Discrete Applied Mathematics 160 (2012) 1094–1103

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

The interval ordering problem

Christoph Dürr [a], Maurice Queyranne [b,c], Frits C.R. Spieksma [d], Fabrice Talla Nobibon [e,f,*], Gerhard J. Woeginger [g]

[a] CNRS, Université Pierre et Marie Curie, LIP6, F-75252 Paris Cedex 05, France

[b] Sauder School of Business at the University of British Columbia, Vancouver, Canada

[c] CNRS, France

[d] University of Leuven, Operations Research Group, Naamsestraat 69, B-3000 Leuven, Belgium

[e] PostDoc researcher for Research Foundation Flanders, Center for Operations Research and Business Statistics (ORSTAT), Faculty of Business and Economics, KULeuven, Leuven, Belgium

[f] Scientific collaborator Centre for Quantitative methods and Operations Management (QuantOM), HEC-Management School, University of Liège, Belgium

[g] Technical University of Eindhoven, Netherlands

ARTICLE INFO

Article history:
Received 18 October 2010
Received in revised form 7 December 2011
Accepted 14 December 2011
Available online 10 January 2012

Keywords:
Dynamic programming
Bottleneck problem
NP-hard
Exposed part
Agreeable intervals
Laminar intervals

ABSTRACT

For a given set of intervals on the real line, we consider the problem of ordering the intervals with the goal of minimizing an objective function that depends on the exposed interval pieces (that is, the pieces that are not covered by earlier intervals in the ordering). This problem is motivated by an application in molecular biology that concerns the determination of the structure of the backbone of a protein.

We present polynomial-time algorithms for several natural special cases of the problem that cover the situation where the interval boundaries are agreeably ordered and the situation where the interval set is laminar. Also the bottleneck variant of the problem is shown to be solvable in polynomial time. Finally we prove that the general problem is NP-hard, and that the existence of a constant-factor-approximation algorithm is unlikely.

1. Introduction

Let us consider a set $\mathcal{I}$ of n intervals $I_j = [a_j, b_j)$ for $j = 1, 2, \ldots, n$ on the real line. The *length* of interval I_j is denoted by $|I_j| = b_j - a_j$. As usual, the length of a union of disjoint intervals is the sum of the lengths of the individual intervals. For an interval I_j and a subset $\mathcal{S} \subset \mathcal{I}$ of the intervals, we define $I_j \setminus \bigcup_{I \in \mathcal{S}} I$ to be that part of interval I_j that is not covered by the union of the intervals in $\mathcal{S}$; throughout this text this uncovered part will be called the *exposed part* of I_j relative to subset $\mathcal{S}$. Notice that the exposed part depends upon $\mathcal{S}$ and in general need not be an interval. (If the intervals in $\mathcal{I}$ are pairwise disjoint, then of course the exposed part of *any* interval I relative to *any* set $\mathcal{S}$ of intervals not containing I is the interval I itself.)

We investigate an interval ordering problem that is built around a cost function f that assigns to every interval of length p a corresponding real cost $f(p)$. The cost of a set $\mathcal{S}$ of pairwise disjoint intervals is the sum of the costs of the individual intervals in $\mathcal{S}$. The cost of an ordering $\alpha = (\alpha(1), \alpha(2), \ldots, \alpha(n))$ of all n intervals is the result of summing up in that order,

* Corresponding author at: PostDoc researcher for Research Foundation Flanders, Center for Operations Research and Business Statistics (ORSTAT), Faculty of Business and Economics, KULeuven, 3000 Leuven, Belgium. Tel.: +32 16326960; fax: +32 16 32 66 24.

E-mail addresses: christoph.durr@lip6.fr (C. Dürr), maurice.queyranne@sauder.ubc.ca (M. Queyranne), Frits.Spieksma@econ.kuleuven.be (F.C.R. Spieksma), Fabrice.TallaNobibon@econ.kuleuven.be, tallanob@gmail.com (F.T. Nobibon), gwoegi@win.tue.nl (G.J. Woeginger).

0166-218X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2011.12.020

for every interval, the cost of its exposed part with respect to the previous intervals. Formally, the problem is defined as follows.

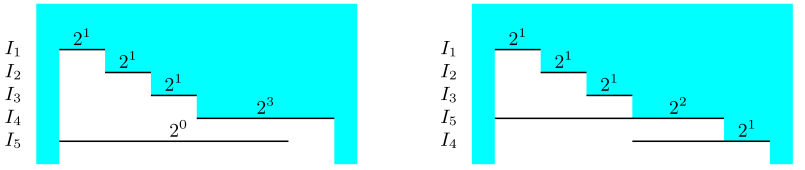
Definition 1 (*The Interval Ordering Problem*)**.** Given a function $f : \mathbb{R} \to \mathbb{R}$ and n intervals $I_1, \ldots, I_n$ over the real line, find an ordering $\alpha \in \Sigma_n$ such that the cost

$$\sum_{k=1}^{n} f\left(\left|I_{\alpha(k)} \setminus \bigcup_{j=1}^{k-1} I_{\alpha(j)}\right|\right),$$

is minimized, where Σ_n denotes the set of all the permutations of $\{1, 2, \ldots, n\}$.

Observe that the interval ordering problem becomes trivial, if all intervals are pairwise disjoint (since then all orderings yield the same cost). In the rest of this paper, an instance of the interval ordering problem is represented by $(\mathcal{I}, f)$ where $\mathcal{I}$ is the set of intervals and f is the cost function.

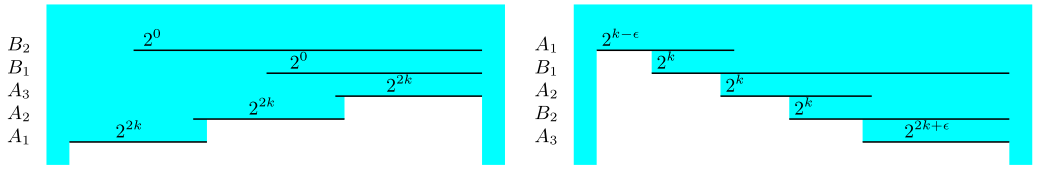
Example 1. Consider the instance that consists of the five intervals $I_1 = [0, 1), I_2 = [1, 2), I_3 = [2, 3), I_4 = [3, 6)$ and $I_5 = [0, 5)$, and the cost function $f(x) = 2^x$. An optimal solution for this instance is given by the sequence $\alpha = (1, 2, 3, 5, 4)$ with a total cost of 12.



greedy ordering with respect to interval length optimal ordering

This example illustrates that in general an optimal solution will *not* sequence the intervals in order of increasing length (and it can be verified that in Example 1 no such sequence can yield the optimal objective value). The next example illustrates that also the following natural greedy algorithm fails: *"Always select the interval with the smallest exposed part relative to the intervals sequenced so far"*. In fact, the greedy algorithm can be arbitrarily bad, as witnessed by the following example.

Example 2. Consider a family of instances, where each instance consists of $2k-1$ intervals: $A_1 = [0,\ 2k), A_2 = [2k-\epsilon,\ 4k)$, $A_3 = [4k-\epsilon,\ 6k), \ldots, A_k = [2k(k-1)-\epsilon,\ 2k^2), B_1 = [k-\epsilon,\ 2k^2), B_2 = [3k-\epsilon,\ 2k^2), B_3 = [5k-\epsilon,\ 2k^2), \ldots, B_{k-1} = [2k^2-3k-\epsilon,\ 2k^2)$, for some constants $k, \epsilon > 0$ with the cost function $f(x) = 2^x$.



greedy ordering with respect to exposed length optimal ordering

A greedy sequence is $(A_1, A_2, \ldots, A_{k-1}, A_k, B_{k-1}, B_{k-2}, \ldots, B_1)$ and achieves a cost of $k2^{2k} + k - 1$, whereas the optimal solution is $(A_k, B_{k-1}, A_{k-1}, B_{k-2}, \ldots, A_2, B_1, A_1)$ and has the cost of $2^{2k+\epsilon} + (2k-3)2^k + 2^{k-\epsilon}$. The ratio between both costs can be made arbitrarily large, by choosing appropriate k and small $\epsilon > 0$.

The contributions of this paper are twofold: on the positive side, we describe polynomial-time algorithms for some natural and fairly general special cases of the problem. On the negative side, we establish the computational complexity (NP-hardness) and the in-approximability of the problem.

The paper is organized as follows. In Section 2, we describe the motivating real world application (in molecular biology) that stands behind the interval ordering problem. In Section 3, we formulate and present a number of special cases of the problem that can be solved in polynomial time. In Section 4, we present complexity and in-approximability results. We conclude in Section 5.

2. Motivation

The interval ordering problem studied in this paper is motivated by a special case of the so-called *distance geometry problem* [8,6,7,5]. Formally, an instance of the latter consists of an undirected graph $G(V, E)$ with positive edge weights $d : E \to \mathbb{R}_+$. The goal is to find an embedding of the vertices into some Euclidean space, say $p : V \to \mathbb{R}^2$, satisfying the requested distances, i.e. for every edge (u, v), we must have $\|p(u) - p(v)\| = d(u, v)$. This problem appears in the areas of graph drawing, localizing wireless sensors, and also in protein folding as we now explain.

The protein folding problem consists of computing the spatial structure of a protein. To simplify the notations, we restrict the problem to the 2-dimensional space; this does not alter its essence. A protein is a huge molecule consisting of many

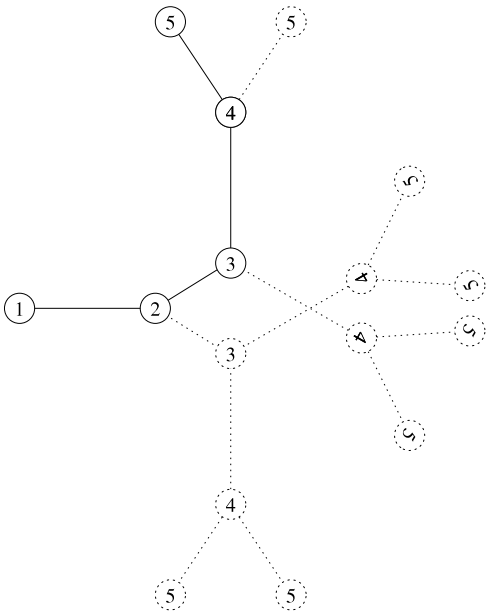


Fig. 1. All 8 possible embeddings of a 5 vertex instance. The embedding described by the bit-string 000 is depicted with solid lines.

different atoms linked together. Consider a simplified version of this problem where we only want to determine the structure of the *backbone* of the protein, that is, we are interested in determining the position of the main string of atoms. The exact sequence of atoms is known, and different approaches are being used in practice to determine their spatial structure. One possibility is to use Nuclear Magnetic Resonance (NMR) to determine the distances between some pairs of atoms. The goal is then to reconstruct a folding that matches the measured distances. This problem, also called *3-dimensional discretizable molecular distance geometry problem*, is NP-hard (see [6]), and different algorithms have been proposed for it; we refer to [7] for a recent overview.

Formally in the problem of reconstructing the backbone of a protein, we are given a vertex set $V = \{1, 2, \ldots, m\}$, enumerating all the atoms of the backbone, together with distances $d(i, j)$ for some pairs $i, j \in V$. It is a common assumption that all $d(i, j)$ with $i + 1 \leq j \leq i + 2$ are given and $|d(i, i + 1) - d(i + 1, i + 2)| < d(i, i + 2) < d(i, i + 1) + d(i + 1, i + 2)$ for all $i = 1, \ldots, m - 2$. The first assumption is motivated by the fact that the NMR reveals distances between atoms which are close to each other. The second assumption is motivated by the chemical fact that in general atoms in molecules are not in co-linear positions. We call $d(i, j)$ a *short range* distance if $i + 1 \leq j \leq i + 2$, and a *long range* distance otherwise.

These assumptions give the problem a combinatorial structure. By translation invariance, without loss of generality we can place vertex 1 in the origin $(0, 0)$. By rotation invariance, without loss of generality we can place vertex 2 in $(d(1, 2), 0)$. Now for vertex 3 there are only two positions respecting the distance $d(1, 3)$, which are the two intersection points of the circle of radius $d(1, 3)$ centered at $(0, 0)$, and the circle of radius $d(2, 3)$ centered at $(d(1, 2), 0)$. In a similar manner, there are exactly two possible positions of vertex $i + 2$ relative to the segment between vertices i and $i + 1$. Therefore with fixed positions for vertices 1 and 2, there are exactly 2^{m-2} embeddings satisfying the short range distances. We could describe each embedding by a binary string $x_3, x_4, \ldots, x_m$, where bit x_i is 1 if and only if the triangle formed by vertices $i - 2, i - 1, i$ is oriented clockwise. But in order to circumvent the symmetry inherent to this problem, we describe each embedding by a binary string $y_3, y_4, \ldots, y_m$, where $y_j = x_3 \oplus x_4 \oplus \cdots \oplus x_j$. See Fig. 1 for illustration.

Now every long range distance $d(i, j)$ implies a constraint on the unknown binary string y. It enforces the bits $y_{i+3}, y_{i+4}, \ldots, y_j$ to those positions that yield an embedding such that atoms i and j are at the right distance. The problem now is to find, as efficiently as possible, values for the bits satisfying all measured distance constraints. Let us now state some notation to arrive at a formal definition of the problem.

Notation. If $a > b$ then $[a, b]$ is the empty interval. For any $a \leq b$ by $\{0, 1\}^{[a,b]}$ we denote the set of all bit strings of length $b - a + 1$ indexed from a to b. If $[a, b] \subseteq [c, d]$ and $y \in \{0, 1\}^{[c,d]}$ then we denote by $y[a, b]$ the restriction of y to the indices from a to b. We use $\{0, 1\}^m$ as a shortcut notation for $\{0, 1\}^{[1,m]}$.

Definition 2 (*The BitString-Reconstruction Problem (BSRP)*)**.** We are given an integer m, and n triplets (a_i, b_i, T_i) where $1 \leq a_i < b_i \leq m$, $T_i : \{0, 1\}^{[a_i,b_i]} \to \{0, 1\}$. The function T_i is an oracle that returns 1 at a single element of the domain. The goal of the BSRP is to find a bit string $y \in \{0, 1\}^m$, such that for all $i = 1, \ldots, n$ we have $T_i(y[a_i, b_i]) = 1$.

The idea is that a triplet in BSRP corresponds to a given distance between atoms i and j with $i + 3 \leq j$ in the folding problem. Formally, a triplet is defined by $(a = i, b = j, T)$ where T is the Boolean function, that accepts a bit string z if and

only if $z = y[a, b]$ for every bit string $y \in \{0, 1\}^m$ describing an embedding where i and j are at the given distance $d(i, j)$. At this point we assume that there is a unique bit string z with this property. In two dimensions, this is equivalent to fixing the position of the third vertex and in three dimensions this boils down to fixing the fourth vertex as well. Already with this strong simplification, we are facing a non-trivial and interesting algorithmic problem.

A straightforward algorithm to solve BSRP employs a brute force approach (see [8] for a similar method called Branch-and-Prune algorithm): by letting ξ be a symbol representing an unspecified bit, the idea of brute-force search is to start with a completely unspecified string $y = \xi^n \in \{0, 1, \xi\}^n$, and to refine it using the distances between atoms i and j with $|i - j| > 3$. More precisely:

Algorithm 1 The BruteForce search algorithm

```
for i = 1, ..., n do
    Let w = y[a_i, b_i] and let ℓ_i be the number of unspecified bits in w
    Search for z such that T_i[z] = 1, ranging over all 2^{ℓ_i} different replacements of ξ in w
    if found then
        replace, in y, y[a_i, b_i] by z
    else
        exit and announce that there is no solution
    end if
end for
Return y, replacing all remaining occurrences of ξ by an arbitrary bit
```

The running time of the BruteForce search algorithm is $O\left(\sum_{i=1}^{n} 2^{\ell_i}\right)$ and it depends on the order in which the triplets in the instance are presented to the algorithm. The only remaining question is in which order to process the given distances. In fact, it is our goal to find an order for the triplets in the instance of the BSRP to be passed to the BruteForce search algorithm in order to minimize the running time. This leads to the interval ordering problem that was described in the introduction with the following additional structure: (i) all data are integral, and (ii) the cost function f is given by $f(x) = 2^x$.

We should point out here that the protein folding application does not necessarily give rise to instances that display the special structures that we will discuss in Section 3: agreeable intervals, and laminar intervals.

3. Some polynomial time solvable cases

In this section, we study some special cases of the interval ordering problem that can be solved in polynomial time. We first consider the case where the intervals are *agreeable*. We derive an $O(n^3)$ dynamic programming algorithm for solving this special case for any cost function f. When the cost function is continuous and convex, we propose a dynamic programming algorithm with time complexity $O(n^2)$. Next, we consider the case where the intervals are *laminar* and describe polynomial-time algorithms for solving the problem when the cost function f is such that the function $g(x) = f(x) - f(0)$ is either super-additive or sub-additive. Finally, we study the bottleneck variant of the interval ordering problem and show that it can be solved in polynomial time when the cost function f is either non-decreasing or non-increasing.

3.1. Agreeable intervals

We say that a set $\mathcal{I}$ of n intervals $I_i = [a_i, b_i)$, for $i = 1, 2, \ldots, n$ is *agreeable* if there exists a permutation γ of $\{1, \ldots, n\}$ such that $a_{\gamma(1)} \leq \cdots \leq a_{\gamma(n)}$ and $b_{\gamma(1)} \leq \cdots \leq b_{\gamma(n)}$. In other words, the ordering of the intervals induced by the left endpoints is the same as the ordering induced by the right endpoints. For ease of exposition, we will assume in the rest of this section that γ is the permutation identity: thus we have $a_1 \leq \cdots \leq a_n$ and $b_1 \leq \cdots \leq b_n$. We can assume that two consecutive intervals I_i and I_{i+1} overlap (that is $a_{i+1} < b_i$) because otherwise this would split the problem into two sub-problems that can be solved independently. In what follows, we first consider the general case with an arbitrary cost function f, followed by a special case where the cost function f is continuous and convex.

3.1.1. Arbitrary cost function

In this section, we consider instances $\left(\mathcal{I}, f\right)$ of the interval ordering problem with $\mathcal{I}$ agreeable and f arbitrary. Observe that in the case of agreeable intervals, after selecting the first interval, the problem decomposes into (at most) two unrelated instances that are each agreeable; we will use this property to derive a dynamic programming algorithm.

For a formal definition of the decomposition, consider the set $\mathcal{I} = \{I_1, I_{i+1} \ldots, I_n\}$ of agreeable intervals. We consider the exposed parts of each of these intervals with respect to $\{I_j\}$, $1 \leq j \leq n$. Since $\mathcal{I}$ is agreeable, the exposed parts are again intervals, and we distinguish between those before I_j and those after I_j.

For convenience define $b_0 = a_1$ and $a_{n+1} = b_n$. For any pair of indices $0 \leq i, k \leq n + 1$ we define the subinstance $\mathcal{I}_{i,k} := \{I_j \cap [b_i, a_k) : i < j < k\}$ (see Fig. 2 for an illustration). Notice that if $b_i \geq a_k$, then $\mathcal{I}_{i,k}$ consists of $k - i - 1$ intervals of zero length. Let $\mathcal{C}(i, k)$ be the cost of an optimum solution to $\left(\mathcal{I}_{i,k}, f\right)$, with $\mathcal{C}(i, k) = 0$ if $\mathcal{I}_{i,k} = \emptyset$. We have the following recursion.

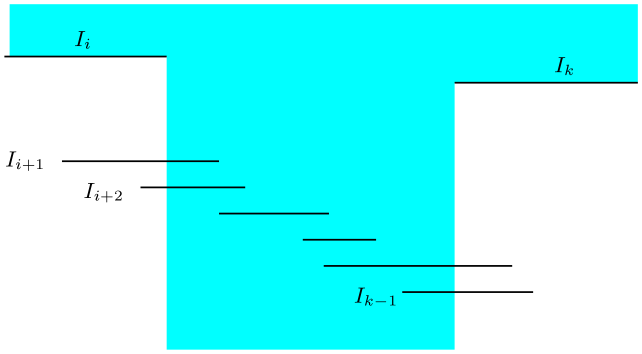


Fig. 2. The subinstance $\mathcal{I}_{i,k}$.

Lemma 1. *For $0 \le i < k \le n+1$ we have $\mathcal{C}(i,k) = 0$ in case $i+1 = k$, and otherwise*

$$\mathcal{C}(i,k) = \min_{i<j<k} \left\{ \mathcal{C}(i,j) + f(|I_j \cap [b_i, a_k)|) + \mathcal{C}(j,k) \right\}.$$

Proof. The case $i+1 = k$ follows from $\mathcal{I}_{i,i+1} = \{\}$ and the remaining case follows from the fact that (1) some interval I_j has to be selected first, and (2) after selecting that interval the problem decomposes into two unrelated instances, $\mathcal{I}_{i,j}$ and $\mathcal{I}_{j,k}$, each being agreeable. □

Theorem 1. *The interval ordering problem $(\mathcal{I}, f)$ with $\mathcal{I}$ agreeable and f arbitrary, can be solved in $O(n^3)$.*

Proof. Lemma 1 leads to a dynamic programming algorithm with $O(n^2)$ variables, each computable in linear time. □

3.1.2. Continuous and convex cost function

In this subsection, we still assume that the intervals in $\mathcal{I}$ are agreeable, but we consider the cost function f to be continuous and convex. Recall that a function f defined on a convex set $\mathbf{dom}(f)$ is *convex* when $f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y)$ for all $x, y \in \mathbf{dom}(f)$, and $0 \le \lambda \le 1$. We need the following result, due to Karamata [4] (see also pages 30–32 in [1]).

Lemma 2. *Given $2q+2$ numbers $\{x_k, y_k\}$, $k = 0, 1, \ldots, q$ satisfying:*

- *$x_0 \ge x_1 \ge \cdots \ge x_q$, and $y_0 \ge y_1 \ge \cdots \ge y_q$,*
- *for each $k = 0, 1, \ldots, q-1$: $\sum_{i=0}^{k} x_i \ge \sum_{i=0}^{k} y_i$, and*
- *$\sum_{i=0}^{q} x_i = \sum_{i=0}^{q} y_i$,*

then, for any continuous, convex function f we have

$$\sum_{i=0}^{q} f(x_i) \ge \sum_{i=0}^{q} f(y_i). \tag{1}$$

Let $(\mathcal{I}, f)$ be an instance of the interval ordering problem where $\mathcal{I}$ is agreeable and contains n intervals $I_i = [a_i, b_i)$, $i = 1, \ldots, n$ and f is continuous and convex. For a given solution to $(\mathcal{I}, f)$ (i.e., a sequence of intervals), we call an interval I_i an *E-interval* if a_i is contained in the exposed part of interval I_i relative to the set of intervals sequenced before I_i (in that solution). Given an integer k, $1 \le k \le n$, let $\mathcal{I}_k$ be the set containing the intervals $I_i = [a_i, b_i)$ for $i = k, \ldots, n$ and let $\mathcal{C}_k$ be the value of an optimal solution to the instance $(\mathcal{I}_k, f)$. Notice that this definition implies that $\mathcal{I} = \mathcal{I}_1$. Further, interval I_k is an E-interval in any feasible solution to $(\mathcal{I}_k, f)$.

Lemma 3. *Let $(\mathcal{I}, f)$ be an instance of the interval ordering problem with $\mathcal{I}$ agreeable and f continuous and convex; and let $\mathcal{I}_k$ defined as above.*

If, in an optimal solution to $(\mathcal{I}_k, f)$, interval I_k is the only E-interval, then

$$\mathcal{C}_k = f(b_k - a_k) + \sum_{i=k+1}^{n} f(b_i - b_{i-1}). \tag{2}$$

Otherwise, in this optimal solution to $(\mathcal{I}_k, f)$, let I_j with $j > k$ be the first E-interval, i.e., the E-interval with minimal a_j.

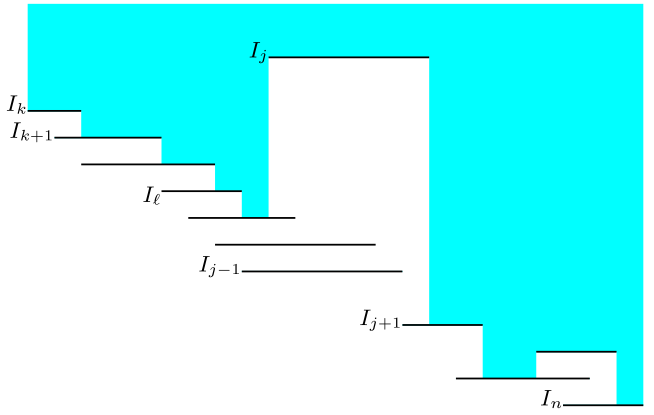


Fig. 3. Recurrence relation of $\mathcal{C}_k$. If I_j is the first E-interval after I_k, then the cost divides into the cost of the intervals between k and j, plus $\mathcal{C}_j$.

If $a_j \le b_k$ *then* $j = k+1$ *and*

$$\mathcal{C}_k = f\left(a_{k+1} - a_k\right) + \mathcal{C}_{k+1}. \tag{3}$$

If $a_j > b_k$, *then*

$$\mathcal{C}_k = f\left(b_k - a_k\right) + \sum_{i=k+1}^{\ell} f\left(b_i - b_{i-1}\right) + f\left(a_j - b_\ell\right) + \left(j - \ell - 1\right) f(0) + \mathcal{C}_j, \tag{4}$$

where I_ℓ *is the latest interval in* $\mathcal{I}_k$ *that satisfies* $b_\ell < a_j$.

Proof. We will show that if interval I_k is the only E-interval, then the optimal sequence to $\left(\mathcal{I}_k, f\right)$ is simply $(k, k+1, \ldots, n)$. Otherwise, if there is another E-interval I_j, where I_j is the first E-interval with $j > k$, then the optimal sequence to $\left(\mathcal{I}_k, f\right)$ is the sequence of the solution to $\left(\mathcal{I}_j, f\right)$ followed by $k, k+1, \ldots, j-1$. See Fig. 3 for illustration.

Case 1: *Interval* I_k *is the only* E*-interval.* We show that $\alpha_0 = (k, k+1, \ldots, n)$ is an optimal sequence to $\left(\mathcal{I}_k, f\right)$. The sequence α_0 partitions $[a_k, b_n)$ into $n-k+1$ nonempty segments, defined by $S_0 = [a_k, b_k)$, $S_i = [b_{k+i-1}, b_{k+i})$ for $i = 1, \ldots, n-k$ (this is true since the intervals are agreeable). Let σ be a permutation of $\{0, 1, \ldots, n-k\}$ such that $|S_{\sigma(i)}| \ge |S_{\sigma(i+1)}|$ for $i = 0, 1, \ldots, n-k-1$; the permutation σ orders the segments induced by α_0 in non-increasing length. Now, let α be some sequence of intervals ($\alpha \ne \alpha_0$) which does not feature another E-interval apart from interval I_k. Clearly, α partitions $[a_k, b_n)$ into less than $n-k+1$ nonempty segments, each segment being defined by a pair from the set $\{a_k, b_k, b_{k+1}, \ldots, b_n\}$ (indeed, notice that the only way to have $n-k+1$ segments is when $\alpha = \alpha_0$). Let us suppose that α partitions $[a_k, b_n)$ into $p+1$ segments ($1 \le p \le n-k-1$) $S'_0, \ldots, S'_p$ satisfying $|S'_0| \ge |S'_1| \ge \cdots \ge |S'_p|$. For convenience set $S'_{p+1} = \cdots = S'_m = \{\}$.

Observation. Any segment S'_i ($i = 0, 1, \ldots, p$) is either identical to a segment S_j for a given $j \in \{0, 1, \ldots, n-k\}$ or is a union of consecutive intervals S_j.

This observation follows from the fact that the segments are defined by points in the set $\{a_k\ b_k, b_{k+1}, \ldots, b_n\}$. We will use this observation to argue that for each $m = 0, 1, \ldots, n-k+1$:

$$\sum_{i=0}^{m} |S'_i| \ge \sum_{i=0}^{m} |S_{\sigma(i)}|. \tag{5}$$

For any m with $p \le m \le n-k$, we have that $\sum_{i=0}^{m} |S'_i| \ge \sum_{i=0}^{m} |S_{\sigma(i)}|$. This is because $\cup_{i=0}^{m} S'_i = [a_k, b_n)$ and $\cup_{i=0}^{m} S_{\sigma(i)} \subseteq [a_k, b_n)$ and the segments are disjoint.

We now show that (5) is also true for $m < p$, by induction on m. For the base case, note that the observation above immediately implies that $|S'_0| \ge |S_{\sigma(0)}|$. For the induction step, we assume that $\sum_{i=0}^{m} |S'_i| \ge \sum_{i=0}^{m} |S_{\sigma(i)}|$. The question now is whether

$$\sum_{i=0}^{m+1} |S'_i| \ge \sum_{i=0}^{m+1} |S_{\sigma(i)}| \tag{6}$$

is true. Let us consider $S_{\sigma(m+1)}$. If each $S_{\sigma(r)}$ with $r \le m$ is contained in the left-hand side of (6), then, using the induction hypothesis $\left(\sum_{i=0}^{m} |S'_i| \ge \sum_{i=0}^{m} |S_{\sigma(i)}|\right)$, the validity of (6) follows. Indeed, if $S_{\sigma(m+1)}$ is also contained in the left-hand side of (6), the inequality is certainly valid, else we know that $S'_{m+1} \ge S_{\sigma(m+1)}$. If there exists an $S_{\sigma(r)}$ with $r \le m$ not contained in

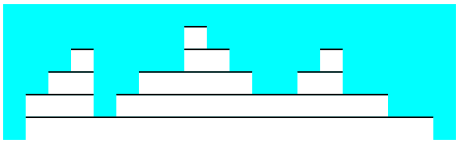


Fig. 4. Illustration of laminar intervals.

the left-hand side of (6), then: $S'_{m+1} \geq S_{\sigma(r)} \geq S_{\sigma(m+1)}$ (where the first inequality holds because the length of a segment $S_{\sigma(j)}$ not contained in the left-hand side of (6) is a lower bound for S'_j). This completes the proof of (5).

We now invoke Lemma 2 by setting $q := n - k + 1$, and for $i = 0, 1, \ldots, n - k + 1$ we set $x_i := |S'_i|$, $y_i := |S_i|$. Clearly, the arguments given above imply that the conditions of Lemma 2 are satisfied. Hence, when f is continuous and convex, the cost of α is greater than or equal to the cost of α_0.

Case 2: *There is another E-interval I_j, where I_j is the first E-interval after I_k.* For this case, we use the following observation. Let I_p and I_q be two consecutive intervals in a solution, and suppose that they are disjoint. Then it does not matter for the cost of the solution whether I_p or I_q is processed first of the two. Now since I_j is an E-interval, it must be processed before all intervals I_i that contain a_j (otherwise I_j is not an E-interval), and it can be processed before all intervals I_i with $b_i < a_j$. Thus, we conclude that I_j is processed before intervals indexed by $k, k + 1, \ldots, j - 1$. Since the intervals are agreeable, the exposed parts (after processing I_j) of the intervals before I_j are disjoint with the intervals with index greater than j. Therefore we can assume that the intervals with index $k, \ldots, j-1$ are processed after the intervals with index $j, \ldots, n$. And of course, the latter intervals are processed optimally by a sequence of the solution to $(\mathcal{I}_j, f)$. Let I_ℓ with $\ell < j$ be the latest interval that does not intersect interval I_j. Notice that by the choice of j, the optimal sequence of the intervals $I_k, \ldots, I_\ell$ contains only one E-interval, namely I_k. Hence, that optimal sequence has a cost of $f(b_k - a_k) + \sum_{i=k+1}^{\ell} f(b_i - b_{i-1})$. Finally, we need to take into account the intervals $I_{\ell+1}, \ldots, I_{j-1}$. Thus, we incur $f(a_j - b_\ell)$ for the exposed part between b_ℓ and a_j, corresponding to interval $I_{\ell+1}$, and we incur a cost of $f(0)$ for each of the remaining intervals. Notice that all intervals $I_{\ell+1}, \ldots, I_{j-1}$ are completely covered in this sequence. This completes the proof of this lemma. □

Theorem 2. *The interval ordering problem $(\mathcal{I}, f)$ with $\mathcal{I}$ agreeable and f convex and continuous, can be solved in $O(n^2)$.*

Proof. The $O(n^2)$-time complexity of the dynamic program following from Lemma 3 (see Eqs. (2) and (3)) is explained by the fact that there are n variables and each is a minimization over $O(n)$ values. □

3.2. Laminar intervals

Let $(\mathcal{I}, f)$ be an instance of the interval ordering problem where $\mathcal{I}$ contains n intervals $I_i = [a_i, b_i)$, for $i = 1, 2, \ldots, n$. We say that the set $\mathcal{I}$ of intervals is *laminar* if for any two intervals I_i and I_j in $\mathcal{I}$, either $I_i \cap I_j = \emptyset$ or one is included in the other. See Fig. 4 for an illustration.

An ordering α *respects the inclusions* if for any two intervals I_i and I_j with $I_i \subsetneq I_j$ we have that i appears before j in α.

Lemma 4. *Let $(\mathcal{I}, f)$ be an instance of the interval ordering problem with $\mathcal{I}$ laminar. If the function g defined by $g(x) = f(x) - f(0)$ is super-additive i.e., $g(x + y) \geq g(x) + g(y)$ then any ordering that respects the inclusions is an optimal solution to $(\mathcal{I}, f)$.*

Proof. Let α be an arbitrary order of optimal cost. We will show that there is another order respecting the inclusions and having a cost not greater than that of α.

Suppose that α does not respect the inclusions. Then there is a pair i, j with $I_i \subsetneq I_j$ and j appears before i in α. Let α' be the result of placing j right after i in the order α. Let x be the length of the exposed part of I_j in α', and y be the length of the exposed part of I_i in α'. Then $x + y$ is the length of the exposed part of I_j in α. Therefore the contribution of I_i and I_j to the cost of α is $f(x + y) + f(0)$ while their contribution to the cost of α' is $f(x) + f(y)$.

Since g is super-additive, it follows that $f(x + y) + f(0) \geq f(x) + f(y)$. We conclude that the cost of α' is not more than the cost of α. By repeating the argument, we eventually obtain an inclusion respecting order with optimal cost. □

An inclusion respecting order can be found simply by sorting the intervals in increasing order of their lengths, breaking ties arbitrarily.

Theorem 3. *The interval ordering problem $(\mathcal{I}, f)$ with $\mathcal{I}$ laminar and f such that the function $g(x) = f(x) - f(0)$ is super-additive, can be solved in $O(n \log n)$ time*

Proof. Immediate. □

We show in Section 4 that the time complexity of any exact algorithm for solving this problem cannot be better than the time complexity of algorithms for sorting. Thus, when restricting ourselves to comparison-based algorithms, the bound in Theorem 3 is the best possible (see [2]).

Remark 4. Notice that the problem $(\mathcal{I}, f)$ with $\mathcal{I}$ laminar and f such that the function $g(x) = f(x) - f(0)$ is sub-additive, can also be solved in $O(n \log n)$ time by sorting the intervals in decreasing order of their lengths.

3.3. Bottleneck variant of the interval ordering problem

In this subsection, we consider the bottleneck variant of the interval ordering problem. Referring to the application described in Section 2, instead of looking for the exact complexity $O\left(\sum_{i=1}^{n} 2^{\ell_i}\right)$ of the BruteForce search algorithm, we focus on the maximum power of two that dominates this complexity. Hence, solving the bottleneck variant gives us a solution that is an approximation of the optimal solution to the interval ordering problem. The bottleneck variant is explicitly defined as follows.

Definition 3 (*The Bottleneck Interval Ordering Problem (BIO)*)**.** Given a function f and a set $\mathcal{I} = \{I_1, \ldots, I_n\}$ of intervals over the real line, find an ordering $\alpha \in \Sigma_n$ that minimizes the value

$$\max_{k=1,\ldots,n} f\left(\left|I_{\alpha(k)} \setminus \bigcup_{j=1}^{k-1} I_{\alpha(j)}\right|\right).$$

A greedy algorithm for this variant would iteratively select the interval with the smallest exposed part. A formal description is given in Algorithm 2.

Algorithm 2 Smallest Exposed Part Algorithm

```
for every i = 1, ..., n, let I'_i := I_i be the exposed part of the i-th interval
Let S = {1, ..., n} be the set of yet unselected intervals
for j = 1, ..., n do
    Identify i ∈ S such that |I'_i| is minimal.
    α(j) := i
    S := S\{i}
    for k ∈ S do
        update I'_k := I'_k \ I'_i
    end for
end for
Return α
```

In the rest of this section we prove that Algorithm 2 solves instances of BIO when the cost function f is non-decreasing. Our proof is based on the following lemmas.

Lemma 5. *Let $(\mathcal{I}, f)$ be an instance of BIO with a non-decreasing cost-function f. There exists an optimal solution to $(\mathcal{I}, f)$ starting with a smallest interval.*

Proof. We prove this result by contradiction. Let $(\mathcal{I}, f)$ be an instance of BIO with a non-decreasing cost-function f. Assume that each optimal sequence to $(\mathcal{I}, f)$ does not start with a smallest interval. Consider an optimal sequence $\alpha = (\alpha(1), \ldots, \alpha(i_0), \ldots, \alpha(n))$ to $(\mathcal{I}, f)$ with the corresponding optimal value $val(\alpha)$. Clearly, $val(\alpha) \geq f(|I_{\alpha(1)}|)$. Let $I_{\alpha(i_0)}$ be the first smallest interval in α, i.e., $|I_{\alpha(i_0)}| \leq |I_{\alpha(j)}|$ for all $j \in \{1, \ldots, n\}$ and $|I_{\alpha(i_0)}| < |I_{\alpha(j)}|$ for all $j \in \{1, \ldots, i_0 - 1\}$. Consider now the sequence $\alpha' = (\alpha(i_0), \alpha(1), \ldots, \ldots, \alpha(n))$ where $\alpha(i_0)$ is moved to the first position in α. It is clear that this move only affects the intervals that were sequenced before $I_{\alpha(i_0)}$ in α. Further, since f is non-decreasing, and the length of each affected interval cannot have become larger, and $|I_{\alpha(i_0)}| \leq |I_{\alpha(1)}|$, we conclude that the objective value achieved by α' does not exceed $val(\alpha)$. Therefore, α' is also an optimal sequence to $(\mathcal{I}, f)$, which is a contradiction. □

Given an arbitrary instance $(\mathcal{I}, f)$ of BIO with n intervals and $I_{i_0} \in \mathcal{I}$ a smallest interval, we define the I_{i_0}-reduced instance $(\bar{\mathcal{I}}_{i_0}, f)$ with $n - 1$ intervals as follows. For any interval $I_j \in \mathcal{I}$,

1. if $I_j \neq I_{i_0}$ and $I_j \cap I_{i_0} = \emptyset$ then $I_j \in \bar{\mathcal{I}}_{i_0}$;
2. if $I_j \neq I_{i_0}$ and $I_j \cap I_{i_0} \neq \emptyset$ then $I_j \setminus I_{i_0} \in \bar{\mathcal{I}}_{i_0}$.

Furthermore, the real line is adapted accordingly such that $I_j \setminus I_{i_0}$ is an interval for all $j \neq i_0$.

Lemma 6. *Let $(\mathcal{I}, f)$ be an instance of BIO with a non-decreasing cost-function f. Let $I_{i_0} \in \mathcal{I}$ be a smallest interval and α_{i_0} be an optimal sequence to $(\bar{\mathcal{I}}_{i_0}, f)$. Then (i_0, α_{i_0}) is an optimal sequence to $(\mathcal{I}, f)$.*

Proof. Let $val(\alpha_{i_0})$ be the total cost of the optimal solution α_{i_0} to $\left(\bar{\mathcal{I}}_{i_0}, f\right)$.

1. If $f\left(|I_{i_0}|\right) \geq val(\alpha_{i_0})$ then $\left(i_0, \alpha_{i_0}\right)$ is clearly an optimal sequence to $\left(\mathcal{I}, f\right)$ (recall that I_{i_0} is a smallest interval).
2. On the other hand, suppose that $f\left(|I_{i_0}|\right) < val(\alpha_{i_0})$. Lemma 5 implies that there exists an optimal sequence α' to $\left(\mathcal{I}, f\right)$ starting with I_{i_0}. After selecting I_{i_0}, the instance that remains is $\left(\bar{\mathcal{I}}_{i_0}, f\right)$, for which α_{i_0} is optimal. Therefore, (i_0, α_{i_0}) is an optimal solution to $\left(\mathcal{I}, f\right)$.

Theorem 5. *The Bottleneck Interval Ordering problem $\left(\mathcal{I}, f\right)$ with $\mathcal{I}$ arbitrary and f non-decreasing can be solved in $O(n^2)$.*

Proof. This result follows from Lemmas 5 and 6. □

Remark 6. Notice that if the function f is non-increasing then the instances of BIO with this cost function can be solved with an $O(n^2)$-time algorithm similar to Algorithm 2 where in line 6 instead of taking the interval with the smallest exposed part, we take the interval with the longest exposed part.

4. Complexity results

This section presents a number of negative results on the computational complexity of the interval ordering problem. Our first result shows that even the easy special cases discussed in Section 3.1 are not completely straightforward, and shows the optimality of the algorithm given in Section 3.2.

Theorem 7. *The interval ordering problem is at least as hard as the SORTING problem, even if* (a) *the intervals are agreeable, or if* (b) *the intervals form a laminar set. Consequently, every comparison-based algorithm for these special cases will have a time complexity $\Omega(n \ln n)$.*

Proof. Let $x_1, \ldots, x_n$ be an arbitrary sequence of positive real numbers that form an instance of the SORTING problem. We construct a corresponding instance of the interval ordering problem that consists of the intervals $I_j = [0, x_j)$ for $j = 1, \ldots, n$, together with the cost function $f(x) = 2^x$. Note that this set of intervals is agreeable and laminar.

Note that the cost function $f(x)$ is such that $g(x) = f(x) - f(0)$ is super-additive on the positive real numbers. This observation easily yields that the optimal ordering of the intervals must sequence them by increasing right endpoint, and hence induces a solution to the SORTING problem. □

Next, we will discuss the computational complexity of the interval ordering problem. We will show that there is little hope for finding a polynomial-time algorithm for solving the interval ordering problem in general. The reduction is from the following variant of the NP-hard PARTITION problem [3, problem SP12].

Instance: A finite set $\{q_1, q_2, \ldots, q_n\}$ of n positive integers with sum $2Q$; an integer k.

Question: Does there exists an index set $J \subseteq \{1, \ldots, n\}$ with $|J| = k$, such that $\sum_{j \in J} q_j = \sum_{j \notin J} q_j = Q$?

Lemma 7. *Let I be an instance of PARTITION and N be an integer such that $2^{N-1} > 2^n Q + k$. Then there exists an instance $\left(\mathcal{I}, f\right)$ of the interval ordering problem that can be built from I in polynomial time and that satisfies the following conditions:*

(i) *If I is a YES-instance of PARTITION, then the total cost of an optimal sequence to $\left(\mathcal{I}, f\right)$ is at most $2^n Q + n - k$.*

(ii) *If I is a NO-instance of PARTITION, then the total cost of an optimal sequence to $\left(\mathcal{I}, f\right)$ is at least $2^N + 2^n Q + n - k$.*

Proof. Consider an arbitrary instance I of PARTITION. We build the instance $\left(\mathcal{I}, f\right)$ of the interval ordering problem as follows. The cost function $f : \mathbb{N} \to \mathbb{N}$ is defined by $f(x) = 0$ if x is a power of two, and by $f(x) = x$ otherwise. The set $\mathcal{I}$ consists of $n + 2$ intervals. First, for $i = 1, \ldots, n$ there is a so-called *element-interval* of length $\ell_i = 2^n q_i + 1$. These element-intervals are pairwise disjoint and put back to back, so that they jointly cover the interval from 0 to $L := \sum_{i=1}^{n} \ell_i = 2^{n+1} Q + n$. Second, there is a so-called *dummy-interval* of length $\ell_{n+1} = 2^N - 2^n Q - k$ that goes from L to $L + \ell_{n+1}$. Third, there is the so-called *main-interval* that covers all other intervals, and that goes from 0 to $L + \ell_{n+1}$; hence the length of the main-interval is $2^N + 2^n Q + n - k$. Clearly, this construction of $\left(\mathcal{I}, f\right)$ can be done in polynomial time. Next, we prove (i) and (ii).

(i) Assume that I is a YES-instance of PARTITION, and let $J \subseteq \{1, \ldots, n\}$ be the corresponding index set. First select the element-intervals that correspond to the q_i with $i \notin J$, then the main-interval, followed by the remaining element-intervals, and finally the dummy-interval. For the first batch of element-intervals we pay a cost of $2^n Q + n - k$. The exposed part of the main-interval then has length 2^N, which yields a cost of 0. This reduces the exposed part of all remaining intervals down to length 0. The overall total cost is then $2^n Q + n - k$.

(ii) Now assume that I is a NO-instance of PARTITION. We claim that in this case no sequencing can ever turn the length of the exposed part of the main-interval into a power of 2. Then the total cost is proportional to the total length, and hence at least $2^N + 2^n Q + n - k$. It remains to prove the claim. We distinguish two cases. The first case deals with the time before the dummy-interval is sequenced. At such a point in time the length of the exposed part of the main-interval equals the length of the dummy-interval plus the length of the currently unsequenced element-intervals. The length of the dummy-interval is $2^N - 2^n Q - k > 2^{N-1}$. The length of the dummy interval plus the length of all element-intervals is $2^N + 2^n Q + n - k < 2^{N+1}$.

Hence the only candidate power of 2 would be 2^N. But in this case the subset of the element-intervals would have a total length of $2^nQ + n - k$, which would correspond to a solution to the PARTITION instance I; a contradiction. The second case deals with the time after the dummy-interval has been sequenced. At such a point in time the length of the exposed part of the main-interval equals the length of the remaining unsequenced element-intervals. But the total length of such a subset can never be a power of 2. □

Of course, Lemma 7 immediately yields the NP-hardness of the interval ordering problem. We will also derive from it the inapproximability of this problem. Suppose for the sake of contradiction that there is a polynomial-time approximation algorithm with some finite worst-case guarantee θ. Pick an arbitrary instance I of PARTITION, and choose an integer N sufficiently large so that

$$2^N > (\theta - 1)(2^nQ + n - k). \tag{7}$$

Then N is roughly $n + \log Q + \log\theta$, and hence its length is polynomially bounded in the size of instance I. We construct the instance $(\mathcal{I}, f)$ of the interval ordering problem as indicated in the proof of Lemma 7, and feed it into the approximation algorithm. The answer allows us to decide in polynomial time whether instance I is a YES-instance or a NO-instance of PARTITION.

Theorem 8. *The interval ordering problem is NP-hard. Furthermore, the interval ordering problem does not possess any polynomial-time approximation algorithm with constant worst-case guarantee, unless $P = NP$.*

5. Conclusion

This paper studies the problem of ordering a given set of intervals on the real line to minimize the total cost, where the cost incurred for an interval depends on the length of its exposed part when it is processed. We were motivated to consider this problem by an application in molecular biology. Our work proposes polynomial-time algorithms for some special cases of the problem. Furthermore, we prove that the problem is NP-hard and is unlikely to have a constant-factor-approximation algorithm.

Some interesting special cases of our problem remain open. For instance, when the cost function is continuous and convex, (and without any assumption on the structure of the intervals), the complexity of the problem is not settled. In particular, the case $f(x) = 2^x$ is interesting (note that our NP-hardness construction does not yield anything for this particular cost function). Finally, it would be interesting to see other special cases that can be solved in polynomial time.

Acknowledgments

We thank an anonymous reviewer, as well as Leo Liberti, for comments on an earlier version of this manuscript.

Gerhard Woeginger has been supported by the Netherlands Organization for Scientific Research (NWO), grant 639.033.403, by DIAMANT (an NWO mathematics cluster), and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

Maurice Queyranne's research was supported in part by a Discovery grant from the Natural Sciences Research Council (NSERC) of Canada.

References

[1] E.F. Beckenbach, R. Bellman, Inequalities, third printing, Springer-Verlag, 1971.
[2] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., McGraw-Hill, 2002.
[3] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., 1979.
[4] J. Karamata, Sur une inégalité relative aux functions convexes, Publications Mathématiques de l'Université Belgrade 1 (1932) 145–148.
[5] C. Lavor, J. Lee, A.L.L. Liberti, A. Mucherino, M. Sviridenko, Optimization Letters (2011) doi:10.1007/s11590-011-0302-6.
[6] C. Lavor, L. Liberti, N. Maculan, A. Mucherino, Computational Optimization and Applications (2011) doi:10.1007/s10589-011-9402-6.
[7] C. Lavor, L. Liberti, N. Maculan, A. Mucherino, European Journal of Operational Research (2011) doi:10.1016/j.ejor.2011.11.007.
[8] A. Mucherino, C. Lavor, L. Liberti, Optimization Letters (2011) (Online first article).