Decomposition Algorithm for the Single Machine Scheduling Polytope

Ruben Hoeksma, Bodo Manthey, and Marc Uetz

University of Twente, Dept. Applied Mathematics, Enschede, The Netherlands
{r.p.hoeksma,b.manthey,m.uetz}@utwente.nl

Abstract. Given an n-vector p of processing times of jobs, the single machine scheduling polytope C arises as the convex hull of completion times of jobs when these are scheduled without idle time on a single machine. Given a point $x \in C$, Carathéodory's theorem implies that x can be written as convex combination of at most n vertices of C. We show that this convex combination can be computed from x and p in time $\mathrm{O}(n^2)$, which is linear in the naive encoding of the output. We obtain this result using essentially two ingredients. First, we build on the fact that the scheduling polytope is a zonotope. Therefore, all of its faces are centrally symmetric. Second, instead of C, we consider the polytope Q of half times and its barycentric subdivision. We show that the subpolytopes of this barycentric subdivison of Q have a simple, linear description. The final decomposition algorithm is in fact an implementation of an algorithm proposed by Grötschel, Lovász, and Schrijver applied to one of these subpolytopes.

1 Introduction & Contribution

Given any point x in a d-dimensional polytope P, Carathéodory's theorem implies that x can be written as convex combination of at most $d+1$ vertices of P. We are interested in an algorithmic version of Carathéodory's theorem for a specific polytope, namely the polytope C that arises as the convex hull of completion times of n jobs when these are sequenced non-preemptively without idle time on a single machine. More specifically, we are given a vector of positive processing times $p \in \mathbb{R}^n_+$ and some $x \in C$, and our goal is to compute an explicit representation of x by at most n vertices v^i of C, such that $x = \sum_i \lambda_i v^i$ for $\lambda_i \geq 0$ for all i and $\sum_i \lambda_i = 1$. We refer to this problem as *decomposition problem*.

The polytope C, also known as the single machine scheduling polytope, is well understood [12]. In particular, it is known to be a polymatroid, and the separation problem for C can be solved in $\mathrm{O}(n \log n)$ time. Therefore, the existence of a polynomial time decomposition algorithm follows from the ellipsoid method [5]. A generic approach to compute a decomposition has been described by Grötschel, Lovász, and Schrijver [6]. We call this the *GLS method* in the following. Figure 1 depicts the idea behind the GLS method.

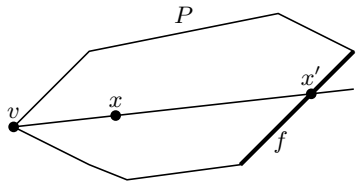

Fig. 1. Illustration of the decomposition algorithm by Grötschel, Lovász, and Schrijver. From some vertex $v \in P$, extend a half-line from v in direction $x - v$ until it intersects a lower dimensional face f of P in a point x'. The point x can be written as a convex combination of v and x'. Recurse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

Yet, already Cunningham [3] remarked that it is interesting to find efficient combinatorial decomposition algorithms for specific polymatroids and that it is in general not straightforward to do so, even if the underlying optimization problem is well understood and can be solved efficiently. Decomposition of feasible points into vertices also plays an important role in algorithms for submodular function minimization, starting with work by Cunningham [2, 3] and including the strongly polynomial time algorithms of Schrijver [14] and Iwata et al. [8].

Apart from this general motivation, the decomposition problem arises in the design of efficient mechanisms for optimization problems in private information settings [1, 7]. In such problems, feasible points correspond to so-called interim (expected) allocations, which are computed as solutions to linear programming formulations. The decomposition is needed to translate these interim allocations into actual implementations of the mechanism.

Let us sketch the state-of-the-art of the question to decompose a given point in the scheduling polytope C into vertices of C. An $O(n^9)$ algorithm follows directly from work by Fonlupt and Skoda [4] on the intersection of a line with a (general) polymatroid using the GLS method. However, a closer look reveals that an $O(n^3 \log n)$ implementation is also possible [7]. Still, this result is unsatisfactory in the following sense. For the permutahedron, Yasutake et al. suggested an $O(n^2)$ decomposition algorithm [15]. The permutahedron is precisely the single machine scheduling polytope for the special case where all processing times are 1. Hence, the natural question is if their $O(n^2)$ algorithm can be generalized to the scheduling polytope.

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. The core of our algorithm remains the GLS method. However, we apply the algorithm to a specific subpolytope of a polyhedral subdivision of the polytope Q of *half times*, i.e., Q is obtained by shifting C by half the processing times of the jobs: $Q = C - p/2$. Second, we augment the algorithm by Yasutake et al. [15] by a simple, geometric interpretation. In particular, this shows that also their algorithm is in fact an implementation of the GLS method.

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. [11] 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.

The crucial ingredient to get our results is to exploit that the scheduling polytope is a zonotope. This means that all its faces are centrally symmetric. As each of the centers of a given face has a representation by two vertices, 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 can show that this subdivision has a simple, linear description, which we can exploit algorithmically.

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 $\mathrm{O}(n^2)$ implementation, one would have to compute a face f and the intersection point of the halfline through v and x with f in $\mathrm{O}(n)$ time in each iteration. We do not see how to do this. Second, considering a naive, unit-cost encoding of the output, the $\mathrm{O}(n^2)$ implementation is only linear in the output size. Third, our structural results shed new light on a well-studied object in polyhedral combinatorics, namely the single machine scheduling polytope.

2 The Single Machine Scheduling Polytope

Consider a set N of n jobs. Job $j \in N$ has processing time $p_j \in \mathbb{R}_+$. Nonpreemptive 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 nonpreemptive 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 $K \subseteq N$ of jobs, let

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

Queyranne [12] defined the single machine scheduling polytope using completion time vectors c and showed that it is described by the following system of inequalities:

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

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

If $p_j > 0$ for all $j \in N$, none of these inequalities is redundant, and the dimension is $n-1$ [12]. 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 $j \in N$ from now on.

In this paper, it is convenient to represent a schedule by h, the vector of half times, instead of by a 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

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

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

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

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

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

The face lattice of the single machine scheduling polytope is well understood [12]. 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 σ. 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

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

3 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 $2c - x \in P$.

Also zonotopes 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 [16]. The standard textbook [16] example for zonotopes is the permutahedron, which is the scheduling polytope in completion times when all processing times are 1.

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 [13, Thm. 4.1]: let the variable δ_{ij} for $i, j \in N$, $i < j$ be ordering variables. The intended meaning 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 h_j of any job j can be computed by

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

We summarize this brief discussion with the following Theorem.

Theorem 2 (Queyranne & Schulz [13, Thm. 4.1]). *The scheduling polytope is a zonotope.*

With respect to the centers of the faces of the scheduling polytope in halftimes, we have the following lemma.

Lemma 3. *Consider an arbitrary face f of Q, defined by the ordered partition $(S_1, \ldots, S_k)$, then the barycenter (or center of mass) $c(f)$ of f is given by*

$$c(f)_j = \sum_{\ell=1}^{i-1} \sum_{h \in S_\ell} p_h + \frac{1}{2} \sum_{h \in S_i} p_h \quad \text{for all} \quad j \in S_i \ . \tag{6}$$

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 here.

Proof. Let f be any 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 v' in f that satisfies this order.

We show that indeed $c(f) = \frac{1}{2}(v + v')$. 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

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

which is indeed $c(f)_j$ as defined by Lemma 3. Therefore we have that $c(f) = \frac{1}{2}(v+v')$ and thus $v' = 2c(f) - v$. Since this holds for any vertex $v \in f$, it follows that for any point $x \in f$ there exists a point $x' \in f$ such that $x' = 2c(f) - x$ and thus $c(f)$ is the center of f. □

In particular, observe that all $j \in S_i$ have the same value, and the center of Q is the point c where all values c_i coincide, i.e., $c_1 = \ldots = c_n$. Note that this is no longer true if we consider the scheduling polytope in start or completion times. The property that all faces of a zonotope are centrally symmetric, as well as the simple description of these centers by Lemma 3, will be important for the design of the decomposition algorithm in Section 5.

4 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 barycenters $c(f)$ of faces f that contain v:

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

Then we have $Q = \bigcup_v Q_v^c$. By construction, v is the only vertex of Q that is also a vertex of Q_v^c. The subdivision thus obtained is also 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 4. *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 $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that both $x_{\sigma(1)} \leq \ldots \leq x_{\sigma(n)}$ and $y_{\sigma(1)} \leq \ldots \leq y_{\sigma(n)}$.*

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

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

Because every permutation σ is represented by a vertex of Q, we have $Q = \bigcup_v Q_v^\sigma$, and v is the only vertex of Q that is also a vertex of Q_v^σ.

The following two lemmas encode the core and geometric intuition behind the decomposition algorithm that we develop in Section 5. 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 5. *Let Q be the single machine scheduling polytope in half times, let v be an arbitrary vertex of Q and let σ denote a permutation such that $v_{\sigma(1)} \leq \ldots \leq v_{\sigma(n)}$. Then Q_v^σ has the following, linear description:*

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

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

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

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

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

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

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

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

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

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

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

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

Lemma 6. *Let Q be the single machine scheduling polytope in half times. Then, for all vertices v of Q, we have*

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

Proof. Lemma 3 implies that the vertices of Q_v^c are given by (6) for all $f \ni v$. From (6), we have $q \sim v$ for any vertex q of Q_v^c. It follows that $Q_v^c \subseteq Q_v^\sigma$.

Now, by Lemma 5, any vertex of Q_v^σ is obtained by having $n-1$ tight constraints among (7) and (8). Consider any such vertex q of Q_v^σ.

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) 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 $f \ni v$. 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 that $q = c(f)$ and, thus, q is a vertex of Q_v^c. Since this holds for any vertex of Q_v^σ, 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 $h \in Q$ for which $h_2 \leq h_1 \leq h_3$. Its vertices are v_{213}, and all centers of faces on which v_{213} lies. Its facets are defined by $h_1 p_1 + h_2 p_2 + h_3 p_3 = (p_1 + p_2 + p_3)^2$ together with one of the following equalities:

$$\begin{aligned} h_1 p_1 + h_2 p_2 &= (p_1 + p_2)^2 \ , \\ h_2 p_2 &= (p_2)^2 \ , \\ h_2 &= h_1 \ , \\ h_3 &= h_1 \ . \end{aligned}$$

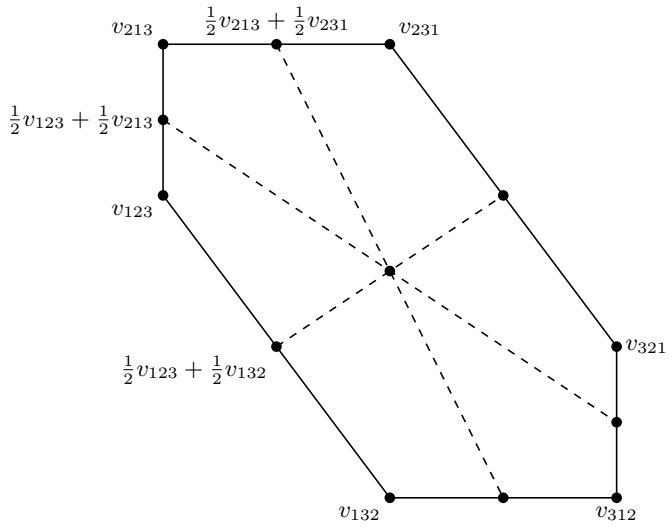

Fig. 2. Barycentric subdivision of a scheduling polytope with three jobs. v_{ijk} denotes the vertex corresponding to the order i, j, k

5 Decomposition Algorithm for the Single Machine Scheduling Polytope

Based on Lemma 5, 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. [15] 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 $h \in Q$ lies in a subpolytope Q_v of the barycentric subdivision of Q, namely for a vertex v for which $v \sim h$ according to Definition 4.[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 all vertices v' of Q. This means that a decomposition of h 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 h 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 $h = h^1 \in Q_v$ (we have $v = v^1$), we extend the difference vector $h^1 - v^1$ towards the intersection with a lower dimensional face of Q_v (this will be a facet of Q_v, unless we accidentally hit a 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

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

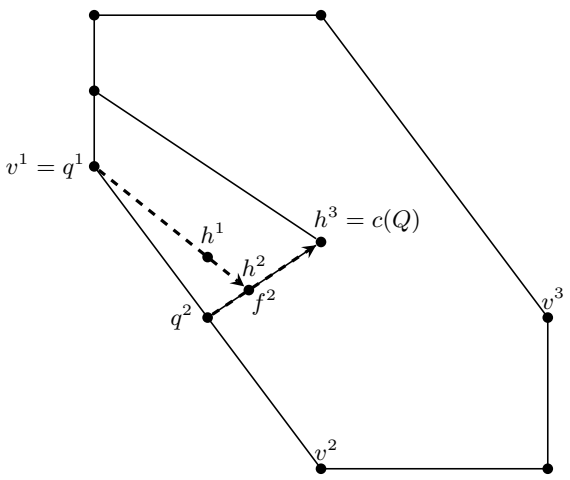

Fig. 3. Visualisation of the decomposition algorithm on a single machine scheduling polytope for three jobs

in time $O(n)$. This is indeed possible because of Lemma 5. 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 h 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 4.

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 $x_j = x_{j+1}$ for all $x \in f^t$ and all $j \in \{1, \ldots, n-1\} \setminus J^t$;
q^t: vertex of f^t;
v^t: vertex of Q such that $q^t = \frac{1}{2}(v + v^t)$;
h^t: point in f^t;
$\tilde{\kappa}_t$: scalar such that $h^t = \tilde{\kappa}_t q^t + (1 - \tilde{\kappa}_t) h^{t+1}$;
κ_t: scalar corresponding to q^t in the convex combination $h = \sum_t \kappa_t q^t$.
λ_t: scalar corresponding to v^t in the convex combination $h = \sum_t \lambda_t v^t$.

Moreover, for ease of notation and without loss of generality, we assume that the given point $h \in Q$ satisfies $h_1 \leq \ldots \leq h_n$.[2]

The subroutine `VERTEX`(J^t) computes the vertex corresponding to the face associated with J^t as follows: Let $J^t(i)$ denote the i-th element in J^t and define

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

Algorithm 1: Decomposition Algorithm

input : processing times p, point $h \in Q$ with $h_1 \leq \ldots \leq h_n$
output: at most n vertices v^t of Q and coefficients $\kappa_t \in [0,1]$

1 $t := 1, h^1 := h, J^1 := \{i \in \{1,\ldots,n-1\} \mid h_i^1 < h_{i+1}^1\}$;
2 let v be the vertex with $v_1 \leq \ldots \leq v_n$;
while $J^t \neq \emptyset$ **do**
3 $\quad q^t := \texttt{VERTEX}(J^t)$;
4 $\quad v^t := 2q^t - v$;
5 $\quad \tilde{\kappa}_t := \min_{j \in J^t}(h_{j+1}^t - h_j^t)/(q_{j+1}^t - q_j^t)$;
6 $\quad h^{t+1} := (h^t - \tilde{\kappa}_t q^t)/(1 - \tilde{\kappa}_t)$;
7 $\quad J^{t+1} := \{i \in J^t \mid h_i^{t+1} < h_{i+1}^{t+1}\}$;
8 $\quad \kappa_t := (1 - \sum_{\tau=1}^{t-1} \kappa_\tau)\tilde{\kappa}_t$;
9 $\quad t := t + 1$;
10 $q^t := h^t$;
11 $v^t := 2q^t - v$;
12 $\kappa_t := 1 - \sum_{\tau=1}^{t-1} \kappa_\tau$;
13 $\lambda_1 := \frac{1}{2} + \frac{1}{2}\kappa_1$;
for $\tau \in \{2,\ldots,t\}$ **do**
14 $\quad \lambda_\tau := \frac{1}{2}\kappa_\tau$;

$J^t(0) = 1$. 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 7. *For any $h \in Q$, Algorithm 1 outputs a convex combination of vertices of Q for h in $\mathrm{O}(n^2)$ time.*

Proof. Q_v is chosen such that $h \in Q_v$. From line 7 of the algorithm we have that inequalities (7) are tight for h^t and all $j \notin J^t$. Thus, if $h^t \in Q_v$ implies that $h^{t+1} \in Q_v$, then $h^t \in f^t$ for all t. 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 $h^t \in Q_v$. Then, of course, h^t and q^t satisfy (8), and we have

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

for all $k \in \{1,\ldots,n\}$. Since we have

$$\tilde{\kappa}_t = \min_{j \in J^t} \frac{h_{j+1}^t - h_j^t}{q_{j+1}^t - q_j^t} \ , \tag{10}$$

and both h^t and q^t satisfy inequalities (7), we have the following for all $j \in \{1, \ldots, n-1\}$:

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

It follows from (10) that $\tilde{\kappa}_t \geq \frac{h_{j+1}^t - h_j^t}{q_{j+1}^t - q_j^t}$ for all $j \in J^t$. Thus,

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

Hence, h^{t+1} satisfies (7)–(9). From Lemma 5, we have $h^{t+1} \in Q_v$ and, therefore, $h^{t+1} \in f^{t+1}$.

In addition, (10) ensures that for at least one $j \in J^t$, we have $h_{j+1}^{t+1} = h_j^{t+1}$ 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 $h^t = c(Q)$, 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

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

Iteratively applying this equality yields

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

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 gives us the following identity for κ_t:

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

So we have

$$h = \sum_{t=1}^{t^*-1} \kappa_t q^t + \left(1 - \sum_{\tau=1}^{t^*-1}\right) h^{t^*} = \sum_{t=1}^{t^*-1} \kappa_t q^t + \kappa_{t^*} q^{t^*} = \sum_{t=1}^{t^*} \kappa_t q^t \ .$$

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

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

where the second equality follows from line 4 of the algorithm. From (10), we obtain that $\tilde{\kappa}_t$ is non-negative for all $t \in \{1, \ldots, t^*\}$. Therefore, also $\kappa_t \geq 0$ and $\sum_{t=1}^{t^*} \lambda_t v^t$ is indeed a correct convex combination.

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 $\mathrm{O}(n^2)$. □

6 Conclusions

The obvious open question is if our algorithm can be generalized for zonotopes. In order to do that, 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.

Acknowledgements

We thank Maurice Queyranne for pointing us to the paper by Yasutake et al. [15].

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 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. Corréa, 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. 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.
12. M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58(1):263–285, 1993.
13. M. Queyranne and A. S. Schulz. Polyhedral approaches to machine scheduling. Preprint 408-1994, TU Berlin, 1994.
14. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory B*, 80:346–355, 2000.
15. 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.
16. G. M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, 1995.