Complexity of Min-Max Subsequence Problems

Wil Michiels[1,2] Jan Korst[1]

[1]Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
[2]Technische Universiteit Eindhoven, Dept. of Mathematics and Computing Science, Eindhoven, The Netherlands
michiels@natlab.research.philips.com

Abstract

We determine the computational complexity of the problem of ordering a set of n numbers, either into a sequence or a cycle, such that the maximum sum of any k successive numbers is minimal. Both problems are easy for $k = 2$ and strongly NP-hard for any $k \geq 3$. However, the two problems allow a polynomial-time approximation scheme that is linear in n.

keywords: computational complexity, polynomial-time approximation scheme, min-max subsequence problem

1 Introduction

In this paper we study the complexity of ordering a set of n numbers such that the maximum sum of any k successive numbers is minimal. In the k-Min-Max Subsequence Problem (k-MSP) we are asked to order the numbers into a sequence, whereas in the Cyclic k-Min-Max Subsequence Problem (k-CMSP) we have to construct a cyclic ordering. The latter problem is also called k-cyclic scheduling [4, 6].

Definition 1. (k-MSP) A problem instance I is given by a set $A = \{0, 1, \ldots, n-1\}$ of n items, where each item $j \in A$ has a nonnegative integer size a_j with $a_0 \leq a_1 \leq \cdots \leq a_{n-1}$. Find a permutation π of the items in A, such that

$$f_{I,k}(\pi) = \max_{0 \leq i \leq n-k} \sum_{j=0}^{k-1} a_{\pi(i+j)}$$

is minimal. □

Definition 2. (k-CMSP) A problem instance I is given by a set $A = \{0, 1, \ldots, n-1\}$ of n items, where each item $j \in A$ has a nonnegative integer size a_j with $a_0 \leq a_1 \leq \cdots \leq a_{n-1}$. Find a permutation τ of the items in A, such that

$$g_{I,k}(\tau) = \max_{0 \leq i < n} \sum_{j=0}^{k-1} a_{\tau((i+j) \bmod n)}$$

is minimal. □

Michiels & Korst [7] give an application of k-MSP in effectively storing multimedia data on multi-zone hard disks. Furthermore, Koop [4] identifies k-CMSP in the construction of cyclic manpower schedules and Choi, Kang & Baek [2] show that it arises in the balancing of turbine fans.

Both problems are easy for $k = 2$. Michiels & Korst [7] prove that an optimal permutation for 2-MSP is defined by $\pi(2i) = n-i-1$ and $\pi(2i+1) = i$ for $i \geq 0$, whereas Koop [4] shows that an optimal solution to 2-CMSP is defined by $\tau(2i) = i$ and $\tau(2i+1) = n-i-1$ for $i \geq 0$. Hence, an optimal permutation is constructed by alternately assigning the largest and smallest unassigned item to the first free position, where we start with the largest item for 2-MSP and with the smallest item for 2-CMSP.

For any $k \geq 3$, Margot [6] shows that k-CMSP is NP-hard in the strong sense. First, he proves that the problem is strongly NP-hard for $k = 3$ and next he generalizes the result to any $k \geq 3$. Along the same lines as for 3-CMSP, it can be proved that 3-MSP is also strongly NP-hard. However, unlike for k-CMSP, the step to generalize the case $k = 3$ to any $k \geq 3$

is not trivial. In Section 2, we prove that k-MSP is NP-hard in the strong sense for each $k \geq 3$.

Consequently, the existence of a polynomial-time approximation scheme (PTAS) is the best we can hope for. Its existence has remained an open problem, which is settled in Sections 3 and 4, where we derive a PTAS with a running time that is linear in n for both problems.

We end this section with introducing some definitions and notational conventions used in this paper. If not already clear from the context, we call a permutation linear if it is a solution of k-MSP and cyclic if it is a solution of k-CMSP. Furthermore, we add a star as superscript to our notation to indicate optimality. For example, $f^*_{I,k}$ gives the optimal objective value of problem instance I for k-MSP. Finally, to refer to k successive items in a permutation, we both use the term *subsequence* and *window*. A subsequence refers to any k successive items, whereas a window has to start at position ik for some integer i.

2 Complexity of k-MSP

Theorem 1. *k-MSP is NP-hard in the strong sense for each $k \geq 3$.*

Proof. When referring to k-MSP in this proof, we mean its decision variant in which we are asked whether a permutation exists with cost at most B for some given integer bound B.

3-MSP can be proved to be strongly NP-complete for the special case that n is a multiple of 3 and $\sum_i a_i = \frac{n}{k}B$ in a similar way as Margot [6] proves this result for 3-CMSP. Correspondingly, we assume that k divides n and $\sum_i a_i = mB$ when considering k-MSP in this proof, where $m = \frac{n}{k}$. Clearly, k-MSP $\in$ NP for each $k \geq 4$. Hence, to prove the theorem, it suffices to give a polynomial-time algorithm that reduces k-MSP to $(k+1)$-MSP for any $k \geq 3$.

Consider an instance I of k-MSP. We assume that $k+1$ divides B, $B > k(k+1)$, and $a_i > \frac{B}{k+1}$ for all $i = 0, 1, \ldots, n-1$. This assumption is made without loss of generality since first adding $B+2$ to all item sizes and $k(B+2)$ to B and next multiplying the item sizes and B by $k+2$, results in an instance that satisfies these conditions and has a solution if and only if the original instance has a solution. Now, we choose the corresponding instance I' of $(k+1)$-MSP to have bound $B' = B$ and the item sizes from I plus m times 0, k times 1, $k+1$ times $\frac{B}{k+1}$, and one time $B-k$. Note that a permutation for I consists of m windows of k items, while a permutation for I' consists of $m+2$ windows of $k+1$ items. We now prove that I has a solution if and only if I' has one.

Let π be a solution of I. We define a permutation π' for I' such that the first window is given by $w'_1 = [(B-k), 1, 1, \ldots, 1]$, the second window w'_2 by $k+1$ times $\frac{B}{k+1}$, and the $(j+2)$th window w'_{j+2} by w_j preceded by a 0, where w_j is the jth window of π; see Figure 1. It can be verified that if $f_{I,k}(\pi) = B$, then $f_{I',k+1}(\pi') = B$, as well. Hence, π' is a solution of I'.

Next, let π' be a solution of I'. As $B > k(k+1)$ implies $\frac{B}{k+1} > k$, we have that the window with $B-k$ contains, next to $B-k$, only zeros and ones. Hence, all mk item sizes from I and the $k+1$ occurrences of $\frac{B}{k+1}$ are assigned to the other $m+1$ windows of π'. Consequently, because $a_i > \frac{B}{k+1}$, $f_{I',k+1}(\pi') = B$, and $\sum_{i=1}^{n} a_i = mB$, we have that π' contains a window $\tilde{w}_1$ with $B-k$ and k times one, a window $\tilde{w}_2$ with $k+1$ times $\frac{B}{k+1}$, and windows $\tilde{w}_j$, $3 \leq j \leq m+2$, with one item size zero and k item sizes from I that sum up to B.

Assume that $\tilde{w}_1$ is the first window and $\tilde{w}_2$ is the second window in π'. Note that this means that we must have $\tilde{w}_1 = [B-k, 1, 1, \ldots, 1]$. Then each window w'_j with $3 \leq j \leq m+2$, starts with item size zero because an item size a_i cannot be in a subsequence containing k other item sizes at least $\frac{B}{k+1}$. Hence, removing the first two windows and all zeros from π' yields a solution of instance I of k-MSP. Similarly, a solution can be constructed if $\tilde{w}_1$ and $\tilde{w}_2$ are the last two windows in π'. Consequently, to prove that I has a solution, it suffices to show that $\tilde{w}_1$ and $\tilde{w}_2$ are either the first or the last two windows of π'. We prove this by contradiction.

Assume that $\tilde{w}_1$ is neither the first nor the last window. Since a subsequence with $B-k$ cannot contain an item size larger than one, the total number of ze-

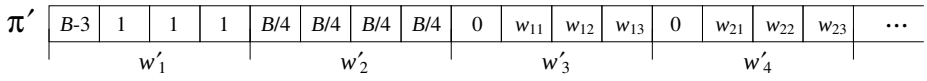

Figure 1. Item sizes in permutation π' for $k = 3$, where w_{ji} is the ith item size of window w_j.

ros and ones in the preceding and succeeding window of $\tilde{w}_1$ is at least k. Hence, either the preceding or succeeding window of $\tilde{w}_1$ contains at least two item sizes at most one. However, this is in contradiction with the observation that except for $\tilde{w}_1$, each other window in π', i.e., each window $\tilde{w}_j$ with $j > 1$, consists of at most one zero and no one.

Next, assume that $w'_1 = \tilde{w}_1$ is the first window in π' and that $w'_j = \tilde{w}_2$ is not the second window, i.e., $j > 2$. Then, window w'_2 consists of a zero and k item sizes from I that sum up to B, where the zero has to be assigned to one of the first k position of w'_2 because the last item size in $\tilde{w}_1$ is a one. As a result, windows $w'_3, w'_4, \ldots, w'_{j-1}$ also contain item size zero at one of the first k position since a subsequence can contain at most k item sizes strictly larger than $\frac{B}{k+1}$. However, if $w'_j = \tilde{w}_2$ is preceded by an item size a_i, then the sum of the subsequence starting with a_i is larger than B, which yields a contradiction. Similarly, we can derive a contradiction for the case that $\tilde{w}_1$ is the last window and $\tilde{w}_2$ is not the second last window. □

3 PTAS for k-MSP

In this section, we present a PTAS for k-MSP for which the running time is linear in n, but exponential in k and $\frac{1}{\varepsilon}$, where ε is the desired precision of approximation. In Section 4, we next discuss how the PTAS can be changed into a PTAS for k-CMSP. The PTAS is based on the work of Hochbaum & Shmoys [3] and Alon, Azar, Woeginger & Yadid [1].

Let for a given problem instance I of k-MSP and precision ε, problem instance I' be obtained from I by rounding down all item sizes to an integer multiple of $\frac{\varepsilon}{k} a_{n-1}$. Then I' contains only item sizes from the set $X = \{x_0, x_1, \ldots, x_q\}$ with $q = \lfloor \frac{k}{\varepsilon} \rfloor$ and $x_i = i \cdot \frac{\varepsilon}{k} a_{n-1}$ for $i = 0, 1, \ldots, q$. We define n_i as the number of occurrences of x_i in I'.

Let π be an arbitrary permutation for I'. Clearly, the sum of any subsequence in π, thus also the cost of π, increases by at most εa_{n-1} when we restore all item sizes in I' to their original value. Consequently, as $a_{n-1} \leq f^*_I$ and $f^*_{I'} \leq f^*_I$, an optimal permutation π of I' satisfies $f_I(\pi) \leq (1+\varepsilon) f^*_I$. This means that to construct a $(1+\varepsilon)$-approximation algorithm for I with an $O(n)$ time complexity, it suffices to derive an $O(n)$ algorithm for determining an optimal permutation for I', where the hidden constant may depend exponentially on k and $\frac{1}{\varepsilon}$. In the remainder of this section, we focus on deriving this algorithm.

First, we define a directed graph G, such that a one-to-one correspondence exists between a permutation π and a walk $w \in W$ of G, where W is a set of walks we define later. Furthermore, we assign weights to the vertices in G in such a way that $f_{I'}(\pi)$ equals the weight of w, where we define the weight of a walk as the maximum weight of any vertex occurring in the walk. Next, we formulate the problem of finding the walk in W with minimum weight as an ILP-problem that can be solved in $O(n)$ time.

The vertex set V of graph G contains a source vertex s, a target vertex t, and all vectors of length i from the Cartesian product X^i of X with $i = 1, 2, \ldots, k$. The weight of the source and target vertices are 0, whereas the weight of a vertex $v = (v_1, v_2, \ldots, v_i) \in X^i$ is given by $\sum_{j=1}^{i} v_j$. The interpretation of a vertex $v \in X^i$ is as follows. Let v be the jth vertex in a walk from s to t, where s is not counted, and let π be the corresponding permutation. If $j \geq k$, then π contains subsequence v and this subsequence ends at the jth position of the sequence. Furthermore, the weight of v gives the sum of the subsequence. On the other hand, if $j < k$, then we will have $j = i$ and the first i item sizes of π are given by $v_1, v_2, \ldots, v_i$, respectively.

The arc set E contains an arc from v to v' in each

of the following cases.

- $v = s$ and $v' \in X$,
- $v \in X^i$, $v' \in X^{i+1}$, and $v' = (v_1, v_2, \ldots, v_i, v'_{i+1})$,
- $v \in X^k$, $v' \in X^k$, and $v' = (v_2, v_3, \ldots, v_k, v'_k)$, or
- $v' = t$.

In the first three cases, we assign the last component of vector v' as label to the arc, i.e., we assign the labels v', v'_{i+1}, and v'_k, respectively. The arcs to the target vertex t are left unlabeled.

Let the length of a walk in a graph be defined by the number of arcs in the walk. Then it follows from the interpretation of the vertices that for any permutation π, graph G contains a walk $w = (s = v_0, v_1, v_2, \ldots, v_n, t)$ of length $n+1$, such that an arc from vertex v_i to v_{i+1} has as label the size of item $\pi(i)$. Furthermore, the maximum weight of any vertex in w equals $f_{I'}(\pi)$. However, if we use this approach to construct a permutation from a walk of length $n+1$, we not necessarily end up in a valid permutation as the number of occurrences of x_i may differ from n_i. This problem is solved by only considering walks from W, where a walk is in W if it has length $n+1$ and if it contains exactly n_i arcs with label x_i.

Now, we can solve the problem of finding an optimal permutation for I' by finding a walk from W with minimum weight. To formulate this problem as an ILP, we need some notation. Let the vertices of G by numbered from 1 to $|V|$, where 1 is the source vertex and $|V|$ the target vertex, let $\omega(i)$ be the weight of vertex i, and let $l(i,j)$ be the label of the arc from vertex i to j. Furthermore, decision variable y_{ij} denotes the number of occurrences of the arc (i,j) in a walk w, binary variable d_i indicates whether vertex i with $i = 2,3,\ldots,|V|-1$ occurs in w ($d_i = 1$) or not ($d_i = 0$), and c gives the weight of the walk. An optimal walk from W, and consequently an optimal permutation of I', can now be constructed from the variables y_{ij} in an optimal solution of the ILP given in Figure 2.

The first set of constraints enforces that c equals the weight of the walk represented by the variables y_{ij} and the second set ties d_i to be 1 if and only if vertex i occurs in a walk, where vertex i is neither the source nor the target vertex. The remaining constraints ensure that the variables y_{ij} determine a walk from W. They respectively state that vertex 1 is left once, vertex $|V|$ is entered once, each other vertex is entered as many times as it is left, the walk has length $n+1$, and the walk has n_α occurrences of x_α. It can be verified that the ILP has $O(|V|^2)$ integer variables, where $|V| = O(q^k)$. The time complexity of the algorithm given by Lenstra [5] for solving ILPs is exponential in the number of integer variables but polynomial in the logarithms of the coefficients. Hence, the problem can be solved in $O(\log^C(n))$ time, where the constant C depends exponentially on $1/\varepsilon$ and k. As the derivative of $\log^C(n)$ approaches 0 for $n \to \infty$, an N exists such that for all $n \geq N$ we have $\log^C(n) \leq n$. This means that we can solve the ILP in $O(n)$ time for fixed k and ε. Furthermore, as the ILP problem can also be constructed in $O(n)$ time, we obtain the desired linear time $(1+\varepsilon)$-approximation algorithm of k-MSP for any fixed $k \geq 3$.

Minimize	c	
such that	$\omega(i) \cdot d_i \leq c$	$1 < i < \lvert V\rvert$
	$d_i \leq \sum_{j=1}^{\lvert V\rvert} y_{ij} \leq n d_i$,	$1 < i < \lvert V\rvert$
	$\sum_{j=1}^{\lvert V\rvert} y_{1j} = 1$	
	$\sum_{i=1}^{\lvert V\rvert} y_{i,\lvert V\rvert} = 1$	
	$\sum_{i=1}^{\lvert V\rvert} y_{ij} = \sum_{i=1}^{\lvert V\rvert} y_{ji}$	$1 < j < \lvert V\rvert$
	$\sum_{i=1}^{\lvert V\rvert} \sum_{j=1}^{\lvert V\rvert} y_{ij} = n+1$	
	$\sum_{l(i,j)=x_\alpha} y_{ij} = n_\alpha$	$0 \leq \alpha \leq q$
	$y_{ij} = 0$	$(i,j) \notin E$
	$y_{ij} \geq 0, y_{ij}$ integer	$1 \leq i,j \leq \lvert V\rvert$,
	$d_i \in \{0,1\}$	$1 < i < \lvert V\rvert$

Figure 2. ILP for finding an optimal permutation for I'.

4 PTAS for k-CMSP

We now discuss how the presented PTAS for k-MSP can be changed into a PTAS for k-CMSP. We construct I' from an arbitrary problem instance I of k-CMSP in a similar way as for k-MSP. Again, we can derive a solution for I with a performance ratio at most $1+\varepsilon$ from an optimal solution for I'. Hence, as for k-MSP it suffices to find an optimal solution for problem instance I'.

Consider a walk $w = (s, v_1, v_2, \ldots, v_n, t) \in W$ in graph G representing a linear permutation π. This walk can also be interpreted as cyclic permutation $\tau = \pi$. However, whereas the maximum weight of the vertices gives $f_{I',k}(\pi)$, it does not necessarily give $g_{I',k}(\tau)$. This problem is solved by not only letting vertex $v_i \in X^i$ with $i = 1, 2, \ldots, k-1$ represent the first i item sizes of τ, but also the last $k-i$ item sizes. Hence, we modify the set of walks W, such that in a walk $w = (s, v_1, v_2, \ldots, v_n, t)$ from W all vertices v_i are from X^k and not only the last $n-k+1$. Correspondingly, we remove the vertices from X^i with $i = 1, 2, \ldots, k-1$ from G and we add an arc from source node s to each vertex $v_i \in X^k$. Furthermore, we impose the additional constraint on a walk $w \in W$ that what vertex v_i with $i = 1, 2, \ldots, k-1$ considers to be the last $k-i$ item sizes of τ corresponds to what v_n considers to be the last $k-i$ item sizes of τ. Formally, this means that $v_{r1} = v_{r+1,n}$ has to hold for all $r = 1, 2, \ldots, k-1$, where v_{ri} gives the rth component of vector v_i. Now, we have a one-to-one correspondence between a walk $w \in W$ in G and a cyclic permutation τ for I'. Furthermore, the maximum weight of a vertex in w gives $g_{I',k}(\tau)$.

Similarly as for k-MSP, we can derive an ILP for the problem of finding a walk $w \in W$ with minimum weight. Thereby, the constraint that $v_{r1} = v_{r+1,n}$ for $r = 1, 2, \ldots, k-1$ is given by

$$y_{i,|V|} = \sum_{j \in \Gamma(i)} y_{1,j}, \quad i = 2, 3, \ldots, |V| - 1,$$

where $j \in \Gamma(i)$ if $v_{r,j} = v_{r+1,i}$ for $r = 1, 2, \ldots, k-1$.

References

[1] N. Alon, Y. Azar, G.J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[2] W. Choi, H. Kang, and T. Baek. A turbine-blade balancing problem. *International Journal of Production Economics*, 60-61:405–410, 1999.

[3] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[4] G.J. Koop. Cyclic scheduling of offweekends. *Operations Research Letters*, 4(6):259–263, 1986.

[5] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[6] F. Margot. Some complexity results about threshold graphs. *Discrete Applied Mathematics*, 49:299–308, 1994.

[7] W. Michiels and J. Korst. Min-max subsequence problems in multi-zone disk recording. *Journal of Scheduling*, 4(5):271–283, 2001.