# Efficient Implementation of Carathéodory's Theorem for the Single Machine Scheduling Polytope

Ruben Hoeksma[a,*], Bodo Manthey[b], Marc Uetz[b]

[a] *Universidad de Chile, Dept. Ingeniería Industrial, República 701, Santiago, Chile*
[b] *University of Twente, Dept. Applied Mathematics, P.O. Box 217, 7500 AE, Enschede, The Netherlands*

**Abstract**

In a fundamental paper in polyhedral combinatorics, Queyranne describes the complete facial structure of a classical object in combinatorial optimization, the single machine scheduling polytope. In the same paper, he answers essentially all relevant algorithmic questions with respect to optimization and separation. In the present paper, motivated by recent applications in the design of optimal incentive compatible mechanisms, we address an algorithmic question that was apparently not addressed before. Namely, we turn Carathéodory's theorem into an algorithm, and ask to write an arbitrary point in the scheduling polytope as a convex combination of the vertices of the polytope. We give a combinatorial $O(n^2)$ time algorithm, which is linear in the naive encoding of the output size. We obtain this result by exploiting the fact that the scheduling polytope is a zonotope, and by the observation that its barycentric subdivision has a simple, linear description. The actual decomposition algorithm is an implementation of a method proposed by Grötschel, Lovász and Schrijver, applied to one of the subpolytopes of the barycentric subdivision. We thereby also shed new light on an algorithm recently proposed for a special case, namely the permutahedron.

## Introduction

Given any point $x$ in a $d$-dimensional polytope $Q$, Carathéodory's theorem implies that $x$ can be written as convex combination of at most $d+1$ vertices of $Q$. We are interested in an algorithmic version of Carathéodory's theorem for a well known polytope in combinatorial optimization, the single machine scheduling polytope. The vertices of this polytope are vectors of completion times corresponding to permutation schedules of $n$ jobs on a single machine. These vectors are obtained by computing the completion times of the jobs when they are processed without idle time in one of the $n!$ possible orders. We denote

*Corresponding author
*Email addresses:* rubenh@dii.uchile.cl (Ruben Hoeksma), b.manthey@utwente.nl (Bodo Manthey), m.uetz@utwente.nl (Marc Uetz)

the convex hull of these vertices by $C$. The algorithmic problem that we want to solve is this: Given some arbitrary $x \in C$, compute the representation of $x$ by at most $n$ vertices $v^i$ of $C$. The fact that this is possible is implied by Carathéodory's theorem since the scheduling polytope for $n$ jobs is $(n-1)$-dimensional [15]. With respect to the input size of the problem, observe that all that is given is a vector of positive processing times $p \in \mathbb{R}^n_+$ and a point $x \in \mathbb{R}^n$. If $x \in C$, the required output is the vertices $v^i$ of $C$ and scalars $\lambda_i \geq 0$ such that $\sum_i \lambda_i = 1$ and $x = \sum_i \lambda_i v^i$. In lack of a better name, we refer to this problem as *decomposition problem* for point $x \in C$.

*Motivation.* As a matter of fact, the single machine scheduling polytope $C$ is very well understood [14]. For instance, it is known to be a polymatroid, and optimization over $C$ (and also the separation problem for $C$) can be done in $\mathrm{O}(n \log n)$ time. Having said this, why would one be interested in solving the decomposition problem? Our motivation is that the problem arises as an algorithmic subproblem in the design of optimal incentive compatible mechanisms in private information settings. In such settings, some of the data, such as job processing times, are private to the jobs. One approach for computing Bayes-Nash optimal mechanisms that recently has received attention is to use linear programming relaxations for the so-called reduced form of the mechanism [1, 7]. Such relaxations yield so-called interim solutions, which are (interior) points of a certain polytope. In the final step of the optimal mechanism, the interim solution has to be translated into a lottery over actual solutions. These actual solutions are the vertices of the polytope. At this point one is confronted with a decomposition problem as described above. Specifically, for the single machine scheduling problem where jobs have private data, the last step to implement a mechanism requires the solution of the decomposition problem of the single machine scheduling polytope. We refer to [7] for a detailed discussion.

*Related Work & Preliminaries.* Already Cunningham [3, Sect. 4.3] observes that an efficient *combinatorial* algorithm to explicitly compute an expression of an interior point of a polytope as convex combination of its vertices is not obvious at all, even if the underlying optimization problem for a given polytope is well understood and can be solved efficiently. With this paper, we exactly follow this line of research and settle the case for the single machine scheduling polytope.

Less directly related to our work is the decomposition of feasible points into vertices in algorithms for submodular function minimization, starting with work by Cunningham [2, 3] and including the strongly polynomial time algorithms of Schrijver [16] and Iwata et al. [8]. More recently, decomposition techniques have also been used to find approximation algorithms for packing and other problems [11, 12, 17], where the fractional solution of an LP-relaxation is first scaled and then decomposed into vertices of the integral polyhedron.

Since the separation problem for the single machine scheduling polytope $C$ can be solved in $\mathrm{O}(n \log n)$ time, the existence of a polynomial time algorithm for the decomposition problem for a given $x \in C$ follows via the ellipsoid method [5]. The decomposition algorithm that does the job is recursive over the dimension

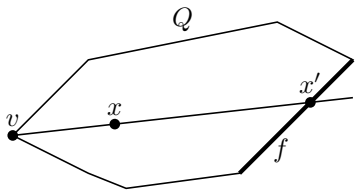


Figure 1: Illustration of the generic decomposition algorithm by Grötschel, Lovász, and Schrijver. From some vertex $v \in Q$, extend a half-line from $v$ in direction $x - v$ until it intersects a lower dimensional face $f$ of $Q$ in a point $x'$. The point $x$ can be written as a convex combination of $v$ and $x'$. Recurse with this face $f$ and the intersection point $x'$ to obtain a convex combination of vertices of $f$ that yields $x'$.

of the polytope and was described by Grötschel, Lovász, and Schrijver in [6]. Figure 1 depicts the idea of that algorithm. For convenience, we refer to this as the *GLS method* in the following.

Let us briefly sketch the state-of-the-art of combinatorial algorithms for the decomposition problem of the single machine scheduling polytope. An $O(n^9)$ algorithm follows directly from work by Fonlupt and Skoda [4] on the intersection of a line with an arbitrary polymatroid and using the GLS method. However, a closer look reveals that an $O(n^3 \log n)$ implementation is possible for the scheduling polytope [7]. Still, this result is unsatisfactory in the following sense. For the permutahedron, Yasutake et al. [18] gave an $O(n^2)$ decomposition algorithm. The permutahedron is precisely the single machine scheduling polytope for the special case where all processing times are equal to one. Hence, the natural question arises if their $O(n^2)$ algorithm can be generalized to the scheduling polytope.

*Contribution & Main Ingredients.* In this paper, we answer this question in the affirmative. Essentially, we show two things. First, we show that there is an $O(n^2)$ decomposition algorithm for the single machine scheduling polytope. Second, we augment the algorithm by Yasutake et al. [18] by a simple and geometric interpretation. In particular, we thereby show that their algorithm is in fact also an implementation of the GLS method.

We now sketch the main ingredients. The backbone of our algorithm is the GLS method. We start by shifting the polytope $C$ by half the processing times of the jobs to obtain $Q$, the polytope of feasible *half time* vectors: $Q = C - p/2$. This shift simply makes the computations a lot easier. However, the main idea is that instead of applying the GLS method directly to the polytope, we apply it to a subpolytope obtained from a polyhedral subdivision of $Q$. Subsequently, we show that the found vertices of this subpolytope can be expressed with at most $n$ vertices of $Q$. The crucial ingredient to get the result is to exploit the fact that the scheduling polytope is a zonotope, that is, all its faces are centrally symmetric. As each of the centers of a given face has a representation by at most two vertices (this is because, by symmetry, for any vertex of such a face, there is another vertex exactly opposite to the center of the face), it suffices to

decompose a given point into (certain) centers. To decompose a given point into centers, we consider the polyhedral subdivision of the scheduling polytope that is induced by these centers. This is also called a barycentric subdivision [9]. For the polytope of half times, we show that this subdivision has a simple, linear description, which we can exploit algorithmically.

It should be mentioned that the idea of using half times, also referred to as midpoints, is not new in scheduling. It has proven to be helpful particularly for the design and analysis of approximation algorithms. Phillips et al. [13] were probably the first to use half times to analyze an approximation algorithm, and Munier et al. [10] were the first to use half times explicitly in the design of approximation algorithms.

We believe that our results are interesting due to the following reasons. First, consider applying the GLS method directly to the scheduling polytope. In order to obtain an $O(n^2)$ implementation, one would have to compute a face $f$ and the intersection point of the half-line through $v$ and $x$ with $f$ in $O(n)$ time in each iteration. We do not see how to do this. Second, considering a naive, unit-cost encoding of the output, the $O(n^2)$ implementation is only linear in the output size, and in that sense "best possible". Third, our structural results shed new light on a well-studied object in polyhedral combinatorics, namely the single machine scheduling polytope. Finally, we believe that the geometric idea behind our approach might prove useful also for other combinatorial problems.

## 1. The Single Machine Scheduling Polytope

Consider a set $N$ of $n$ jobs. Job $j \in N$ has positive processing time $p_j \in \mathbb{R}_+$. Non-preemptive schedules of jobs on a single machine are usually represented by vectors of either starting times $s_j$ or completion times $c_j$. For any non-preemptive schedule without idle time, the starting time of job $j$ is $s_j = \sum_{k<j} p_k$, where $k < j$ denotes that job $k$ is scheduled before job $j$. Then the completion time of job $j$ is $c_j = s_j + p_j$. For all sets $J \subseteq N$ of jobs, let

$$g(J) := \frac{1}{2}\left(\sum_{j\in J} p_j\right)^2 .$$

Queyranne [14] defined the single machine scheduling polytope using completion time vectors $c$ and showed that it is described by the following system of inequalities:

$$\sum_{j\in J} c_j p_j \geq g(J) + \frac{1}{2}\sum_{j\in J} p_j^2 \qquad \text{for all } J \subset N \text{ and} \tag{1}$$

$$\sum_{j\in N} c_j p_j = g(N) + \frac{1}{2}\sum_{j\in N} p_j^2 . \tag{2}$$

Since we assume $p_j > 0$ for all $j \in N$, none of these inequalities is redundant, and the dimension is $n-1$ [14]. Note that, for the degenerate case where $p_k = 0$

for some jobs $k$, we would have to add constraints $0 \leq c_k \leq \sum_{j \in N} p_j$ in order to describe the convex hull of schedules. However, for all algorithmic purposes that we can think of, this degenerate case does not add anything interesting, since we can simply eliminate such jobs and reintroduce them afterwards. In particular, this is true for the problem we address here. Thus, we assume that $p_j > 0$ for all jobs $j \in N$.

In this paper, it is convenient to represent a schedule by $x$, the vector of half times, instead of the vector of completion times. The *half time* of a job is the time at which the job has finished half of its processing. We have

$$x_j = s_j + \frac{1}{2}p_j = c_j - \frac{1}{2}p_j \,.$$

Equivalent to Queyranne's description, the single machine scheduling polytope of half times is completely described by

$$\sum_{j \in J} x_j p_j \geq g(J) \qquad \text{for all } J \subset N \text{ and} \tag{3}$$

$$\sum_{j \in N} x_j p_j = g(N)\,, \tag{4}$$

which is the scheduling polytope of completion times shifted by the vector $-p/2$. Let $Q$ denote the single machine scheduling polytope of half times. The polytope $Q$ is the set of all $x \in \mathbb{R}^n$ that fulfill (3) and (4).

The face lattice of the single machine scheduling polytope is well understood [14]. Every $(n-k)$-dimensional face $f$ of $Q$ corresponds one-to-one with an ordered partition of $N$ into $k$ sets. With an ordered partition, we mean a tuple $(S_1, \ldots, S_k)$ with $S_i \cap S_j = \emptyset$ for all $i \neq j$, $i, j \in \{1, \ldots, k\}$, and $\bigcup_{i=1}^{k} S_i = N$. The intended meaning is that inequalities (3) are tight for all $T_i := S_1 \cup \ldots \cup S_i$, $i \in \{1, \ldots, k\}$. This corresponds to convex combinations of all schedules where jobs in $T_i$ are scheduled before jobs in $N \setminus T_i$, for all $i \in \{1, \ldots, k\}$. The schedules correspond to the ordered partitions $(\{\sigma(1)\}, \ldots, \{\sigma(n)\})$ for all permutations $\sigma$. Each such ordered partition corresponds to a vertex of $Q$ as follows: let $(\{\sigma(1)\}, \ldots, \{\sigma(n)\})$ be an ordered partition and $v$ the vertex it corresponds to, then

$$v_{\sigma(j)} = \frac{1}{2}p_{\sigma(j)} + \sum_{i=1}^{j-1} p_{\sigma(i)} \qquad \text{for all } j \in N. \tag{5}$$

## 2. Zonotopes

In this paper, we make heavy use of the fact that the scheduling polytope is a zonotope.

**Definition 1 (centrally symmetric polytope, zonotope).** Let $P \subseteq \mathbb{R}^n$ be a polytope. $P$ is *centrally symmetric* if it has a center $c \in P$, such that $c + x \in P$ if and only if $c - x \in P$. If all faces of $P$ are centrally symmetric, then $P$ is called a *zonotope*.

An equivalent definition of centrally symmetric is that there is a center $c \in P$ such that for all $x \in P$ also $2c - x \in P$.

Zonotopes also have alternative definitions. They are exactly the images of (higher-dimensional) hypercubes under affine projections, and they are exactly the Minkowski sum of line segments [19]. The standard textbook example for zonotopes is the permutahedron [19], which is the scheduling polytope of completion times when all processing times are equal to one.

The scheduling polytope with arbitrary processing times is a zonotope, too. This can be seen in several ways. For example, the scheduling polytope can be obtained as affine transformation from a hypercube in dimension $\binom{n}{2}$ via linear ordering variables as follows [15, Thm. 4.1]: let the variable $\delta_{ij}$ for $i, j \in N$, $i < j$ be ordering variables. The intended meaning is that $\delta_{ij} = 1$ if and only if job $i$ is processed before job $j$. Then the vertices of this $\binom{n}{2}$-dimensional hypercube correspond one-to-one with all permutations, and the halftime $x_j$ of any job $j$ can be computed by

$$x_j = \frac{1}{2}p_j + \sum_{i<j} \delta_{ij} p_i + \sum_{i>j} (1 - \delta_{ji}) p_i \,.$$

We summarize this brief discussion with the following theorem.

**Theorem 1 (Queyranne & Schulz [15, Thm. 4.1]).** *The scheduling polytope is a zonotope.*

With respect to the centers of the faces of the scheduling polytope of halftimes, we have the following lemma that gives their explicit description.

**Lemma 2.** *Let $f$ be any face of $Q$, defined by the ordered partition $(S_1, \ldots, S_k)$. Then the center of symmetry (barycenter) $c(f)$ of $f$ is given by*

$$c(f)_j = \sum_{\ell=1}^{i-1} \sum_{s \in S_\ell} p_s + \frac{1}{2} \sum_{s \in S_i} p_s \,, \quad j = 1, \ldots, n \,, \tag{6}$$

*where $i$ is the index such that $j \in S_i$.*

Given that a face $f$ of $Q$ corresponds to some ordered partition $(S_1, \ldots, S_k)$, this is not difficult to verify. For the sake of completeness, we give a proof.

Proof. Let $f$ be a face of $Q$, and let $v$ be a vertex of $f$. Let $(S_1, \ldots, S_k)$ be the ordered partition corresponding to $f$. Then $v$ corresponds to an ordering such that jobs in $S_a$ are ordered before jobs in $S_b$ for all $a < b$. Now let $v'$ be the vertex of $f$ that corresponds to the following order: for any two jobs $i, j \in S_a$ and $i \neq j$, we let $j$ be ordered before $i$ if and only if $i$ is ordered before $j$ in $v$. Note that, for any $v$, there is exactly one such $v'$.

We argue that $c(f) = \frac{1}{2}(v + v')$. Suppose that $j \in S_a$. Then for any $b < a$, in both $v$ and $v'$ any job $i \in S_b$ is ordered before job $j$. For any $b > a$, in both $v$ and $v'$ any job $i \in S_b$ is ordered after job $j$. And, for any job $i \in S_a$, $i \neq j$, job

$i$ is ordered before job $j$ in one of $v$ and $v'$ and ordered after job $j$ in the other. From this we have that

$$\frac{1}{2}(v+v')_j = \sum_{\ell=1}^{a-1} \sum_{s\in S_\ell} p_s + \frac{1}{2} \sum_{s\in S_a} p_s \quad \text{for all } j \in S_a,$$

which equals $c(f)_j$ as defined by Lemma 2. Therefore we have that $c(f) = \frac{1}{2}(v+v')$, or $v' = 2c(f) - v$. As $v$ was an arbitrary vertex of $f$, it follows that for any point $x \in f$ there exists $x' \in f$ such that $x' = 2c(f) - x$. Thus $c(f)$ is the center of $f$. □

In particular, observe that the value $c(f)_j$ is the same for all $j \in S_i$ and the center of $Q$ is the point $c$ where all values $c_i$ coincide, i.e., $c_1 = \cdots = c_n$. This is no longer true if we consider the scheduling polytope of start or completion times. The property that all faces of a zonotope are centrally symmetric, as well as the simple description of the centers of these faces by Lemma 2, will be important for the design of the decomposition algorithm in Section 4.

## 3. Barycentric subdivision

Consider the following, polyhedral subdivision of the scheduling polytope $Q$. For any vertex $v$ of $Q$, define polytope $Q_v^c$ as the convex hull of all centers $c(f)$ of faces $f$ that contain $v$:

$$Q_v^c := \operatorname{conv}\{c(f) \mid v \in f\}.$$

Note that $v$ itself is a 0-dimensional face of $Q$ and therefore $v \in Q_v^c$ and, by construction, $v$ is the only vertex of $Q$ that is also a vertex of $Q_v^c$. The polytopes $Q_v^c$ form a subdivision of $Q$, which is known as *barycentric subdivision* [9].

Another polyhedral subdivision of the scheduling polytope $Q$ is obtained by subdividing the polytope according to orders as follows.

**Definition 2.** Let $P \subseteq \mathbb{R}^n$ be a polytope. We define a relation $\sim$ on $P$ as follows: for two points $x, y \in P$, we have $x \sim y$ if there exists a permutation $\sigma : \{1,\dots,n\} \to \{1,\dots,n\}$ such that both $x_{\sigma(1)} \le \ldots \le x_{\sigma(n)}$ and $y_{\sigma(1)} \le \ldots \le y_{\sigma(n)}$.

Based on this definition, define for any vertex $v \in Q$ the polytope

$$Q_v^\sigma := \{x \in Q \mid x \sim v\}.$$

Since vertices of $Q$ correspond to permutation schedules, for every permutation $\sigma$ there is exactly one vertex of $Q$ such that $v_{\sigma(1)} \le \ldots \le v_{\sigma(n)}$. Therefore we have $Q = \bigcup_v Q_v^\sigma$ and $v$ is the only vertex of $Q$ that is also a vertex of $Q_v^\sigma$.

The following two lemmas encode the core and geometric intuition behind the decomposition algorithm that we develop in Section 4. They show that the two above polyhedral subdivisions are in fact equivalent. Thus, we obtain a description of the barycentric subdivision in terms of vertices and facets, all of which can be described explicitly by simple expressions. These insights can be exploited algorithmically.

**Lemma 3.** *Let $Q$ be the single machine scheduling polytope of half times, let $v$ be an arbitrary vertex of $Q$ and let $\sigma$ denote the permutation such that $v_{\sigma(1)} \leq \ldots \leq v_{\sigma(n)}$. Then $Q_v^\sigma$ has the following, linear description:*

$$x_{\sigma(j)} \leq x_{\sigma(j+1)} \qquad \text{for all } j \in \{1, \ldots, n-1\}, \tag{7}$$

$$\sum_{j=1}^{k} x_{\sigma(j)} p_{\sigma(j)} \geq \frac{1}{2} \left( \sum_{j=1}^{k} p_{\sigma(j)} \right)^2 \qquad \text{for all } k \in \{1, \ldots, n-1\}, \text{ and} \tag{8}$$

$$\sum_{j \in N} x_j p_j = \frac{1}{2} \left( \sum_{j \in N} p_j \right)^2. \tag{9}$$

Proof. Since $Q_v^\sigma \subseteq Q$, (8) and (9) are satisfied for every point in $Q_v^\sigma$. Since $\sigma$ is the only permutation with $v_{\sigma(1)} \leq \ldots \leq v_{\sigma(n)}$, we have that $x$ satisfies (7) if $x \sim v$. Therefore, (7) holds for any point in $Q_v^\sigma$.

It remains to be shown that (7), (8), and (9) imply $x \in Q_v^\sigma$. Let $x$ satisfy (7), (8) and (9). For simplicity of notation and without loss of generality, let all vectors be sorted such that $x_i \leq x_j$ if and only if $i \leq j$. Then, for each $j$, we have

$$\left( \sum_{i=1}^{j} p_i \right) x_j \geq \sum_{i=1}^{j} p_i x_i \geq \frac{1}{2} \left( \sum_{i=1}^{j} p_i \right)^2.$$

Thus, $x_j \geq \frac{1}{2} \sum_{i=1}^{j} p_i$ for all $j$. Now suppose $x$ satisfies (7), (8), and (9), but $x \notin Q$. Then there is a set $J$ of minimal cardinality, such that (3) is not satisfied. This means that

$$\sum_{i \in J} p_i x_i < \frac{1}{2} \left( \sum_{i \in J} p_i \right)^2.$$

But then, for $j = \max_{k \in J} k$, we have

$$\begin{aligned}
\sum_{i \in J \setminus \{j\}} p_i x_i = \sum_{i \in J} p_i x_i - p_j x_j &< \frac{1}{2} \left( \sum_{i \in J} p_i \right)^2 - p_j x_j \\
&\leq \frac{1}{2} \left( \sum_{i \in J} p_i \right)^2 - p_j \frac{1}{2} \left( \sum_{i=1}^{j} p_i \right) \\
&\leq \frac{1}{2} \left( \sum_{i \in J} p_i \right)^2 - p_j \frac{1}{2} \left( \sum_{i \in J} p_i \right) = \frac{1}{2} \left( \sum_{i \in J \setminus \{j\}} p_i \right)^2.
\end{aligned}$$

This contradicts that $J$ is a set of minimal cardinality that does not satisfy (3). So (7), (8), and (9) imply $x \in Q$.

Now suppose $x \in Q \setminus Q_v^\sigma$, then $x \in Q_{v'}^\sigma$ for some other vertex $v' \in Q$, which would imply that (7) is not valid for $x$. Hence, $x \in Q_v^\sigma$. □

**Lemma 4.** *Let $Q$ be the single machine scheduling polytope of half times. Then, for all vertices $v$ of $Q$, we have*

$$Q_v^c = Q_v^\sigma \,.$$

PROOF. Lemma 2 implies that the vertices of $Q_v^c$ are given by (6) for all $f \ni v$. Moreover, it also follows from Lemma 2 that for any face $f$ and any vertex $v$ of $f$ we have that $v \sim c(f)$. Thus, for any vertex $q$ of $Q_v^c$ we have that $q \sim v$. It follows that $Q_v^c \subseteq Q_v^\sigma$.

Let $q$ be a vertex of $Q_v^\sigma$. Then, by Lemma 3, at least $n-1$ inequalities among (7) and (8) are tight for $q$. Let $\ell \in \{1, \ldots, n-1\}$. If (8) is tight for $q$ for $k = \ell$, then (7) cannot be tight for $q$ for $j = \ell$. This is because if (8) is tight for $q$ and $k = \ell$, then jobs $1, \ldots, \ell$ are scheduled before jobs $\ell+1, \ldots, n$. Therefore,

$$q_{\ell+1} \geq \frac{1}{2} p_{\ell+1} + \sum_{j=1}^{\ell} p_j$$

and

$$q_\ell \leq \frac{1}{2} p_\ell + \sum_{j=1}^{\ell-1} p_j \,.$$

Thus, $q_\ell < q_{\ell+1}$ since all processing times are assumed to be positive. This implies that for any $\ell \in \{1, \ldots, n-1\}$, we have that $q$ satisfies exactly one of the following: (8) is tight for $k = \ell$ or (7) is tight for $j = \ell$. The inequalities (8) that are tight for $q$ induce an ordered partition $(S_1, \ldots, S_k)$ that corresponds to a face $f$. Moreover, since all inequalities (8) are tight for $v$, we have that $f \ni v$. The inequalities (7) that are tight for $q$ ensure that $q_j = q_{j+1}$ for all $j \in S_i$ and any $i \in \{1, \ldots, k\}$.

It follows from Lemma 2 that $q = c(f)$ and, thus, $q$ is a vertex of $Q_v^c$. Since this holds for any vertex of $Q_v^\sigma$, we have $Q_v^\sigma \subseteq Q_v^c$. Thus, $Q_v^\sigma = Q_v^c$. □

For simplicity of notation, we define $Q_v := Q_v^c \ (= Q_v^\sigma)$.

Figure 2 illustrates the barycentric subdivision of the scheduling polytope. It shows the scheduling polytope for three jobs together with its barycentric subdivision (indicated by dashed lines). The subpolytope containing vertex $v_{213}$ contains all vectors $x \in Q$ for which $x_2 \leq x_1 \leq x_3$. Its vertices are $v_{213}$, and all centers of faces on which $v_{213}$ lies. Its facets are defined by $x_1p_1 + x_2p_2 + x_3p_3 = (p_1+p_2+p_3)^2$ together with one of the following equalities:

$$\begin{aligned} x_1p_1 + x_2p_2 &= (p_1+p_2)^2 \,, \\ x_2p_2 &= (p_2)^2 \,, \\ x_2 &= x_1 \,, \\ x_3 &= x_1 \,. \end{aligned}$$

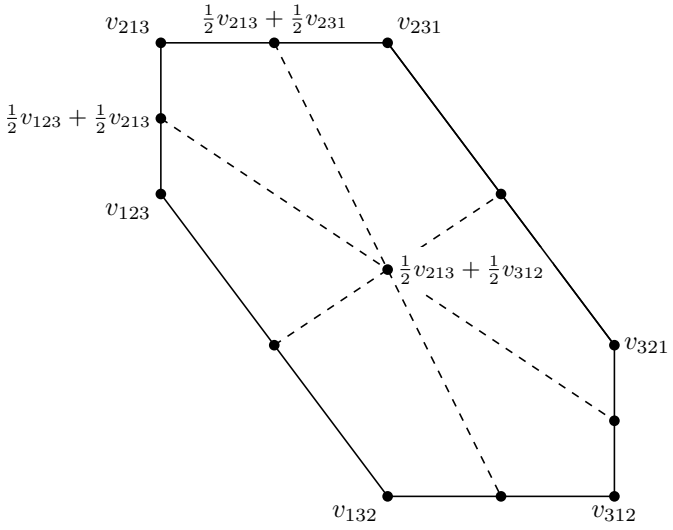


Figure 2: Barycentric subdivision of a scheduling polytope with three jobs. $v_{ijk}$ denotes the vertex corresponding to the order $i, j, k$. Only vertices of $Q$, and the vertices of the subpolytope $Q_v$ corresponding to vertex $v_{213}$ are labeled. The latter are $v_{213}$ and convex combinations of $v_{213}$ with one "opposite" vertex.

## 4. Decomposition Algorithm for the Single Machine Scheduling Polytope

Based on Lemma 3, we next develop a decomposition algorithm for the scheduling polytope that runs in time $O(n^2)$. This algorithm can be seen as a generalization of an algorithm recently proposed by Yasutake et al. [18] for the permutahedron. We argue here that this algorithm is in fact an application of the GLS method [6, Thm. 6.5.11]. Before diving into technical details, we describe the high level idea.

We know that any point $x \in Q$ lies in a subpolytope $Q_v$ of the barycentric subdivision of $Q$, namely for a vertex $v$ for which $v \sim x$ according to Definition 2.[1] Moreover, $Q_v$ is described by inequalities (7) and (8), and the vertices of $Q_v$ consist of the points $\frac{v+v'}{2}$ for some vertices $v'$ of $Q$. This means that a decomposition of $x$ into vertices of $Q_v$ also yields a decomposition into vertices of $Q$.

The idea of our algorithm is as follows: We find a decomposition of $x$ into vertices of $Q_v$ by using the GLS method [6, Thm. 6.5.11]. The idea of this algorithm is illustrated in Figure 3: Given $x = x^1 \in Q_v$ (we have $v = v^1$), we extend the difference vector $x^1 - v^1$ towards the intersection with a lower dimensional face of $Q_v$ (this will be a facet of $Q_v$, unless we accidentally hit a

[1] In case of ties, $x$ lies on the intersection of several of such subpolytopes, namely those corresponding to vertices $v$ with $v \sim x$. We can break such ties arbitrarily.

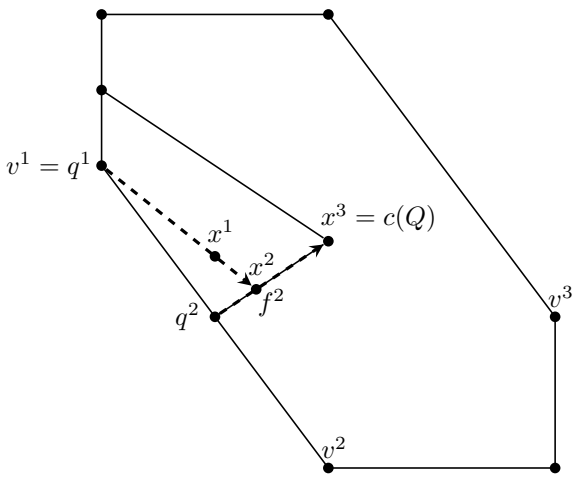


Figure 3: Visualization of the decomposition algorithm on a single machine scheduling polytope for three jobs

face of even lower dimension). Then recurse with this intersection point and the face on which it lies. To arrive at the claimed computation time, it is crucial that both the intersection point and the face(t) on which it lies can be computed in time $O(n)$. This is indeed possible because of the explicit, linear description of $Q_v$ given in Lemma 3. As the number of iterations is bounded by the dimension of $Q_v$, which is equal to the dimension of $Q$, this gives an $O(n^2)$ implementation. Finally, by the fact that all vertices of $Q_v$ can be written as $\frac{v+v'}{2}$ for vertices $v'$ of $Q$, we obtain a decomposition of $x$ into at most $n$ vertices of $Q$.

In order to describe the technical details of the algorithm, we use the following notation and facts, all of which follow from the structural insights of Section 3.

$v$: vertex of $Q$ corresponding to the permutation $1, 2, \ldots, n$; we have $v = v^1$;

$J^t$: set of indices;

$f^t$: face of $Q_v$ associated with $J^t$ such that $y_j = y_{j+1}$ for all $y \in f^t$ and all $j \in \{1, \ldots, n-1\} \setminus J^t$;

$q^t$: vertex of $f^t$;

$v^t$: vertex of $Q$ such that $q^t = \frac{1}{2}(v + v^t)$;

$x^t$: point in $f^t$;

$\tilde{\kappa}_t$: scalar such that $x^t = \tilde{\kappa}_t q^t + (1 - \tilde{\kappa}_t) x^{t+1}$;

$\kappa_t$: scalar corresponding to $q^t$ in the convex combination $x = \sum_t \kappa_t q^t$;

$\lambda_t$: scalar corresponding to $v^t$ in the convex combination $x = \sum_t \lambda_t v^t$.

Moreover, for ease of notation and without loss of generality, we assume that

**Algorithm 1:** Decomposition Algorithm

**input :** processing times $p$, point $x \in Q$ with $x_1 \leq \ldots \leq x_n$
**output:** at most $n$ vertices $v^t$ of $Q$ and coefficients $\kappa_t \in [0,1]$

$t := 1, x^1 := x, J^1 := \{i \in \{1,\ldots,n-1\} \mid x_i^1 < x_{i+1}^1\}$;
let $v$ be the vertex with $v_1 \leq \ldots \leq v_n$;
**while** $J^t \neq \emptyset$ **do**
  $q^t := \texttt{VERTEX}(J^t)$;
  $v^t := 2q^t - v$;
  $\tilde{\kappa}_t := \min_{j \in J^t} (x_{j+1}^t - x_j^t)/(q_{j+1}^t - q_j^t)$;
  $x^{t+1} := \frac{1}{1-\tilde{\kappa}_t}(x^t - \tilde{\kappa}_t q^t)$;
  $J^{t+1} := \{i \in J^t \mid x_i^{t+1} < x_{i+1}^{t+1}\}$;
  $\kappa_t := (1 - \sum_{\tau=1}^{t-1} \kappa_\tau)\tilde{\kappa}_t$;
  $t := t + 1$;
$q^t := x^t$;
$v^t := 2q^t - v$;
$\kappa_t := 1 - \sum_{\tau=1}^{t-1} \kappa_\tau$;
$\lambda_1 := \frac{1}{2} + \frac{1}{2}\kappa_1$;
**for** $\tau \in \{2,\ldots,t\}$ **do**
  $\lambda_\tau := \frac{1}{2}\kappa_\tau$;

the given point $x \in Q$ satisfies $x_1 \leq \ldots \leq x_n$.[2]

The subroutine $\texttt{VERTEX}(J^t)$ computes the vertex of $Q$ corresponding to the face associated with $J^t$ as follows: Let $J^t(i)$ denote the $i$-th element in $J^t$ and define $J^t(0) = 1$. Then, for $j \in \{J^t(i), \ldots, J^t(i+1) - 1\}$, we compute

$$q_j^t = \sum_{k=1}^{J^t(i)-1} p_k + \frac{1}{2} \sum_{k=J^t(i)}^{J^t(i+1)-1} p_k \,.$$

Note that vertex $q^t$ can be computed in linear time per iteration by just computing $P_i^t := \sum_{k=J^t(i)}^{J^t(i+1)-1} p_k$ for all $i$, in time $\mathrm{O}(n)$. Then, $q_1^t = \frac{1}{2}P_1^t$, and for $j \in \{J^t(i), \ldots, J^t(i+1) - 1\}$ and $k \in \{J^t(i+1), \ldots, J^t(i+2) - 1\}$, the values for $q^t$ are computed iteratively as $q_k^t = q_j^t + \frac{1}{2}(P_i^t + P_{i+1}^t)$.

**Theorem 5.** *For any $x \in Q$, Algorithm 1 outputs a convex combination of vertices of $Q$ for $x$ in $\mathrm{O}(n^2)$ time.*

[2]This comes at the expense of sorting, which costs $\mathrm{O}(n \log n)$ time and, thus, falls within the $\mathrm{O}(n^2)$ time complexity of the proposed algorithm.

PROOF. We first show by induction that $x^t \in f^t$ during any iteration $t$ of the algorithm. Note that $Q_v$ is chosen such that $x^1 = x \in Q_v = f^1$, which is our base step. Now, as the inductive step, we show that $x^t \in f^t$ implies $x^{t+1} \in f^{t+1}$. From line 7 of the algorithm we have that inequalities (7) are tight for $x^{t+1}$ and all $j \notin J^{t+1}$. Thus, it is sufficient to show that $x^t \in Q_v$ implies that $x^{t+1} \in Q_v$. By construction, $q^t$ is the vertex of $f^t$ for which (7) is tight for all $j \notin J^t$ and (8) is tight for all $k \in J^t$. Now suppose $x^t \in Q_v$. Then, of course, $x^t$ and $q^t$ satisfy (8), and we have

$$\sum_{j=1}^{k} x_j^{t+1} p_j = \sum_{j=1}^{k} p_j \frac{x_j^t - \tilde{\kappa}_t q_j^t}{1 - \tilde{\kappa}_t} = \frac{1}{1 - \tilde{\kappa}_t} \left( \sum_{j=1}^{k} p_j x_j^t - \tilde{\kappa}_t \sum_{j=1}^{k} p_j q_j^t \right) \geq \frac{1}{2} \left( \sum_{j=1}^{k} p_j \right)^2$$

for all $k \in \{1, \ldots, n\}$. From the definition of $x^{t+1}$ we have

$$\begin{aligned} x_{j+1}^{t+1} - x_j^{t+1} &= \frac{x_{j+1}^t - \tilde{\kappa}_t q_{j+1}^t}{1 - \tilde{\kappa}_t} - \frac{x_j^t - \tilde{\kappa}_t q_j^t}{1 - \tilde{\kappa}_t} \\ &= \frac{1}{1 - \tilde{\kappa}_t} \left( x_{j+1}^t - x_j^t - \tilde{\kappa}_t \left( q_{j+1}^t - q_j^t \right) \right) . \end{aligned}$$

By definition we have for all $j \notin J^t$ that $x_j^t = x_{j+1}^t$ and $q_j^t = q_{j+1}^t$. Moreover, we have by line 5 of the algorithm

$$\tilde{\kappa}_t = \min_{j \in J^t} \frac{x_{j+1}^t - x_j^t}{q_{j+1}^t - q_j^t} . \tag{10}$$

Therefore $\tilde{\kappa}_t \leq \frac{x_{j+1}^t - x_j^t}{q_{j+1}^t - q_j^t}$ for all $j \in J^t$. Thus, we obtain for all $j \in N$

$$\begin{aligned} x_{j+1}^{t+1} - x_j^{t+1} &\geq \frac{1}{1 - \tilde{\kappa}_t} \left( x_{j+1}^t - x_j^t - \frac{x_{j+1}^t - x_j^t}{q_{j+1}^t - q_j^t} \left( q_{j+1}^t - q_j^t \right) \right) \\ &= 0 . \end{aligned}$$

Hence, $x^{t+1}$ satisfies (7)–(9). From Lemma 3, we have $x^{t+1} \in Q_v$ and, therefore, $x^{t+1} \in f^{t+1}$. We conclude, by mathematical induction, that $x^t \in f^t$ during any iteration $t$ of the algorithm.

In addition, (10) ensures that for at least one $j \in J^t$, we have $x_{j+1}^{t+1} = x_j^{t+1}$ and thus $|J^{t+1}| < |J^t|$. Since $|J^1| \leq n - 1$, the algorithm terminates after at most $n-1$ iterations. Let $t^*$ be the value of $t$ as the algorithm terminates. Note that $J^{t^*} = \emptyset$ and thus $x^{t^*} = c(Q)$, the center of $Q$. Furthermore, from line 12 of the algorithm, we have $\kappa_{t^*} = 1 - \sum_{j=1}^{t^*-1} \kappa_j$, which implies $\sum_{j=1}^{t^*} \kappa_j = 1$. For $t \in \{1, \ldots, t^* - 1\}$, we have

$$x^t = \tilde{\kappa}_t q^t + (1 - \tilde{\kappa}_t) x^{t+1} .$$

Iteratively applying this equality yields

$$x = \prod_{\tau=1}^{t^*-1} (1 - \tilde{\kappa}_\tau) x^{t^*} + \sum_{t=1}^{t^*-1} \prod_{\tau=1}^{t-1} (1 - \tilde{\kappa}_\tau) \tilde{\kappa}_t q^t . \tag{11}$$

We also have that

$$\begin{aligned}
1 - \sum_{\tau=1}^{t} \kappa_\tau &= 1 - \sum_{\tau=1}^{t-1} \kappa_\tau - \kappa_t \\
&= 1 - \sum_{\tau=1}^{t-1} \kappa_\tau - \tilde{\kappa}_t \left( 1 - \sum_{\tau=1}^{t-1} \kappa_\tau \right) \\
&= (1 - \tilde{\kappa}_t) \left( 1 - \sum_{\tau=1}^{t-1} \kappa_\tau \right) ,
\end{aligned}$$

where the second equality follows from line 8 of the algorithm. Applying this equality iteratively yields

$$1 - \sum_{\tau=1}^{t} \kappa_\tau = \prod_{\tau=1}^{t} (1 - \tilde{\kappa}_\tau) \ .$$

This also gives us the following identity for $\kappa_t$, for $t \neq t^*$:

$$\kappa_t = \tilde{\kappa}_t \left( 1 - \sum_{\tau=1}^{t-1} \kappa_\tau \right) = \tilde{\kappa}_t \prod_{\tau=1}^{t-1} (1 - \tilde{\kappa}_\tau) \ .$$

So, expanding from (11) and using the above two identities and the definitions of $q^{t^*}$ and $\kappa^{t^*}$ in lines 10 and 12 of the algorithm, we have

$$\begin{aligned}
x &= \left( 1 - \sum_{\tau=1}^{t^*-1} \kappa_\tau \right) x^{t^*} + \sum_{t=1}^{t^*-1} \kappa_t q^t \\
&= \kappa_{t^*} x^{t^*} + \sum_{t=1}^{t^*-1} \kappa_t q^t \\
&= \kappa_{t^*} q^{t^*} + \sum_{t=1}^{t^*-1} \kappa_t q^t = \sum_{t=1}^{t^*} \kappa_t q^t \, .
\end{aligned}$$

Now since $v^t = 2q^t - v$, we have $q^t = \frac{1}{2}(v + v^t)$. From lines 13 and 14 of the

algorithm we have $\lambda_1 = \frac{1}{2} + \frac{1}{2}\kappa_1$ and $\lambda_t = \frac{1}{2}\kappa_t$, for $t = 2, \ldots, t^*$. This yields:

$$\begin{aligned} x &= \sum_{t=1}^{t^*} \kappa_t q^t \\ &= \sum_{t=1}^{t^*} \kappa_t \frac{1}{2}(v + v^t) \\ &= \sum_{t=1}^{t^*} \kappa_t \frac{1}{2} v + \sum_{t=1}^{t^*} \kappa_t \frac{1}{2} v^t \\ &= \frac{1}{2} v + \frac{1}{2}\kappa_1 v + \sum_{t=2}^{t^*} \frac{1}{2}\kappa_t v^t \\ &= \lambda_1 v + \sum_{t=2}^{t^*} \lambda_t v^t = \sum_{t=1}^{t^*} \lambda_t v^t \,, \end{aligned}$$

where the second equality follows from line 4 of the algorithm. It follows from line 12 of the algorithm that $\sum_{\tau=1}^{t^*} \lambda_\tau = \sum_{\tau=1}^{t^*} \kappa_\tau = 1$. From (10), we obtain that $0 \leq \tilde{\kappa}_t \leq 1$ for all $t \in \{1, \ldots, t^*\}$. Therefore, it follows from line 8 of the algorithm that $\kappa_t \geq 0$ and $\sum_{\tau=1}^{t} \kappa_\tau \leq 1$ for all $t \in \{1, \ldots, t^*\}$. Thus, $x = \sum_{t=1}^{t^*} \lambda_t v^t$ is indeed an expression for $x$ as a convex combination of no more than $n$ vertices of $Q$.

Since none of the steps within each of at most $n - 1$ iterations takes more than $\mathrm{O}(n)$ time, the total computation time of the algorithm is $\mathrm{O}(n^2)$. □

## 5. Conclusions

The obvious open question is if our algorithm can be generalized to arbitrary zonotopes. As a blueprint of an algorithm that is of course true. However, in order to shape this into a combinatorial algorithm, we would have to find explicit expressions for the centers of symmetry, as well as the faces of the resulting barycentric subdivision that is induced by these centers. In the case of the single machine scheduling polytope we were able to do this, because the faces of the subdivisions are exactly defined by the linear inequalities induced by orderings $x_{\sigma(1)} \leq \cdots \leq x_{\sigma(n)}$. We do not see how that could be done in general for zonotopes. However for other, concrete combinatorial problems it might very well be doable.

## Acknowledgements

We thank Maurice Queyranne for a helpful discussion and for pointing us to the paper by Yasutake et al. [18]. We also thank a careful reviewer for many helpful and constructive comments. The first author was partially supported by the Millennium Nucleus Information and Coordination in Networks ICM-FIC RC130003.

## References


[1] Y. Cai, C. Daskalakis, and S. M. Weinberg. Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization. In *Proc. 53rd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 130–139. IEEE, 2012.

[2] W. H. Cunningham. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, 36:161–188, 1984.

[3] W. H. Cunningham. On submodular function minimization. *Combinatorica*, 5:186–192, 1985.

[4] J. Fonlupt and A. Skoda. Strongly polynomial algorithm for the intersection of a line with a polymatroid. In W. Cook, L. Lovász, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 69–85. Springer, 2009.

[5] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[6] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[7] R. Hoeksma and M. Uetz. Two dimensional optimal mechanism design for a sequencing problem. In M. Goemans and J. Correa, editors, *Integer Programming and Combinatorial Optimization*, volume 7801 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2013.

[8] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial time algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

[9] C. W. Lee. Subdivisions and triangulations of polytopes. In *Handbook of Discrete and Computational Geometry*, chapter 17. Chapman & Hall/CRC, 2nd edition, 2004.

[10] A. Munier, M. Queyranne, and A. S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, volume 1412 of *Lecture Notes in Computer Science*, pages 367–382. Springer, 1998.

[11] O. Parekh. Iterative packing for demand and hypergraph matching. In O. Günlük and G. J. Woeginger, editors, *Integer Programming and Combinatoral Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 349–361. Springer, 2011.

[12] O. Parekh and D. Pritchard. Generalized hypergraph matching via iterated packing and local ratio. In E. Bampis and O. Svensson, editors, *Approximation and Online Algorithms*, volume 8952 of *Lecture Notes in Computer Science*. Springer, 2015.

[13] C. Phillips, C. Stein, and J. Wein. Scheduling jobs that arrive over time. In S. G. Akl, F. Dehne, J.-R. Sack, and N. Santoro, editors, *Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 1995.

[14] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58(1):263–285, 1993.

[15] M. Queyranne and A. S. Schulz. Polyhedral approaches to machine scheduling. Preprint 408-1994, TU Berlin, 1994.

[16] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80:346–355, 2000.

[17] G. Stamoulis. Approximation algorithms for bounded color matchings via convex decompositions. In E. Csuhaj-Varjú, M. Dietzfelbinger, and Z. Ésik, editors, *Mathematical Foundations of Computer Science 2014*, volume 8635 of *Lecture Notes in Computer Science*, pages 625–636. Springer, 2014.

[18] S. Yasutake, K. Hatano, S. Kijima, E. Takimoto, and M. Takeda. Online linear optimization over permutations. In *Algorithms and Computation*, volume 7074 of *Lecture Notes in Computer Science*, pages 534–543. Springer, 2011.

[19] G. M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1995.