.g., $\{J_{s'}\}$ denotes that its elements are scheduled in a fixed arbitrary order):

$$\Big(J_s, P_{s,i,1}, X_1, P_{s,i,2}, X_2, \dots, P_{s,i,\mu}, X_\mu, \{J_{s'} \colon s' \neq s\},$$
$$\left\{P_{s',j,\ell} \colon (s' \neq s \text{ or } j \neq i) \text{ and } \ell \in [\mu]\right\}\Big).$$

We claim that the answer to the initial instance I is "yes" if and only if the schedule σ^* optimizing the total tardiness is the following one: $\left(\mathcal{J}_1, X_1, \mathcal{J}_2, X_2, \mathcal{J}_\mu, X_\mu, \mathcal{P}\right)$, where for each $i \in [\mu]$, $\mathcal{J}_i$ is a set consisting of jobs from $\mathcal{J_S}$ with lengths summing up to s_T (see Figure 1). If such a schedule exists, then the answer to I is "yes". Below we will prove the other implication.

Observe that any job from $\mathcal{J_S}$ should be scheduled before each job from $\mathcal{P}$. Indeed, for each pair $P_{s,i,j}$ and $J_{s'}$ only a single agent $a = a_{s,i}$ ranks $P_{s,i,j}$ before $J_{s'}$; at the same time there exists another agent $a' = a_{s',k}$ who ranks $J_{s'}$ first. As $J_{s'}$ is shorter than $P_{s,i,j}$, a' gains more from $J_{s'}$ scheduled before $P_{s,i,j}$, than a gains from $P_{s,i,j}$ scheduled before $J_{s'}$. Thus, if $P_{s,i,j}$ were scheduled before $J_{s'}$, we could swap these two jobs and improve the schedule (such a swap could only improve the completion times of other jobs since $J_{s'}$ is shorter than $P_{s,i,j}$).

By a similar argument, any job from $\mathcal{X}$ should be scheduled before each job from $\mathcal{P}$. Indeed, if it was not the case, then there would exist jobs $P = P_{s,i,j}$ and $X = X_{i'}$ such that P is scheduled right before X (this follows from the reasoning given in the previous paragraph—a job from $\mathcal{J_S}$ cannot be scheduled after a job from $\mathcal{P}$). Also, since all the jobs from $\mathcal{J_S}$ are scheduled before P, the completion time of X would be at least $s_\Sigma + \frac{s_T}{2} + 1 \geq s_\Sigma + \mu + 2$. For each agent, the completion time of X in their preferred schedule is at most equal to $\mu(s_T + 1) = s_\Sigma + \mu$. Thus, if we swap X and P the improvement of the tardiness due to scheduling X earlier would be at least equal to $2s_\Sigma$. Such a swap increases the completion time of P only by one, so the increase of the tardiness due to scheduling P later would be at most equal to s_Σ. Consequently, a swap would decrease the total tardiness, and so X could have not been scheduled after P in σ^*.

We further investigate the structure of an optimal schedule σ^*. We know that $\mathcal{J_S}\ \sigma^*\ \mathcal{P}$ and that $\mathcal{X}\ \sigma^*\ \mathcal{P}$, but we do not yet know the optimal order of jobs from $\mathcal{J_S} \cup \mathcal{X}$. Before proceeding further, we introduce one useful class of schedules, $\mathcal{T}$, that execute jobs in the order $(\mathcal{J_S}, \mathcal{X}, \mathcal{P})$. Observe that σ^* can be constructed starting from some schedule $\tau \in \mathcal{T}$ and performing a sequence of swaps, each swap involving a job $J \in \mathcal{J_S}$ and a job $X \in \mathcal{X}$. The tardiness of σ^* is equal to the tardiness of the initial τ adjusted by the changes due to the swaps. Below, we further analyze $\mathcal{T}$. First, any ordering of $\mathcal{J_S}$ in τ results in the same tardiness. Indeed, consider two jobs J_s and $J_{s'}$ such that $J_{s'}$ is scheduled right after J_s. If we swap J_s and $J_{s'}$, then the total tardiness of s agents increases by s' and the total tardiness of s' agents decreases by s. In effect, the total tardiness of all agents remains unchanged. Second, there exists an optimal schedule where the relative order of the jobs from $\mathcal{X}$ is $X_1\ \sigma^*\ X_2\ \sigma^* \dots \sigma^*\ X_\mu$. Thus, w.l.o.g., we constrain $\mathcal{T}$ to schedules in which $\mathcal{X}$ are put in exactly this order.

Since we have shown that all $\mathcal{T}$ always have the same tardiness, no matter how we arrange the jobs from $\mathcal{J_S}$, the tardiness of σ^* only depends on the change of the tardiness due to the swaps. Consider the job X_1, and consider what happens if we swap X_1 with a number of jobs from $\mathcal{J_S}$ so that eventually X_1 is scheduled at time s_T (its start time in all preferred schedules). In such a case, moving X_1 forward decreases the tardiness of each of s_Σ agents by $(s_\Sigma - s_T)$. Moving X_1 forward to s_T requires however delaying some jobs from $\mathcal{J_S}$. Assume that the jobs from $\mathcal{J_S}$ with the processing times $s_{i_1}, \dots s_{i_\ell}$ are delayed. Each such job needs to be scheduled one time unit later. Thus, the total tardiness of s_{i_1} agents increases by 1 (the agents who had this job as the first in their preferred schedule), of other s_{i_2} agents increases by 1, and so on. Since $s_{i_1} + \dots + s_{i_\ell} = s_\Sigma - s_T$, the total tardiness of all agents increases by $s_\Sigma - s_T$. Thus, in total, executing X_1 at s_T decreases the total tardiness by $s_\Sigma(s_\Sigma - s_T) - (s_\Sigma - s_T)$, a positive number. Also, observe that this value does not depend on how the jobs from $\mathcal{J_S}$ were initially arranged, provided that X_1 can be put so that it starts at s_T.

Starting X_1 earlier than s_T does not improve the tardiness of X_1, yet it increases tardiness of some other jobs, so it is suboptimal. By repeating the same reasoning for $X_2, \dots, X_\mu$ we infer that we obtain the optimal decrease of the tardiness when X_1 is scheduled at time s_T, X_2 at time $2s_T + 1$, etc., and if there are no gaps between the jobs. However, such schedule is possible to obtain if and only if the answer to the initial instance of 3-Partition is "yes". □

A similar strategy (yet, with a more complex construction) can be used to prove the NP-hardness of Σ-U.

Theorem 3.4. *The problem of finding a collective schedule minimizing the total number of late jobs (Σ-U) is strongly* NP*-hard.*

Nonetheless, if the jobs have the same size, the problem can be solved in polynomial time (highlighting the additional complexity brought by the main element of the collective scheduling). Our proof uses the idea of Dwork et al. [11] who proved an analogous result for the Spearman distance.

Proposition 3.5. *If all jobs have the same size, for each $f \in \{T, U, L, E, D, SD\}$ rule $\sum$-f can be computed in polynomial time.*

Proof. Let us fix $f \in \{T, U, L, E, D, SD\}$. We reduce the problem of finding a collective schedule to the assignment problem. Observe that when the jobs have all the same size, say p, then in the optimal schedule each job should be started at time ℓp for some $\ell \in \{0, \ldots, m-1\}$. Thus, we construct a bipartite graph where the vertices on one side correspond to m jobs and the vertices on the other side to m possible starting times of these jobs. The edge between a job J and a starting time ℓp has a cost which is equal to the total cost caused by job J being scheduled to start at time ℓp. The cost can be computed independently of how the other jobs are scheduled, and is equal to $\sum_{a \in N} f\big(\ell p + 1, C_i(\sigma_a)\big)$. Thus, a schedule that minimizes the total cost corresponds to an optimal assignment of m jobs to their m slots. Such an assignment can be found in polynomial time, e.g., by the Hungarian algorithm. □

We conclude this section by observing that hardness of computing $\sum$-K and $\sum$-S rules can be deduced from the hardness of computing Kemeny rankings [11].

Proposition 3.6. *Computing $\sum$-K and $\sum$-S is* NP*-hard even for $n = 4$ agents and when all jobs have the same unit size.*

3.2 L_p-norm of Delay Costs, $p > 1$

We show NP-hardness first for two agents, and, second, for unit jobs. The first proof works also for $p = \infty$, i.e., for max-$\{T, E, D\}$.

Theorem 3.7. *For each $p > 1$, finding a schedule returned by L_p-$\{T, E, D\}$ is* NP*-hard, even for two agents.*

Theorem 3.8. *For each delay cost $f \in \{T, E, D, SD\}$, finding a schedule returned by* max-f *is* NP*-hard, even for unit-size jobs.*

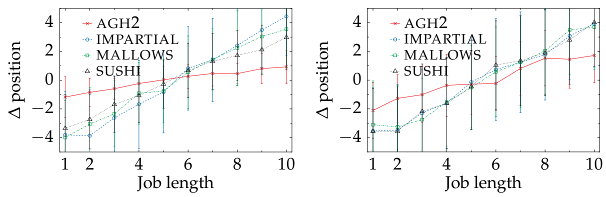
4 EXPERIMENTAL EVALUATION

The goal of our experimental evaluation is, first, to demonstrate that, while most of the problems are NP-hard, an Integer Linear Programming (ILP) solver finds optimal solutions for instances with reasonable sizes. Second, to quantitatively characterize the impact of collective scheduling compared to the base social choice methods. Third, to compare schedules built with different approaches (cost functions and axioms). We use tardiness T as a representative cost function: it is NP-hard in both Σ and max aggregations; and easy to interpret.

Settings. A single *experimental scenario* is described by a profile with preferred schedules of the agents and by a maximum length of a job $p_{\max}$. We instantiate the preferred schedules of agents using PrefLib [21]. We treat PrefLib's candidates as jobs. We use datasets where the agents have strict preferences over all candidates. We restrict to datasets with both large number of candidates and large number of agents: we take two datasets on AGH course selection (agh1 with 9 candidates and 146 agents; and agh2 with 7 candidates and 153 agents) and sushi dataset with 10 candidates and 5000 agents. Additionally, we generate preferences using the Mallows [20] model (mallows) and Impartial Culture (impartial), both with 10 candidates and 500 agents. We use three different values for $p_{\max}$: 10, 20 and 50. For each experimental scenario we generate 100 instances—in each instance pick the lengths of the jobs uniformly at random between 1 and $p_{\max}$ (in separate series of experiments we used exponential and normal distributions; we found similar trends to the ones discussed below). For each scenario, we present averages and standard deviations over these 100 instances.

Computing Optimal Solutions. We use standard ILP encoding: for each pair of jobs (i, j), we introduce two binary variables $prec_{i,j}$ and $prec_{j,i}$ denoting precedence: $prec_{i,j} = 1$ iff i precedes j in the schedule. ($prec_{i,j} + prec_{j,i} = 1$ and, to guarantee transitivity of $prec$, for each triple i, j, k, we have $prec_{i,j} + prec_{j,k} - prec_{i,k} \leq 1$). We run Gurobi solver on a 6-core (12-thread) PC. An agh instance takes, on the average, less than a second to solve, while a sushi instance takes roughly 20 seconds. In a separate series of experiments, we analyze the runtime on impartial instances as a function of number of jobs and number of voters. A 20 jobs, 500 voters instance with $\sum$-T goal takes 8 seconds; while a max-T goal takes two minutes. A 10 jobs, 5000 voters takes 8 seconds with $\sum$-T goal and 28 seconds with max-T goal. Finally, 20 jobs, 5000 voters take 23 seconds for with $\sum$-T and 20 minutes with max-T. For 30 jobs, the solver does not finish in 60 minutes. Running times depend thus primarily on the number of jobs and on the goal. We conclude that, while the problem is strongly NP-hard, it can be solved in practice for thousands of voters and up to 20 jobs. We consider these running times to be satisfactory: first, for a population it might be difficult to meaningfully express preferences for dozens of jobs [22] (therefore, the decision maker would probably combine jobs before eliciting preferences); second, gathering preferences takes non-negligible time; and, finally, in our motivating examples (public works, lecture hall) individual jobs last hours to weeks.

Analysis of the Results. First, we analyze job's rank as a function of its length. We compute a reference collective schedule for an instance with the same agents' preferences, but unit-size jobs (it thus corresponds to the classic preference aggregation problem with Σ-T or max-T goal). We then compute and analyze the collective schedules. Over 100 instances, as jobs' durations are assigned randomly, all the jobs' durations should be in the preferred schedules in, roughly, all positions. Thus, *on the average*, short jobs should be executed earlier, and long jobs later than in the reference schedule (in contrast, in any single experiment, if a large majority puts a short job at the end of their preferred schedules, the job is



Σ-T max-T

Figure 2: The average change in jobs' position. A point (x, y) in the plot denotes that a job of length x is on the average scheduled by y positions later than when we ignore jobs' durations. $p_{\max} = 10$ ($p_{\max} \in \{20, 50\}$ show very similar trends.)

Dataset	PTA C. Paradox		PTA Copeland $\cdot/\cdot$		ΔGini
	Σ-T	max-T	Σ-T	max-T	
AGH1	6%	15%	1.03	1.23	0.07
AGH2	5%	18%	1.03	1.28	0.12
SUSHI	7%	24%	1.02	1.22	0.06
IMPARTIAL	3%	8%	1.00	1.01	0.00
MALLOWS	10%	24%	1.03	1.21	0.08

Table 1: "PTA C. Paradox" gives the mean frequencies of violating the PTA Condorcet principle for optimal solutions for Σ-T and max-T. "PTA Copeland $\cdot/\cdot$" denotes the ratio of sum/max T for PTA Copeland's schedule to their optimums. "ΔGini" shows the average of differences in the Gini indices: Gini(max-T) - Gini(Σ-T).

not automatically advanced). To confirm this hypothesis, for each instance and each job we compare its position to the position in the reference schedule. Figure 2 shows the average position change as a function of the job lengths. In collective schedules, short jobs (e.g., of size 1) are advanced, on the average, 2-4 positions in the schedule, compared to schedules corresponding to the standard preference aggregation problem. The experiments thus confirm that the lengths of the jobs have profound impact on the schedule.

Second, we check how frequent are PTA-Condorcet paradoxes. For each instance, we counted how many out of $\binom{m}{2}$ job pairs are scheduled in a non-PTA-Condorcet consistent order. Table 1 shows that both Σ-T and max-T often violate the PTA Condorcet principle. Table 1 also shows the average ratio between the tardiness of the schedule returned by the PTA Copeland's rule, and the tardiness of optimal Σ and max schedules. These ratios are small: roughly 3% degradation for Σ and 24% for max. Thus, though PTA Copeland's rule does not explicitly optimize max-T and Σ-T, on average, it returns schedules close to the optimal for these criteria.

Third, we analyze how fair are Σ-T and max-T. We analyzed Gini indices of the vectors of agents' tardiness. Table 1 shows that, interestingly, Σ-T is more fair (smaller average Gini index), even though max-T seemingly cares more about less satisfied agents. Yet, the focus of max-T on the worst-off agent makes it effectively ignore all the remaining agents, increasing the societal inequality.

5 DISCUSSION AND CONCLUSIONS

The principal contribution of this paper is conceptual—we introduce the notion of the *collective schedule*. We believe that collective scheduling addresses natural problems involving jobs or events having diverse impacts on the society. Such problems do not fit well into existing scheduling models. We demonstrated how to formalize the notion of the collective schedule by extending well-known methods from social choice. While collective scheduling is closely related to preference aggregation, these methods have to be extended to take into account lengths of jobs. Notably, we proposed to judge the quality of a collective schedule by comparing the jobs' completion times between the collective and the agents' preferred schedules. We also showed how to extend the Condorcet principle to take into account lengths of jobs.

We conclude that there is no clear winner among the proposed scheduling mechanisms. Similarly, in the classic voting, there is no clear consensus regarding which voting mechanism is the best. For example, we showed that the comparison of the cost-based and PTA-Condorcet-based scheduling exposes a tradeoff between reinforcement and the PTA Condorcet principle. Thus, the question which mechanism to choose is, for example, influenced by the subjective assessment of the mechanism designer with respect to which one of the two properties she considers more important.

Our main conclusion from the theoretical analysis of computational complexity and from the experimental analysis is that using cost-based scheduling methods is feasible only if the sizes of the input instances are moderate (though, these instances may represent many realistic situations). In contrast, PTA Condorcet-based methods are feasible even for large instances. We drew a boundary between NP-hard and polynomial-time solvable problems. In several cases, problems become NP-hard with non-unit jobs, therefore showing additional complexity stemming from scheduling, as opposed to standard voting. Moreover, our experiments suggest that there is a clearly visible difference between schedules returned by different methods of collective scheduling.

Both scheduling and social choice are well-developed fields with a plethora of models, methods and results. It is natural to consider more complex scheduling models in the context of collective scheduling, such as processing several jobs simultaneously (multiple processors with sequential or parallel jobs), jobs with different release dates or dependencies between jobs. Each of these extensions raises new questions on computability/approximability of collective schedules. Another interesting direction is to derive desired properties of collective schedules (distinct from PTA-Condorcet), and then formulate scheduling algorithms satisfying them.

ACKNOWLEDGMENTS

This research has been partly supported by the Polish National Science Center grant Sonata (UMO-2012/07/D/ST6/02440), a Polonium grant (joint programme of the French Ministry of Foreign Affairs, the Ministry of Science and Higher Education and the Polish Ministry of Science and Higher Education), the French ANR grant "CoCoRICo-CoDec" (ANR-14-CE24-0007-01), and project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651). Piotr Skowron was also supported by a Humboldt fellowship for postdoctoral researchers.

REFERENCES

[1] A. Agnetis, J. Billaut, S. Gawiejnowicz, D. Pacciarelli, and A. Soukhal. *Multiagent Scheduling: Models and Algorithms.* Springer, 2014.

[2] K. J. Arrow, A. Sen, and K. Suzumura, editors. *Handbook of Social Choice & Welfare*, volume 2. Elsevier, 2010.

[3] G. Benade, S. Nath, A. Procaccia, and N. Shah. Preference elicitation for participatory budgeting. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI-2017)*, pages 376–382, 2017.

[4] P. Brucker. *Scheduling Algorithms.* Springer, 2006.

[5] Y. Cabannes. Participatory budgeting: a significant contribution to participatory democracy. *Environment and Urbanization*, 16(1):27–46, 2004.

[6] I. Caragiannis, E. Hemaspaandra, and L. A. Hemaspaandra. Dodgson's rule and Young's rule. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 2. Cambridge University Press, 2015.

[7] E. Celis, D. Straszak, and N. Vishnoi. Ranking with fairness constraints. Technical Report arXiv:1704.06840, arXiv.org, 2017.

[8] J. Colomer. Ramon Llull: from 'Ars electionis' to social choice theory. *Social Choice and Welfare*, 40(2):317–328, 2013.

[9] J. Du and J. Leung. Minimizing total tardiness on one machine is np-hard. *Mathematics of operations research*, 15(3):483–495, 1990.

[10] P. F. Dutot, F. Pascual, K. Rzadca, and D. Trystram. Approximation algorithms for the multiorganization scheduling problem. *IEEE Transactions on Parallel and Distributed Systems*, 22(11), 2011.

[11] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web (WWW-2001)*, pages 613–622. ACM, 2001.

[12] B. Fain, A. Goel, and K. Munagala. The core of the participatory budgeting problem. In *Proceedings of the 12th Conference on Web and Internet Economics (WINE-2016)*, pages 384–399, 2016.

[13] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. Multiwinner voting: A new challenge for social choice theory. In U. Endriss, editor, *Trends in Computational Social Choice.* AI Access, 2017.

[14] F. Fischer, O. Hudry, and R. Niedermeier. Weighted tournament solutions. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 2. Cambridge University Press, 2015.

[15] A. Goel, A. Krishnaswamy, S. Sakshuwong, and T. Aitamurto. Knapsack voting: Voting mechanisms for participatory budgeting. Manuscript, 2016.

[16] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[17] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

[18] J. Lang and L. Xia. Voting over multiattribute domains. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 9. Cambridge University Press, 2015.

[19] J. Levin and B. Nalebuff. An introduction to vote-counting schemes. *Journal of Economic Perspectives*, 9(1):3–26, 1995.

[20] C. L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1-2):114–130, June 1957.

[21] N. Mattei and T. Walsh. Preflib: A library for preferences http: //www.preflib.org. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT-2013)*, pages 259–270, 2013.

[22] G. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[23] F. Pascual, K. Rzadca, and P. Skowron. Collective schedules: Scheduling meets computational social choice. Technical Report arXiv:1803.07484 [cs.GT], arXiv.org, 2018.

[24] PBP. The Participatory Budgeting Project. www.participatorybudgeting.org/, 2018.

[25] P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence*, 241:191–216, 2016.

[26] P. Skowron, M. Lackner, M. Brill, D. Peters, and E. Elkind. Proportional rankings. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-2017)*, pages 409–415, 2017.

[27] B. Vöcking. Selfish load balancing. In *Algorithmic Game Theory*. Cambridge, 2007.

[28] H. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.