Two Dimensional Optimal Mechanism Design for a Sequencing Problem

Ruben Hoeksma and Marc Uetz

University of Twente, Dept. Applied Mathematics, P.O. Box 217, 7500AE Enschede, The Netherlands, {r.p.hoeksma, m.uetz}@utwente.nl

Abstract. We propose an optimal mechanism for a sequencing problem where the jobs' processing times and waiting costs are private. Given public priors for jobs' private data, we seek to find a scheduling rule and incentive compatible payments that minimize the total expected payments to the jobs. Here, incentive compatible refers to a Bayes-Nash equilibrium. While the problem can be efficiently solved when jobs have single dimensional private data, we here address the problem with two dimensional private data. We show that the problem can be solved in polynomial time by linear programming techniques, answering an open problem in [13]. Our implementation is randomized and truthful in expectation. The main steps are a compactification of an exponential size linear program, and a combinatorial algorithm to decompose feasible interim schedules. In addition, in computational experiments with random instances, we generate some more insights.

1 Introduction & Contribution

In this paper, we address an optimal mechanism design problem for a sequencing problem introduced by Heydenreich et al. in [13]. While that paper mainly addresses the version with single dimensional private data, we focus on the case with two dimensional private data. Indeed, starting with the seminal paper by Myerson [16], optimal mechanism design with single dimensional private data is pretty well understood, also from an algorithmic point of view, e.g. [12], while algorithmic results for optimal mechanism design with multi dimensional private data have been obtained only recently, e.g. [1,3].

Our starting point is the **open problem** formulated in [13], who 'leave it as an open problem to identify (closed formulae for) optimal mechanisms for the 2-d case.' Here, the '2-d case' refers to the problem of computing a Bayes-Nash optimal mechanism for the following sequencing problem on a single machine: There are n jobs with two dimensional private data, namely a cost per unit time w_j and a processing time p_j. Jobs need to be processed sequentially, and each job requires a compensation for the disutility of waiting. With given priors on the private data of jobs, the optimal mechanism seeks to minimize the total expected payments made to the jobs, while being Bayes-Nash incentive compatible. This problem is an abstraction of economic situations where clients queue for a single scarce resource (e.g., a specialized operation theatre), while the information on the urgency and duration to treat each client is private, yet known probabilistically. A concrete example are waiting lists for medical treatments in the Netherlands, see [14].

The **main contribution of this paper** is to answer the open problem in [13], by giving an optimal mechanism and showing that it can be computed and implemented in polynomial time. Our solution is based on linear programming techniques, and results in an optimal randomized mechanism. In that sense, we do not obtain analytic 'closed formulae' for the solution, and our results can be seen in the tradition of 'automated mechanism design' as proposed e.g. by Conitzer and Sandholm [4,20], in that the design of the mechanism itself is based on (integer) linear programming.

The major **technical contributions** are twofold: The first is the compactification of an exponential size linear programming formulation of the mechanism design problem, which is the crucial ingredient that allows a polynomial time algorithm to compute payments and a so-called interim schedule. The second is an algorithm that allows to compute, in polynomial time, the implementation for the given interim schedule. To that end, we give a combinatorial $O(n^3 \log n)$ algorithm that computes, for any given point s in the single machine scheduling polytope as defined by Queyranne [18], a representation of s as convex combination of $\leq n$ vertices. This result generalizes a similar result for the permutahedron by Yasutake et al. [23], but in contrast to that paper, our algorithm follows the geometric construction as proposed by Grötschel et al. in [11, Thm. 6.5.11].

Finally, again in the flavor of automated mechanism design, we present **computational results** based on the (integer) linear programming formulations. These computations have the primary goal to test and validate hypotheses on the structure of solutions. Our computations, based on randomly generated instances, show that optimal mechanisms in the two dimensional setting do *not* share several of the nice properties of the solutions to the single dimensional problem: The scheduling rules of optimal Bayes-Nash incentive compatible mechanisms are not necessarily *iia* (a desirable property to be defined later), and neither do optimal Bayes-Nash mechanisms allow an implementation in dominant strategies. This in contrast to the single dimensional problem which has these properties [13,5].

We conclude this section with a brief discussion of our result in relation to the recent results of Cai et al. [3]. Apart from some methodological similarities in Section 4, we specifically ask the question if the problem that we consider here fits into the general framework presented there. This is not the case: In order to formulate the problem considered here in that context, we can either represent a schedule as an assignment of n jobs to n slots, in which case the problem has informational externalities because the utility of a job for a given slot then depends on the types (specifically, processing times) of other jobs. Or, we can represent a schedule as a vector of starting times, but then the feasibility of such vector depends on the types (specifically, processing times) of jobs. Either way, we leave the framework of [3], and we do not see a simple way to fix this.

2 Definitions, Preliminary & Related Results

We consider a sequencing (or single machine scheduling) problem with n agents denoted $j \in N$, each owning a job with weight w_j and processing time p_j. We identify jobs with agents. The jobs need to be processed (sequenced) on a single machine, with the interpretation that w_j is job j's individual cost for waiting one unit of time, while p_j is the time it requires to process job j. In a schedule that yields a start time s_j for job j, the cost for waiting is $w_j s_j$. The *type* of a job j is the vector of weight and processing time, denoted $t_j = (w_j, p_j)$. Note that the type is two dimensional. With t_j being public, the total waiting cost is well known to be minimized by sequencing the jobs in order of non increasing ratios w_j/p_j, also known as Smith's rule [21].

In the setting we consider here, weight and processing time are private to the agent that owns the job. There is a public belief about this private information, which is[1]

- the types that job j might have are $T_j = \{t_j^1, \dots, t_j^{m_j}\}$, and
- the probability of job j having type t_j^i is $\varphi_j(t_j^i)$, $i = 1 \dots, m_j$.

By $T = T_1 \times \ldots \times T_n$ we denote the type space of all jobs, with $t = (t_1, \dots, t_n) \in T$. Define $m := \sum_{j \in N} m_j$, and note that $m \geq n$. For a type $t_j^i \in T_j$, we let w_j^i and p_j^i be the corresponding weight and processing time, respectively. We sometimes abuse notation by identifying i with t_j^i, to avoid excessive notation. Moreover, (t_j, t_{-j}) denotes a type vector where t_j is the type of job j and t_{-j} are the types of all jobs except j, with $t_{-j} \in T_{-j} := \prod_{k \neq j} T_k$. For given $t \in T$ and $K \subseteq N$, we also define the shorthand notation $\varphi(t_K) := \prod_{k \in K} \varphi_k(t_k)$ for the product distribution of the types of jobs in K, particularly $\varphi(t_{-j}) := \prod_{k \neq j} \varphi_k(t_k)$.

We assume, just like [13], that the mechanism designer needs to compensate the jobs for waiting by a payment π_j that the job receives. We seek to compute and implement a (direct) mechanism, consisting of a scheduling rule and a payment rule, assigning to any $t \in T$ a permutation $\sigma(t)$ of jobs which yields a schedule $s^\sigma(t)$ of start times, together with compensation payments $\pi(t)$. In the mechanism design and auction literature, for obvious reasons, what is a scheduling rule here is referred to as *allocation rule*. Clearly, jobs may have an incentive to strategically misreport their true types in order to receive higher compensation payments. The optimal mechanism that we seek, however, is one that minimizes the total payments made to the jobs. Since reporting a processing time smaller than the true processing time is verifiable while processing a job, we assume, again like [13], that only larger than the true processing times can be reported by any job.

It is Myerson's revelation principle [16] that makes this problem (and many others [22]) amendable to optimization techniques: it asserts that it is no loss of generality to restrict to *truthful* mechanisms, where each job maximizes utility by reporting the type truthfully. In the considered setting with given priors on

[1] Note that the discrete type space make the problem amendable for (I)LP techniques.

private data, a mechanism is truthful, or more precisely *Bayes-Nash incentive compatible*, if it fulfills the following, linear constraint

$$\pi_j^i - w_j^i E s_j^i \geq \pi_j^{i'} - w_j^i E s_j^{i'} \quad \text{for all jobs } j \text{ and types } t_j^i, t_j^{i'} \in T_j\,.$$

Here, Es_j^i and π_j^i are defined as expected start time and payment for job j when he reports to be of type t_j^i, where the expectation is taken over all (truthful) reports of other jobs $t_{-j} \in T_{-j}$. Then, assuming utilities are quasi-linear, the expected utility for job j with true type t_j^i is $\pi_j^i - w_j^i E s_j^i$ for reporting truthfully, while a false report $t_j^{i'}$ yields expected utility $\pi_j^{i'} - w_j^i E s_j^{i'}$. The scheduling rule corresponding to a Bayes-Nash incentive compatible mechanism is called Bayes-Nash *implementable*.

Moreover, in order to have the problem bounded, we make the standard assumption that the expected utilities of truthful jobs are nonnegative, known as *individual rationality*,

$$\pi_j^i - w_j^i E s_j^i \geq 0\,.$$

It is interesting to ask if a scheduling rule (more generally, allocation rule) can even be implemented in the stronger *dominant strategy* equilibrium; in [15] the equivalence of Bayes-Nash and dominant strategy implementations is shown for the case of standard single unit private value auctions. In a dominant stategy equilibrium, reporting the true type maximizes the utility of a job not only in expectation but for *any* report t_{-j} of the other jobs, that is, $\pi_j(t_j^i, t_{-j}) - w_j^i s_j(t_j^i, t_{-j}) \geq \pi_j(t_j^{i'}, t_{-j}) - w_j^i s_j(t_j^{i'}, t_{-j})$ for all $t_j^i, t_j^{i'} \in T_j$ and all $t_{-j} \in T_{-j}$. The latter obviously implies the former, but generally not vice versa [10].

In the setting considered here, a mechanism is Bayes-Nash implementable if and only if the expected start times Es_j^i are monotonically increasing in the reported weight w_j^i. The same result holds for dominant strategy implementability, but then the start times $s_j(t_j^i, t_{-j})$ need to be monotonically increasing in the reported weight w_j^i, for all $t_{-j} \in T_{-j}$. This is a standard result in single-dimensional mechanism design [17], but it is also true for the 2-dimensional problem considered here [13]. The problem to find an *optimal* mechanism for the 2-dimensional mechanism design problem was left open in [13].

For the single dimensional mechanism design problem, where only weights are private information and processing times are known, the optimal mechanism has a simple structure: It is Smith's rule, but with respect to virtual instead of the original weights w_j; see [13] for details. In particular, in that case the optimal Bayes-Nash incentive compatible mechanism can be computed and implemented in polynomial time, and it can even be implemented (with the same expected cost) in dominant strategies [5].

3 Problem Formulations & Linear Relaxation

Let us start by giving a natural, albeit exponential size ILP formulation for the mechanism design problem at hand. Recall that $s_j^\sigma(t)$ denotes the start time of job j if the permutation of jobs is σ under type vector t. We use the natural variables

$$x_\sigma(t) = \begin{cases} 1 & \text{if for type vector } t \text{ permutation } \sigma \text{ is used}\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Then the formulation reads as follows.

$$\min \sum_{j\in N} \sum_{i\in T_j} \varphi_j^i \pi_j^i \tag{1}$$

$$\pi_j^i \geq w_j^i E s_j^i \qquad \forall j \in J, i \in T_j \tag{2}$$

$$\pi_j^i \geq \pi_j^{i'} - w_j^i (E s_j^{i'} - E s_j^i) \qquad \forall j \in N, i \in T_j, i' \in T_j, p_j^{i'} \geq p_j^i \tag{3}$$

$$E s_j^i = \sum_{t_{-j}\in T_{-j}} \varphi(t_{-j}) \sum_\sigma x_\sigma(t_j^i, t_{-j}) s_j^\sigma(t_j^i, t_{-j}) \qquad \forall j \in N, t_j^i \in T_j \tag{4}$$

$$\sum_\sigma x_\sigma(t) = 1 \qquad \forall t \in T \tag{5}$$

$$x_\sigma(t) \in \{0, 1\} \qquad \forall \sigma \in \Sigma, t \in T \tag{6}$$

Here we use the shorthand notation φ_j^i for $\varphi_j(t_j^i)$, and Σ is the set of all permutations of N. The objective (1) is the total expected payment. Constraints (2) and (3) are the individual rationality and incentive compatibility constraints: (2) requires the expected payment to at least match the expected cost of waiting when the type is t_j^i, and (3) makes sure that the expected utility is maximized when reporting truthfully. Values $E s_j^i$ are also referred to as *interim schedule*, and equations (4) are the feasibility constraints for interim schedules, expressing the fact that the expected starting times in the interim schedule need to comply with the scheduling rule encoded by x. While the input size of the mechanism design problem is $\mathrm{O}(m)$, this ILP formulation is colossal as the number of variables $x_\sigma(t)$ is $|T|\, n!$ with $|T| = \prod_j m_j$.

Observe that, for given type vector t, the vectors $s^\sigma(t)$ are the vertices of the well known single machine scheduling polytope $Q(t)$ [6,18], only here we consider start instead of completion times. In other words, $s^\sigma(t)$ are the start times of permutation schedules. Recall from [18] that the polytope $Q(t)$ is defined by

$$\sum_{j\in K} p_j(t) s_j(t) \geq \frac{1}{2} \left(\sum_{j\in K} p_j(t) \right)^2 - \frac{1}{2} \sum_{j\in K} p_j(t)^2 \qquad \forall K \subseteq N \tag{7}$$

$$\sum_{j\in N} p_j(t) s_j(t) = \frac{1}{2} \left(\sum_{j\in N} p_j(t) \right)^2 - \frac{1}{2} \sum_{j\in N} p_j(t)^2 \;, \tag{8}$$

where we use $p_j(t)$ to denote the processing time of job j in type profile t. The last equality excludes schedules with idle time. Allowing randomization, any point of $Q(t)$ represents feasible expected start times. Note that the scheduling polytope $Q(t)$ is a polymatroid via variable transform to $p(t)s(t)$. In this particular case, both optimization and separation for $Q(t)$ can be done in time $\mathrm{O}(n^2)$ [7,18].

3.1 Linear Ordering Formulation

It turns out to be convenient for our purpose to consider another formulation, namely using linear ordering variables d_{kj}, with intended meaning

$$d_{kj}(t) = \begin{cases} 1 & \text{if for type vector } t \text{ we use a schedule where job } k \text{ precedes job } j\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Using linear ordering variables yields the following formulation of the optimal mechanism design problem.

$$\min \sum_{j\in N}\sum_{i\in T_j} \varphi_j^i \pi_j^i \tag{9}$$

$$\pi_j^i \geq w_j^i E s_j^i \qquad \forall j,i \tag{10}$$

$$\pi_j^i \geq \pi_j^{i'} - w_j^i(Es_j^{i'} - Es_j^i) \qquad \forall j,i,i' \tag{11}$$

$$Es_j^i = \sum_{t_{-j}\in T_{-j}} \varphi(t_{-j}) s_j(t_j^i, t_{-j}) \qquad \forall j,i \tag{12}$$

$$s_j(t) = \sum_{k\in N} d_{kj}(t) p_k(t) \qquad \forall j,t \tag{13}$$

$$d_{jj}(t) = 0 \qquad \forall j,t \tag{14}$$

$$d_{kj}(t) + d_{jk}(t) = 1 \qquad \forall j,k,t\ j\neq k \tag{15}$$

$$d_{jk}(t) \geq 0 \qquad \forall j,k,t \tag{16}$$

$$d_{jk}(t) + d_{kl}(t) \leq 1 + d_{jl}(t) \qquad \forall j,k,l,t \tag{17}$$

$$d_{jk}(t) \in \{0,1\} \qquad \forall j,k,t\ . \tag{18}$$

Observe that, in contrast to the previous x_σ formulation, the number of variables $d_{jk}(t)$ now equals $n^2 \cdot |T|$. However this formulation is in general exponential as well, since the type space T can be exponential in m.

The vertices of $Q(t)$ are the solutions $s(t)$ of (13)-(18), and moreover, a vector of starting times $s(t)$ satisfies (13)-(16) if and only if it satisfies (7) and (8); see for instance [19, Thm. 4.1]. More specifically, via (13), the scheduling polytope $Q(t)$ is an affine image of both the linear ordering polytope (14)-(18) and its relaxation (14)-(16). This important observation is crucial for what follows, as we can continue to work with the relaxation (14)-(16) instead of (14)-(18).

3.2 Relaxation & Compactification

A linear relaxation of the optimal mechanism design problem (9)-(18) is obtained by dropping the last two sets of constraints (17) and (18). By moving from the ILP formulation to its LP relaxation, we in fact move from deterministic scheduling rules to randomized ones, which follows from our previous discussion about the equivalence of (13)-(16) and (7)-(8), as well as the fact that the scheduling polytope $Q(t)$ is an affine image of the relaxation (14)-(16) via (13).

In what follows we also combine (12) and (13) into just one constraint, and (17) and (18) are omitted. This gives us the following formulation.

$$\min \sum_{j\in N}\sum_{i\in T_j} \varphi_j^i \pi_j^i \tag{19}$$

$$\pi_j^i \geq w_j^i Es_j^i \qquad \forall j,i \tag{20}$$

$$\pi_j^i \geq \pi_j^{i'} - w_j^i(Es_j^{i'} - Es_j^i) \qquad \forall j,i,i' \tag{21}$$

$$Es_j^i = \sum_{t_{-j}\in T_{-j}} \sum_{k\in N} \varphi(t_{-j}) d_{kj}(t_j^i, t_{-j}) p_k(t_{-j}) \qquad \forall j,i \tag{22}$$

$$d_{jj}(t) = 0 \qquad \forall j,t \tag{23}$$

$$d_{kj}(t) + d_{jk}(t) = 1 \qquad \forall j,k,t,\ k\neq j \tag{24}$$

$$d_{kj}(t) \geq 0 \qquad \forall j,k,t\ . \tag{25}$$

We now focus on the projection to variables Es_j^i, that is, vectors $Es \in \mathbb{R}^m$ satisfying (22)-(25). These are interim schedules in the linear relaxation. Let us refer to this projection as the *relaxed interim scheduling polytope.* Notice that, even though it is a linear relaxation, (22)-(25) is still an exponential size formulation, as it depends on the size of T. The crucial insight is that, in the linear relaxation, this exponential size formulation is actually not necessary. Instead of using $d_{kj}(t)$ where $t \in T$, we propose an *LP compactification* by restricting to variables

$$d_{kj}(t_k, t_j)\,,$$

where t_k and t_j are the types of jobs k and j, respectively. This reduces the number of d_{kj}-variables to $\mathrm{O}(m^2)$, yielding a polynomial size formulation. Doing so, we obtain

$$Es_j^i = \sum_{k\in N}\sum_{t_k\in T_k} \varphi(t_k) d_{kj}(t_j^i, t_k) p_k(t_k) \qquad \forall j,i \tag{26}$$

$$d_{jj}(t_j, t_j) = 0 \qquad \forall j,t_j \tag{27}$$

$$d_{kj}(t_k,t_j) + d_{jk}(t_j,t_k) = 1 \qquad \forall j,k,t_j,t_k,\ k\neq j \tag{28}$$

$$d_{kj}(t_k,t_j) \geq 0 \qquad \forall j,k,t_j,t_k\ . \tag{29}$$

The following lemma is the core insight of the results in this paper.

Lemma 1. *The relaxed interim scheduling polytope defined by* (22)-(25) *can be equivalently described by* (26)-(29).

Proof. Let P be the projection of (22)-(25) to variables Es_j^i, and P' be the projection of (26)-(29) to variables Es_j^i. It is obvious that if $Es \in P'$, then $Es \in P$, simply by letting $d_{kj}(t) = d_{kj}(t_k, t_j)$, for all $t \ni t_k, t_j$. So all we need to show is that, if $Es \in P$, then $Es \in P'$. So let $Es \in P$ with corresponding $d_{kj}(t)$. Now define

$$d_{kj}(t_k,t_j) = \sum_{t\ni t_k,t_j} \frac{\varphi(t)}{\varphi(t_k)\varphi(t_j)} d_{kj}(t)\ \ ,$$

then the $d_{kj}(t_k, t_j)$ clearly satisfy (27)-(29). Moreover, we have for all $j \in N$ and $i \in T_j$,

$$\begin{aligned} Es_j^i &= \sum_{t_{-j} \in T_{-j}} \sum_{k \in N} \varphi(t_{-j}) d_{kj}(t_j^i, t_{-j}) p_k(t_{-j}) \\ &= \sum_{k \in N} \sum_{t \ni t_j^i} \frac{\varphi(t)}{\varphi(t_j^i)} d_{kj}(t) p_k(t) \\ &= \sum_{k \in N} \sum_{t_k \in T_k} \varphi(t_k) \sum_{t \ni t_k, t_j^i} \frac{\varphi(t)}{\varphi(t_j^i)\varphi(t_k)} d_{kj}(t) p_k(t_k) \\ &= \sum_{k \in N} \sum_{t_k \in T_k} \varphi(t_k) d_{kj}(t_k, t_j) p_k(t_k) \ , \end{aligned}$$

which is exactly the RHS of (26). □

We conclude with the following theorem.

Theorem 1. *Computing an optimal interim schedule together with optimal payments for the mechanism design problem can be done in time polynomial in the input size of the problem.*

Proof. The input size of the problem is $\Theta(m)$. The formulation (19)-(21) together with (26)-(29) has $O(m^2)$ variables and $O(m^2)$ constraints. Hence, this linear program can be solved in time polynomial in the input size. □

Now that we can compute optimal payments and interim schedule, two issues remain: The first is the interpretation of Theorem 1, because it is based on a relaxation and has a reduced number of variables. The second, which is an issue because we consider a relaxation, is the actual implementation of the optimal mechanism: We have to link the computed solution of the LP relaxation, specifically the computed interim schedule Es, to a (randomized) schedule $s(t)$ for any given type profile $t \in T$. The first issue is discussed next, the second in Section 4.

3.3 Discussion of the Result

We consider a true relaxation of the linear ordering polytope by dropping triangle and integrality constraints, yet the affine image of the variables $d_{kj}(t)$, respectively $d_{kj}(t_k, t_j)$, via (13) still yields a feasible point in the scheduling polytope. This allows us to interpret the solution as a (randomized) schedule; this is discussed in the next section. Also, we have drastically reduced the number of variables. It seems that thereby we are reducing the (number of) feasible mechanisms, because the variables $d_{kj}(t_k, t_j)$ only depend on the types of jobs k and j, while $d_{kj}(t)$ depends on the whole type vector t. For deterministic mechanisms, this is also known as *iia-property* [13].

Definition 1 (iia). *A deterministic scheduling rule is* independent of irrelavent alternatives*, or iia, if the relative order of two jobs does not depend on anything but the types of those two jobs, that is,* $d_{kj}(t) = d_{kj}(t_k, t_j)$*. We call a mechanism for which the scheduling rule is iia, an iia*-mechanism.

Lemma 1 shows that the reduction of variables is in fact no loss of generality for the relaxation. Interestingly, it *is* a loss of generality for the linear ordering polytope itself, respectively for the deterministic optimal mechanism design problem (9)-(18): Theorem 3 in Section 5 shows an optimality gap in general. With this in mind, a possible interpretation of Lemma 1 would be that the restriction to *iia*-mechansims is no loss of generality once randomization is allowed. But this interpretation is problematic, as the variables d_{kj} in the relaxation cannot in general be interpreted as the probability of job k preceding job j: By definition of the relaxation, neither the vector of variables $d_{kj}(t_k, t_j)$ nor $d_{kj}(t)$ do necessarily lie in the linear ordering polytope; see e.g. [8]. A detour via the scheduling polytope, however, fixes this.

4 Implementation

Recall from the previous discussion that the fractional solution in variables d_{kj} as suggested by the LP relaxation cannot in general be decomposed into linear orders, as it may lie outside the linear ordering polytope. Yet by taking the detour via the scheduling polytope we can easily fix this.

First, observe that for given solution Es and $d_{jk}(t_j, t_k)$, and fixed type vector t we can compute a corresponding vector of start times $s(t)$ by

$$s_j(t) = \sum_{k \in N} d_{kj}(t_j, t_k) p_k(t_k) \ \text{ for all } j \, .$$

Recall that $s(t)$ is simply a point in the scheduling polytope $Q(t)$ defined in (7) and (8), and the dimension of the scheduling polytope is $n - 1$. It follows from Caratheodory's Theorem that $s(t)$ can be expressed as the convex combination of at most n vertices of $Q(t)$, that is, permutation schedules. In what follows, we describe a combinatorial algorithm to compute this representation, where for convenience, we drop the dependence on t.

A straightforward adaptation of a recent algorithm by Yasutake et al. [23] for the permutahedron results in an $\mathrm{O}(n^2)$ algorithm. However, this outputs a convex combination of $\mathrm{O}(n^2)$ vertices, while we know that a convex combination of at most n vertices exists. Therefore, we follow a geometric approach proposed by by Grötschel, Lovász and Schrijver in [11, Thm. 6.5.11]: Given some $s \in Q$, pick a (random) vertex v of Q, and compute the point $s' \in Q$ where the half-line through v and s leaves Q. This point lies on a facet of Q, and we can recurse on that facet[2]. However, we need a way to efficiently compute s' and a facet on

[2] Note that, independent of our work, and apparently also independent of [11], a similar decomposition algorithm is also suggested by Cai et al. [2,3]. References [11,2,3] do not result in combinatorial algorithms. However, in contrast to our work, they do address arbitrary polytopes.

which it lies. This can be done with an algorithm described by Fonlupt and Skoda in $O(n^8)$ time [9]. Here, we improve on this result for the scheduling polytope and give a simple algorithm that runs in $O(n^2 \log n)$ time. The total time for computing the representation of $s(t)$ as convex combination of $\leq n$ permutation schedules will be $O(n^3 \log n)$.

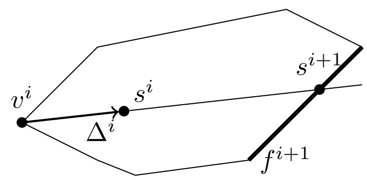

Fig. 1. Illustration of one iteration of Algorithm 1.

Algorithm 1 (Decomposition Algorithm) *For a given point $s^i \in Q$ (in iteration i), order the jobs ascending in their start time s^i_j and define vertex v^i corresponding to that permutation schedule. We aim to find a point $s^{i+1} \in Q$ on a facet of Q such that $s^i = \lambda^i v^i + (1-\lambda^i)s^{i+1}$, for some $\lambda^i \in [0,1]$. Let $\Delta^i = s^i - v^i$. Then $\delta_{\max} := \max_{\delta \geq 0}\{v^i + \delta\Delta^i \in Q\}$, so that $s^{i+1} = v^i + \delta_{\max}\Delta^i$ and $\lambda^i = (1 - 1/\delta_{\max})$. If we now compute a facet f^{i+1} of Q containing s^{i+1}, we recurse with $s^{i+1} \in f^{i+1}$, and terminate after n iterations.*

The algorithm is illustrated in Figure 1. The following lemma is a consequence of our choice of vertex v^i; it shows that Algorithm 1 is well defined.

Lemma 2. *Both $v^i \in f^i$ and $s^i \in f^i$ (where $f^0 := Q$), hence $s^{i+1} \in f^i$.*

We are left to show that, in any iteration, computing s^{i+1} and f^{i+1} can be done in time $O(n^2 \log n)$. The crucial idea is that the set K^{i+1} that defines facet f^{i+1} can be computed from one of the $O(n^2)$ different orderings of the elements of the vectors on the half-line $L = \{v^i + \delta\Delta^i \mid \delta \geq 0\}$. There are no more than $O(n^2)$ such orderings, because the relative order of any two elements x_j an x_k, with $x \in L$, can change at most once while moving along L, by linearity.

Now imagine that the target point s^{i+1} lies on a facet defined by set $K^{i+1} \subseteq N$. Then, assuming for simplicity of notation that the ordering of elements of s^{i+1} is $s^{i+1}_1 \leq \cdots \leq s^{i+1}_n$, the set K^{i+1} appears as as one of the n nested sets $[k] := \{1, \ldots, k\}$, $k = 1, \ldots, n$. This follows directly from the simple separation algorithm for the scheduling polytope Q [18].

Since we do not know a priori which ordering the elements of s^{i+1} have, the simplest algorithm is to try them all, which works because we know that there are no more than $O(n^2)$ such orders for all points of L. Each of them gives n candidates for K^{i+1}, and computing their intersection with L yields s^{i+1} as the intersection point closest to s^i. This argument directly yields a $O(n^4)$

algorithm. With a more clever bookkeeping of the candidate sets, we end up with the following lemma; for details we refer to the full version of this paper.

Lemma 3. *The computation of vector* s^{i+1} *with* $s^i = \lambda^i v^i + (1 - \lambda^i)s^{i+1}$*, and facet* $f^{i+1} \ni s^{i+1}$ *of* Q *in Algorithm 1 can be done in time* $O(n^2 \log n)$*.*

We can now conclude.

Theorem 2. *A point* $s \in Q$ *can be decomposed into the convex combination of at most* n *vertices (= permutation schedules) of* Q *in* $O(n^3 \log n)$ *time.*

5 Computational Results

We have implemented all models discussed in this paper; let us briefly comment on these experiments. As already mentioned, the most straightforward ILP formulation (1)-(6) for the deterministic mechanism design problem is colossal, which is confirmed by large computation times. In comparison, the linear ordering formulation (9)-(18), even though exponential in size as well, yields an average improvement in computation times of a factor 3-40 for small scale instances, depending on the model considered. In particular, the latter allows to drastically reduce the number of variables and constraints for *iia*-mechanisms, while the former formulation doesn't.

We end this short computational section by listing the following insights that we could obtain through generating random instances, and comparing the corresponding optimal solutions for different models. More detailed computational results are deferred to a full version of this paper.

Theorem 3. *Optimal deterministic mechanisms for both Bayes-Nash and dominant strategy implementations, in general do not satisfy the iia condition.*[3]

Theorem 4. *The optimal deterministic Bayes-Nash mechanism is generally not implementable in dominant strategies.*

Theorem 5. *Randomized Bayes-Nash mechanisms perform better than deterministic Bayes-Nash mechanisms in terms of total optimal payment.*

Proof. These theorems follow from instances which exhibit corresponding optimality gaps; they are deferred to a full version of this paper. □

6 Concluding Remarks

Our solution is randomized and truthful in expectation. The complexity to find an optimal deterministic mechanism remains open, and it is not even clear if the decision problem is contained in NP. An interesting future path to follow is to worst-case analyze the gaps between the solutions of different models.

Acknowledgements. Thanks to Maurice Queyranne for pointing us to [23], to Jelle Duives for his contribution in the experiments, and to Walter Kern, Marc Pfetsch, Rudolf Müller, and Gergely Csapó for helpful discussions. Also thanks to the thoughtful comments by an anonymous referee.

[3] Note: The example given in [13] to prove the same theorem is flawed.

References

1. S. Alaei, H. Fu, N. Haghpanah, J. Hartline, and A. Malekian. Bayesian Optimal Auctions via Multi- to Single-agent Reduction. In: Proc. 13th EC, 2012, p. 17
2. Y. Cai, C. Daskalakis and S.M. Weinberg. An Algorithmic Characterization of Multi-Dimensional Mechanisms. In: Proc. 44th STOC, 2012.
3. Y. Cai, C. Daskalakis and S.M. Weinberg. Optimal Multi-Dimensional Mechanism Design: Reducing Revenue to Welfare Maximization. In: Proc. 53rd FOCS, 2012.
4. V. Conitzer and T. Sandholm. Complexity of mechanism design. In: Proc. 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2002), 103-110.
5. J. Duives, B. Heydenreich, D. Mishra, R. Müller, and M. Uetz. Optimal Mechanisms for Single Machine Scheduling. Manuscript, 2012.
6. M.E. Dyer and L.A. Wolsey. Formulating the single machine sequencing problem with release dates as a mixed integer program. Disc. Appl. Math. 26, 1990, 255-270.
7. J. Edmonds. Matroids and the greedy algorithm. Math. Prog. 1, 1971, 127-136.
8. P.C. Fishburn. Induced binary probabilities and the linear ordering polytope: A status report. Mathematical Social Sciences 23, 1992, 67-80.
9. J. Fonlupt and A. Skoda. Strongly polynomial algorithm for the intersection of a line with a polymatroid. Research Trends in Combinatorial Optimization, 69-85, 2009, Springer.
10. A. Gershkov, J.K. Goeree, A. Kushnir, B. Moldovanu, X. Shi. On the Equivalence of Bayesian and Dominant Strategy Implementation. Econometrica, to appear.
11. M. Grötschel, L. Lovász and A. Schrijver. Geometric algorithms and combinatorial optimization. Algorithms and combinatorics, 1988, Springer.
12. J.D. Hartline and A. Karlin. Profit Maximization in Mechanism Design. Chap. 13 in: N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani. Algorithmic Game Theory, Cambridge University Press, 2007.
13. B. Heydenreich, D. Mishra, R. Müller, and M. Uetz. Optimal Mechanisms for Single Machine Scheduling. In: Proc. WINE 2008, LNCS 5385, 2008, 414-425.
14. P. Kenis, (2006). Waiting lists in Dutch health care: An analysis from an organization theoretical perspective. J. Health Organization and Mgmt. 20, 2006, 294-308.
15. A.M. Manelli and D.R. Vincent. Bayesian and Dominant Strategy Implementation in the Independent Private Values Model. Econometrica 78, 2010, 1905-1938.
16. R.B. Myerson. Optimal Auction Design. Math. OR 6, 1981, 58-73.
17. N. Nisan. Introduction to Mechanism Design (for Computer Scientists). Chap. 9 in: N. Nisan, T. Roughgarden, É. Tardos, and V. Vazirani. Algorithmic Game Theory, Cambridge University Press, 2007.
18. M. Queyranne. Structure of a simple scheduling polyhedron. Math. Prog. 58, 1993, 263-285.
19. M. Queyranne and A.S. Schulz Polyhedral Approaches to Machine Scheduling. TU Berlin Technical Report 408/1994.
20. T. Sandholm. Automated Mechanism Desgin: A New Application Area for Search Algorithms. In: Proc. CP 2003, LNCS 2833, 2003, 19-36.
21. W. E. Smith. Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 1956, 59-66.
22. R. Vohra. Optimization and mechanism design. Math. Prog. 134, 2012, 283-303.
23. S. Yasutake, K. Hatano, S. Kijima, E. Takimoto and M. Takeda. Online Linear Optimization over Permutations. In: Proc. ISAAC 2011, LNCS 7074, 534-543.