On Minimum k-Modal Partitions of Permutations[*]

Gabriele Di Stefano[†] Stefan Krause[‡] Marco E. Lübbecke[§]
Uwe T. Zimmermann[¶]

October 5, 2005, revised July 22, 2007

Abstract

Partitioning a permutation into a minimum number of monotone subsequences is $\mathcal{NP}$-hard. We extend this complexity result to minimum partitioning into k-modal subsequences; here unimodal is the special case $k = 1$. Based on a network flow interpretation we formulate both, the monotone and the k-modal version, as mixed integer programs. This is the first proposal to obtain provably optimal partitions of permutations. LP rounding gives a 2-approximation for minimum monotone partitions and a $(k+1)$-approximation for minimum (upper) k-modal partitions. For the online problem, in which the permutation becomes known to an algorithm sequentially, we derive a logarithmic lower bound on the competitive ratio for minimum monotone partitions, and we analyze two (bin packing) online algorithms. These immediately apply to online cocoloring of permutation graphs.

Keywords: Mixed integer program; approximation algorithm; LP rounding; online algorithm; $\mathcal{NP}$-hardness; monotone sequence; k-modal sequence; coloring; cocoloring

MSC (2000): 90C11, 90C27, 05A05, 68Q25

1 Introduction

Given a sequence S of distinct integers, we seek a partition into a minimum number of subsequences with particular monotonicity properties. Research in this direction dates back to the famous Erdős/Szekeres theorem of 1935 stating that every sequence of n distinct reals contains a monotone subsequence of length $\lceil\sqrt{n}\rceil$, see the review [15]. Iteratively extracting longest monotone subsequences yields a partition of size $2\lfloor\sqrt{n}\rfloor$ in $O(n^{1.5})$ [1]. However, finding a minimum size partition into monotone subsequences, or shortly a *minimum monotone partition*, is $\mathcal{NP}$-hard [17]. For fixed ℓ and m, a partition into exactly ℓ increasing and m

[*]An extended abstract appeared in the *Proceedings of Latin American Theoretical Informatics (LATIN)* [16].

[†]Dipartimento di Ingegneria Elettrica, Università dell'Aquila, email: `gabriele@ing.univaq.it`

[‡]Institut für Optimierung und Diskrete Mathematik, Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria, email: `stkrause@opt.math.tugraz.at`

[§]Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1, Straße des 17. Juni 136, D-10623 Berlin, email: `m.luebbecke@math.tu-berlin.de`

[¶]Institut für Mathematische Optimierung, Technische Universität Braunschweig, Pockelsstraße 14, D-38106 Braunschweig, email: `u.zimmermann@tu-bs.de`

decreasing subsequences can be computed in $O(n^{\ell+m})$, if one exists [4]. A minimum monotone partition can be approximated within a factor of 1.71 in $O(n^{2.5})$ [9].

A natural generalization asks for partitions into k-modal subsequences which have at most k internal local extrema. In particular for 1-modal, or *unimodal*, subsequences Chung [5] proves that any permutation of length n contains such a subsequence of length $\lceil\sqrt{3(n-1/4)}-1/2\rceil$, and this is best possible. She also mentions the guaranteed length of $\lceil\sqrt{2n+1/4}-1/2\rceil$ for contained *upper unimodal* subsequences, i.e., subsequences with no internal minimum. One cannot guarantee contained k-modal subsequences longer than $\sqrt{(2k+1)n}$ [5]. Steele [14] proves that the average length of k-modal subsequences of a permutation of size n asymptotically grows as $2\sqrt{(k+1)n}$. Based on these bounds, one can derive results on the size of the partitions generated by recursively extracting a respective longest subsequence. In particular, this greedy approach yields an upper unimodal partition of size $O(\sqrt{n})$ in $O(n^{2.5})$ time [7].

Our Contribution. We show that partitioning a permutation into a minimum number of k-modal (in particular unimodal) subsequences is $\mathcal{NP}$-hard for any fixed $k \geq 1$. On the positive side, we propose a linear programming (LP) rounding algorithm which is the first approximation algorithm for this problem: Its approximation factor is $k+1$ for upper k-modal partitions. An easy observation allows us to derive a combinatorial $1.71(k+1)$-approximation first. In contrast to prior more structural investigations, we are also interested in actually computing optimum partitions. To this end we introduce mixed integer programming (MIP) formulations. We further give the first negative and (weakly) positive results concerning online algorithms for minimum monotone partitions. These immediately apply to cocoloring of permutation graphs, for which no online algorithms were known either.

Motivation. In railroad shunting yards incoming freight trains are split up and re-arranged according to their destinations. In stations and depots passenger trains and trams are parked overnight or during low traffic hours. In either case we are given an ordering of arriving *units*, and we have to decide for each unit on which track it will be stored [2, 7, 18]. Our choice is limited by the fixed number of available tracks and by the way tracks may be accessed: Entrance and exit may be on one or on both ends. The parked units have to leave each track one by one without additional reordering. The goal is to use as few tracks as possible.

Units on each track represent a subsequence of the incoming sequence of units. The different entry/exit combinations lead to increasing, monotone, unimodal, and what we call unimodec subsequences. Such sorting with stacks (queues, deques) is well-studied in computer science [3], e.g., one characterizes which permutations are (not) stack-sortable. Knuth [11] already speaks of re-arranging railroad cars. Our work has been originally motivated by the more algorithmic question for the smallest number of stacks needed for sorting.

2 Preliminaries

Let $S = [s_1, s_2, \ldots, s_n]$ be a permutation of $\{1, \ldots, n\}$. A *subsequence* σ of S is a sequence $\sigma = [s_{i_1}, s_{i_2}, \ldots, s_{i_m}]$ with $1 \leq i_j < i_h \leq n$ for all $j < h$. A sequence is called *increasing* if $s_i < s_j$ for $i < j$. It is called *decreasing* if $s_i > s_j$ for $i < j$. These two cases are also subsumed under *monotone*. An *internal extremum* of S is an index i with $2 \leq i \leq n-1$ and

$s_{i-1} < s_i$, $s_{i+1} < s_i$ or $s_{i-1} > s_i$, $s_{i+1} > s_i$. A sequence S is *k-modal* if it has at most k internal extrema; the first can be of either type. If the first extremum is a maximum, i.e., the first part of the sequence is increasing, S is called *upper k-modal*, otherwise *lower k-modal*. These notions stem from the 1-modal case, well-known as *unimodal*. Some authors specify *upper* or *lower unimodal*.

We use an intuitive set notation and language to work with sequences; e.g., when referring to the sequence consisting of all the elements contained in two sequences we speak of their union. A *unimodec* sequence is the union of a decreasing and an upper unimodal sequence, the two of which have identical first element. Figure 1 sketches the sequences we introduced. A *partition* of S of size m is a collection P of m disjoint subsequences of S, the union of which is precisely S. For a given S we are interested in finding a partition P of minimum size. We name the resulting minimization problem after the type of subsequence into which we partition, that is, we have problems MONOTONE, UNIMODAL, k-MODAL, etc. Obviously, any statement for an UPPER problem analogously holds for its LOWER counterpart. A *cover* of S is a collection of not necessarily disjoint subsequences, the union of which is S. Eliminating multiply covered elements from all but one subsequence, one can turn a cover into a partition without increasing the number of subsequences. This is why our problems are also known as covering a permutation [17].

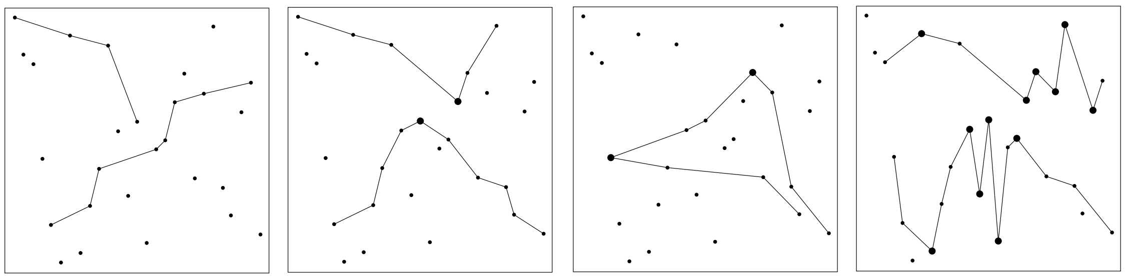

Figure 1: We identify elements s_i, $i = 1, \ldots, n$ with points (i, s_i) in the plane; that is, we use points and elements interchangeably. Feasible subsequences are represented as polygonal lines connecting the contained points: Depicted are decreasing and increasing, lower unimodal and upper unimodal, unimodec, and (upper and lower) 6-modal subsequences.

Relations to Coloring Problems. Increasing and monotone partitions are well studied in graph coloring. The *permutation graph* $G = (S, E)$ associated with a permutation S has an edge $(s_i, s_j) \in E$ if and only if $s_i > s_j$ and $i < j$. An increasing subsequence in S corresponds to an independent set in G, and a decreasing subsequence in S corresponds to a clique. The complement $\bar{G}$ of G is again a permutation graph; it is associated with the *reverse* permutation of $S = [s_1, \ldots, s_n]$, that is, the permutation $[s_n, \ldots, s_1]$.

A partition of the vertices of a graph into independent sets is called a *coloring*. A minimum partition of a permutation graph into *either* independent sets *or* cliques, that is, a solution to problem DECREASING or INCREASING, can be given in $O(n \log n)$, see e.g., [13]. *Cocoloring* a graph asks for partitioning its vertex set into a minimum number of parts in which each part is either an independent set or a clique (so the partition may contain a mixture of both). Thus, in MONOTONE we compute an optimal cocoloring of a permutation graph, which is

$\mathcal{NP}$-hard [17]. In general, any statement about partitioning permutations corresponds to a graph theoretical statement about partitioning the vertex set of permutation graphs.

Di Stefano and Koči [7] formulate UPPER UNIMODAL and UNIMODEC as coloring problems for particular 3- and 4-uniform hypergraphs. In their approach, hyperedges correspond to forbidden subsequences. For example, $[s_i, s_j, s_k]$ is a forbidden subsequence within an upper unimodal sequence if $i < j < k$, $s_i > s_j$ and $s_j < s_k$. If we interpret these forbidden triples as hyperedges of a hypergraph with vertex set S, then a hypergraph coloring corresponds to an upper unimodal partition. In general, uniform hypergraph coloring problems are known to be $\mathcal{NP}$-hard. Our complexity results imply that even these restricted problems are $\mathcal{NP}$-hard.

3 Complexity results

Theorem 1 *Problems k-MODAL, UPPER k-MODAL, and UNIMODEC are $\mathcal{NP}$-hard for any fixed k, in particular for $k = 1$.*

Proof. We use a reduction from the $\mathcal{NP}$-hard problem MONOTONE [17]. Finding a partition of S into at most p monotone subsequences can be reduced to finding a partition of S into at most ℓ increasing and at most $m = p - \ell$ decreasing subsequences, where m and ℓ are part of the input. This requires solving a series of $p + 1$ restricted problems. We reduce from this restricted version.

We identify elements and points (and subsequences and lines) as in Figure 1 above. Having arranged the points of a given permutation S, we construct an extended arrangement of points which can be covered by p subsequences of the requested monotonicity type if and only if the original set S of points can be covered by ℓ increasing and m decreasing lines.

A basic component of our construction is given below, it is denoted by A_h for some $1 \leq h \leq n$. It consists of $h(k+1)$ quadratic blocks arranged in a sequence going rightwards and upwards. Each block contains h points located on a line going rightwards and downwards, see Figure 2. Since we may assume that $k \leq n$, the number of points in A_h is polynomial in n.

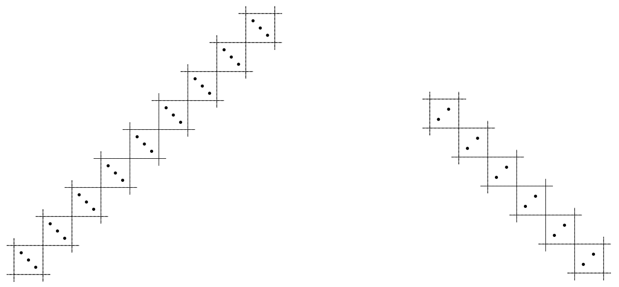

Figure 2: Components A_3 and B_2 for $k = 2$ used in the proof of Theorem 1.

We first prove the following property.

Claim. The component A_h can be covered by h increasing lines and it cannot be covered by $h - 1$ k-modal lines. If A_h is covered by h k-modal lines then each of them has to be increasing somewhere inside A_h.

Evidently, A_h can be covered by h parallel increasing lines where the ith line covers the ith point in each block. For the remaining assertions the proof is by induction. For $h = 1$ the claim is trivially true. So consider $h \geq 2$. A k-modal line can cover more than one point in at most $k + 1$ blocks since for any two consecutive blocks the left one contains a minimum and the right one contains a maximum. Consequently, there are at least $(h - 1)(k + 1)$ blocks where just one point is covered. Now consider a decreasing line. It can cover points of at most one block of A_h. So in this case $h(k + 1) - 1 \geq (h - 1)(k + 1)$ blocks without covered points remain. By the induction hypothesis, in both cases at least $h - 1$ lines are necessary to cover these blocks.

We define components B_h which actually are A_h flipped vertically. For B_h the above claim analogously holds with *increasing* replaced by *decreasing*.

For the reduction we arrange several of such A_h and B_h components around a given instance S of MONOTONE. Starting at S and going leftwards we add a component B_m and then alternatingly components A_p and B_p where the A_p components are below S, and B_m and the B_p components are above S. The total number of these additional components is $k + 1$. By a suitable placement of blocks we ensure that no two points have the same vertical position (and thus we obtain a permutation), see Figure 3 for a suggestive arrangement. Finally, to the right of S we add B_ℓ located above all other components.

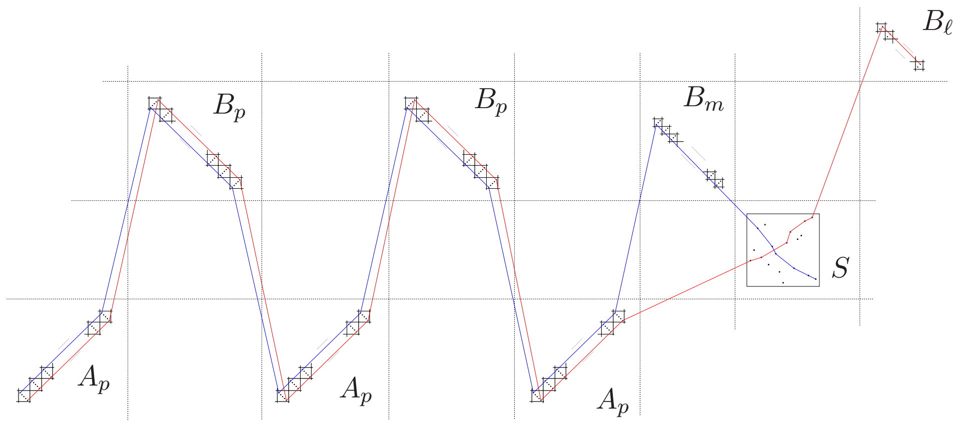

Figure 3: Arrangement of A and B blocks for $k = 5$ ($\ell = 2$, $m = 3$). The relative vertical position of blocks in one row is of no significance to our construction since A and B blocks are vertically separated.

We prove that there is a solution with $p = \ell + m$ lines to this instance of k-MODAL if and only if there is a partition of S into m decreasing and ℓ increasing lines. So let a k-modal cover for the constructed instance be given. By the claim all p lines cover points in all A_p's and B_p's. At least m of these lines also cover points in B_m. These lines must have at least k internal extrema to the left of S. Therefore only the remaining decreasing part can cover points in S. None of these lines can be extended to cover points in B_ℓ. Consequently, B_ℓ must be covered by ℓ lines not covering points in B_m and these lines are increasing in S. Altogether there are m lines with decreasing part in S and ℓ lines with increasing parts in S.

On the other hand a given solution of the instance S of MONOTONE with m decreasing and ℓ increasing lines can be extended to a solution of k-MODAL in the obvious way where additional increasing and decreasing parts of the lines cover components A_p, B_p, B_m, and B_ℓ as described in the proof of the claim.

The constructed instance of k-MODAL has the property that in any cover with p lines these lines are upper k-modal, when k is odd, and lower k-modal, when k is even. Therefore, this instance can be interpreted as an instance of UPPER k-MODAL, possibly after vertical flipping.

Finally, UPPER UNIMODAL reduces to UNIMODEC by adding a component B_p to the left and below a given instance S of the latter problem, see Figure 4. To cover B_p at least p decreasing parts are necessary as was shown before. Therefore a solution of UNIMODEC with p lines exists if and only if a solution of UPPER UNIMODAL with p lines exists. □

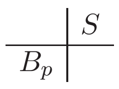

Figure 4: Arrangement of blocks B_p and S used to prove $\mathcal{NP}$-hardness of UNIMODEC.

4 Optimal Partitions: Mixed Integer Programs

4.1 Set Partitioning Formulation

A most natural formulation of our problems is as a set partitioning program. We use a binary variable λ_σ for each feasible (monotone, k-modal, ...) subsequence $\sigma \subseteq S$. Every element of S has to appear exactly once in some selected subsequence:

$$\begin{aligned}
\min \quad & \sum_{\sigma} \lambda_\sigma & & & (1)\\
\text{s.t.} \quad & \sum_{\sigma : i \in \sigma} \lambda_\sigma &= 1 & \quad \text{for all } i & (2)\\
& \lambda_\sigma &\in \{0,1\} & \quad \text{for all feasible } \sigma & (3)
\end{aligned}$$

This concise integer program may contain an exponential number of variables and one would solve it by branch-and-price [12]. We postpone this discussion and present first its *compact* counterpart from which the set partitioning model follows by a Dantzig-Wolfe decomposition.

4.2 Network Flow and Network Design Models

We will see that a minimum cost flow computation solves problem DECREASING. By means of additional binary variables, the flow model extends to a network design problem which solves k-MODAL in general. We describe the respective directed graphs from which our mixed integer programs (MIPs) immediately follow.

Given a sequence S we start with its associated (directed) permutation graph $G = (S, E)$. We split all vertices $s_i \in S$, and obtain a new arc e_i for each element s_i. Arcs previously entering and leaving s_i now enter and leave the tail and the head of e_i, respectively. We add to a source s and a sink t, arcs from s to the tail of e_i, and arcs from the head of e_i to t.

Arcs leaving the source have unit cost, all other arc have cost zero. We introduce arc flow variables $x(e) \geq 0$ for arc e and add lower bounds $x(e_i) \geq y(e_i) := 1$ on all arcs e_i. Since this augmented network component is acyclic, every s-t-flow decomposes into a collection of s-t-paths, each of which represents a decreasing subsequence in S. The cost structure and the lower bounds $y(e_i)$ ensure that a minimum cost s-t-flow represents a minimum cover with the flow value equal to the cover size. This is the basis for our mixed integer programs.

Starting from $\bar{G}$ rather than from G we obtain an augmented network component for representing increasing subsequences, see Figure 5.

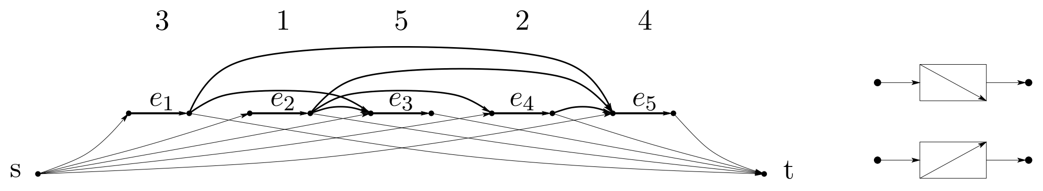

Figure 5: Acyclic augmented network component on the left: An s-t-path represents an increasing subsequence in $S = [3, 1, 5, 2, 4]$. Depicted on the right is our symbolic shorthand notation for decreasing and increasing network components, repectively.

If we concatenate several copies $N^1, N^2, \ldots$ of the above two network components, we can represent more general sequences. Two consecutive network components N^j and N^{j+1} are connected with directed arcs from the head of e_i^j to the head of e_i^{j+1}, where e_i^j denotes the arc representing element s_i in N^j. The lower bounds $y(e_i^j)$ on arcs e_i^j are now binary variables satisfying

$$\sum_j y(e_i^j) = 1, \text{ for all } i. \tag{4}$$

Again, this chain of network components is augmented by a source s and a sink t. Only the first and the last network component are connected to s and t, respectively. Figure 6 shows the construction for UPPER UNIMODAL. If necessary, the arcs connecting two network components can be modified in such a way that every s-t-path has exactly one internal extremum.

One can interpret a unimodec sequence as first traversing a decreasing subsequence in *reverse* order followed by an upper unimodal subsequence. Therefore, it is convenient to have one more network component which is the *decreasing* component with all arcs reversed, see Figure 7.

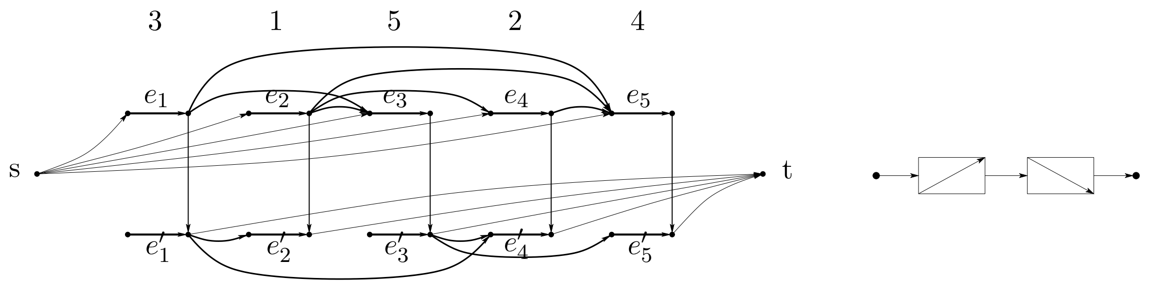

Figure 6: Augmented network with two acyclic network components and its shorthand notation. An s-t-path represents an upper unimodular subsequence in $S = [3, 1, 5, 2, 4]$.

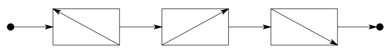

Figure 7: Augmented network with three acyclic network components in shorthand notation. An s-t-path represents a unimodec subsequence.

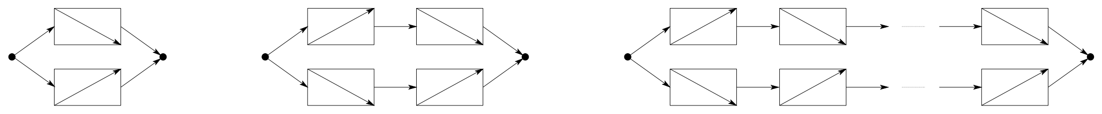

Figure 8: Augmented networks in shorthand notation. An s-t-path represents a monotone, unimodal, and k-modal subsequence, respectively.

In the general case we use two parallel chains of network components, again augmented by a single source and a single sink. These are connected to both chains, cf. Figure 8. Then, a minimum cost s-t-flow (in fact, a collection of disjoint s-t-paths) corresponds to minimum monotone, unimodal, and k-modal partitions of S, respectively. In the latter case each chain has $k+1$ network components.

4.3 Comparison of the Linear Relaxations

For solving the linear relaxation of (1)–(3) by column generation [12] one starts with a restricted set of feasible subsequences and dynamically generates more as needed. Associated to every constraint (2), i.e., to every element $s_i \in S$ is a dual variable $\pi(s_i) \in \mathbb{R}$. From an optimal dual solution we check the reduced cost $1 - \sum_{i \in \sigma} \pi(s_i)$ of variables λ_σ which should be all non-negative. This is done via the following pricing subproblem

$$\min\{1 - \sum_{i \in \sigma} \pi(s_i) \mid \sigma \text{ is a feasible subsequence}\}. \tag{5}$$

This problem can be solved efficiently by a shortest s-t-path computation in a modified version of the acyclic networks as described in Section 4.2: Capacities are removed, and we introduce costs $-\pi(e_i)$ on arcs e_i^j. A consequence of this is the following.

Lemma 2 *The LP relaxations of the network design model and the LP relaxation of the corresponding set partitioning model have the same optimal objective function value.*

Proof. The set partitioning model is a Dantzig-Wolfe decomposition of the network design model; an optimal solution to the pricing problem (5) is always integer. Under these circumstances the bound of the linear relaxations are known to coincide [10]. □

In principle the set partitioning model surpasses the network design models in terms of modeling flexibility. This advantage comes at the cost of extra implementation efforts.

Remark. The interpretation as hypergraph coloring mentioned at the end of Section 2 leads to a MIP with partitioning and packing constraints. However, we found that its LP relaxation gives very weak bounds, and we omit a further discussion.

5 Approximation Algorithms

Fomin, Kratsch, and Novelli [9] develop a 1.71-approximation algorithm for finding a minimum partition of a partially ordered set into chains and antichains. In particular, this algorithm gives a 1.71-approximation for MONOTONE. From this we easily derive a $1.71(k+1)$-approximation algorithm for k-MODAL as follows.

Lemma 3 *An α-approximate solution for* MONOTONE *is a $(k+1)\alpha$-approximate solution for k-*MODAL*. An α-approximate solution for k-*MODAL *can be converted to a $(k+1)\alpha$-approximate solution for* MONOTONE*.*

Proof. Denote by z^{α}_{mon} and by z^{α}_{k} the size of an α-approximate partition for MONOTONE and for k-MODAL, respectively. Since any k-modal sequence can be split into at most $k+1$ monotone subsequences, the optimal partition sizes z_{mon} and z_k relate as $z_{\text{mon}} \leq (k+1) \cdot z_k$. This gives

$$z^{\alpha}_{\text{mon}} \leq \alpha \cdot z_{\text{mon}} \leq (k+1) \cdot \alpha \cdot z_k,$$

which proves the first part. Any monotone sequence is k-modal, and therefore $z_k \leq z_{\text{mon}}$. Together with the above mentioned splitting of a k-modal sequence we immediately obtain

$$(k+1) \cdot z^{\alpha}_{k} \leq (k+1) \cdot \alpha \cdot z_k \leq (k+1) \cdot \alpha \cdot z_{\text{mon}},$$

which proves the second part. □

It is an open question whether there exists a polynomial time approximation scheme (PTAS) for MONOTONE, let alone k-MODAL. We see no way of obtaining stronger approximation results from the elegant algorithm in [9] since its analysis is tight for MONOTONE, and the combinatorial argument used for its correctness does not generalize to k-MODAL. However, we will describe an LP rounding approach which applies to our network design model and yields improved approximation factors.

We consider a particular integer programming problem

$$z^* := c^T x^* := \min\{c^T x \mid x \in P,\ y \in Y,\ x \geq y,\ x \text{ integer},\ y \text{ binary}\} \tag{IP}$$

with optimal solution x^*, y^*. Let $Y \subseteq [0,1]^p$ be defined by a set of generalized bound constraints: The variables $y_j, j \in I_q$, appear in the q-th generalized bound constraint

$$\sum_{j \in I_q} y_j = 1\ ,$$

all other y_j variables are fixed with $L := \{j \mid y_j = 0\}$, $U := \{j \mid y_j = 1\}$. The sets L, U, and the union $I = \bigcup_q I_q$ are pairwise disjoint. Let ρ denote the maximum number of variables in a generalized bound constraint, i.e., $\rho = \max_q |I_q|$, We require $\rho \geq 2$. The polyhedral set $P \subseteq \mathbb{R}^p$ has the important property $\rho \cdot P \subseteq P$.

In particular, our network design models are of this form since any multiple (larger than one) of a feasible flow in a network without upper bounds is again a feasible flow. Formally, the network design models only use y-variables on a subset of the arcs, but we can easily add zero valued variables for all other arcs. The model contains n generalized bound constraints (4), one for each element of the permutation, and ρ is the number of its network components.

Algorithm: LP Rounding for IP

- *Compute an optimal solution* x^{LP}, y^{LP} *of the linear relaxation* LP *of* IP.
- *Choose a set of representatives* $R \subseteq \{j \in I \mid y_j^{LP} \geq \frac{1}{\rho}\}$ *with* $|R \cap I_q| = 1$ *for all* q. *Rounding up according to* R *defines the binary vector* $\hat{y}$ *with* $\hat{y}_j = 1 \iff j \in R \cup U$.
- *Compute an optimal solution* $\bar{x}$ *of the linear programming relaxation* LP2 *of the integer program* IP2 $c^T\hat{x} := \min\{c^T x \mid x \in P,\ x \geq \hat{y},\ x \text{ integer}\}$.

The choice of ρ implies the existence of a suitable set R of representatives. Moreover, $\rho \cdot x^{LP} \geq \hat{y} \in Y$ and $\rho \cdot x^{LP} \in P$, i.e., $\rho \cdot x_{LP}$ is feasible for LP2. In particular, the algorithm can be applied to our network design model. Since for fixed lower bound vector $\hat{y}$ the network design model reduces to a network flow problem, the corresponding LP relaxation has an integer optimum.

Theorem 4 *Let* $\hat{y}$ *denote the rounded vector obtained from* LP ROUNDING *and let* ρ *be the maximum number of variables in a generalized bound constraint* (4). *If the LP relaxation* LP2 *has an integer optimum* $\bar{x}$, *then* $\bar{x}, \hat{y}$ *is a* ρ*-approximation of* IP.

Proof. If $\bar{x}$ is integer, then $\bar{x}, \hat{y}$ are feasible for IP. As remarked above, $\rho \cdot x^{LP}$ is feasible for LP2 and $\hat{y} \in Y$. Therefore,

$$c^T x^{LP} \leq c^T x^* \leq c^T \bar{x} \leq c^T(\rho \cdot x^{LP}),$$

proving that $\bar{x}, \hat{y}$ is a ρ-approximation of IP. □

Corollary 5 LP ROUNDING *yields*

a 2-approximation for MONOTONE, UPPER UNIMODAL, *and* LOWER UNIMODAL,

a 3-approximation for UNIMODEC,

a $(k+1)$*-approximation for* UPPER k-MODAL *and* LOWER k-MODAL, *and*

a $2(k+1)$*-approximation for* k-MODAL.

In general, the approximation factor is determined by the number of monotone pieces in the considered subsequences, with an additional factor of 2 if we do not fix the monotonicity of the first part. It would be interesting to see whether one can eliminate this factor 2.

We note that the integrality gap of our MIP model for MONOTONE and UPPER UNIMODAL is at least $\frac{3}{2}$ as is shown e.g., by the sequence $[6, 2, 1, 4, 3, 5]$. For both problems, the optimal LP value is 2.0; the optimal integer objective is 3.0.

6 Online Algorithms

Not least in view of our practical motivation it is natural to ask for the online version of our problems in which the permutation becomes known sequentially. We have to assign elements to subsequences without looking at the remaining elements of the permutation, see e.g., [8] for background on online algorithms. For INCREASING and DECREASING the (optimal) greedy algorithm is in fact an online algorithm [7]. For MONOTONE the situation is much worse.

Theorem 6 *There is no $o(\log n)$ competitive online algorithm for* MONOTONE*.*

Proof. Consider any online algorithm $\mathcal{A}$ for MONOTONE. Depending on the decisions made by $\mathcal{A}$ we construct a sequence $S = [1, \ldots, n]$ with $n = 2^h - 1$ elements, for some integer h. Initially, we have the range of numbers $a = 1$ to $b = n$. The first element of S is $(a + b)/2 = 2^{h-1}$, and $\mathcal{A}$ has to open a subsequence. We arbitrarily set $a = 2^{h-1} + 1$ or $b = 2^{h-1} - 1$, and serve $(a+b)/2$ as second element. In general, $\mathcal{A}$ has three options (of which in fact only two are actually possible). We describe this for the second iteration. First note that a decision to append to an existing subsequence decides upon whether that sequence is increasing or decreasing.

If $\mathcal{A}$ decides to append in an increasing way we set $b = 2^{h-1} - 1$. If $\mathcal{A}$ decides to append in a decreasing way we set $a = 2^{h-1} + 1$. In either case we have a connected range of $2^{h-1} - 1$ numbers none of which can be appended to an already existing subsequence of at least two elements. If a new subsequence is opened we adapt either a or b arbitrarily as above. We iterate with the new values of a and b, and it follows by induction that $\mathcal{A}$ generates at least $h/2$ subsequences for the first h elements of S (since each subsequence contains at most two elements).

Let $a_1, \ldots, a_h$ and $b_1, \ldots, b_h$ be the values of a and b throughout the first h iterations described above. The ith element of S is either $a_{i+1} - 1$ or $b_{i+1} + 1$. Since the sequences $a_1, \ldots, a_h$ and $b_1, \ldots, b_h$ are increasing and decreasing, respectively, the first h elements of S can be covered by an increasing subsequence of $a_1 - 1, \ldots, a_h - 1$ and by a decreasing subsequence of $b_1 + 1, \ldots, b_h + 1$.

If the remaining elements of S are arranged in an increasing way the optimal solution contains 3 subsequences. However, the solution determined by $\mathcal{A}$ contains at least $h/2$ subsequences. Therefore, $\mathcal{A}$ is $\log_2(n + 1)/6$-competitive at best. □

Restatement of Theorem 6 *The problem of cocoloring a permutation graph with n vertices does not allow an online algorithm with competitive ratio better than $\Omega(\log n)$.*

Since the publication of the conference version [16] of this paper, Demange and Leroy-Beaulieu [6] adapted our proof to the situation where the range of numbers in the permutation is not known in advance. For this case, they strengthen the lower bound to $n/4 + 1/2$.

We next discuss the performance of two online algorithms for MONOTONE, UPPER UNIMODAL, and LOWER UNIMODAL, both of which are reminiscent of simple bin packing online algorithms.

Online Algorithm Next Fit

Open an empty subsequence.

Add arriving elements s to one and the same open subsequence as long as feasibility of the generated subsequence is not violated. Otherwise close the currently open subsequence and open a new subsequence with s.

Lemma 7 NEXT FIT *is exactly $n/4$-competitive for* MONOTONE *and* UPPER UNIMODAL.

Proof. Any two elements of the input sequence S form a monotone (unimodal) subsequence. Thus, we have $n/2$ as a trivial upper bound for the number of subsequences determined by NEXT FIT. If S itself is monotone (unimodal) the algorithm finds the optimal solution. Otherwise, the optimal solution consists of at least two subsequences giving a competitive ratio of $n/4$. To see that this bound is tight consider the sequence $S = [n,\ 1,\ n-1,\ 2, \dots]$. For MONOTONE and UPPER UNIMODAL NEXT FIT will determine a solution consisting of $n/2$ subsequences with two elements each. The optimal solution consists of two sequences in both cases. Therefore, NEXT FIT is exactly $n/4$-competitive. □

Next we make use of the fact that we know the set of pending elements, which are the numbers in $1, \dots, n$ we have not yet seen in the input sequence.

Online Algorithm Best Fit

Start with two subsequences containing only one dummy element 0 and $n+1$, respectively.

Add arriving elements s to one subsequence P of the current solution for which $P \cup s$ stays feasible and for which the number of pending elements between s and the last element of P is minimal. If P is a dummy subsequence then P is redefined by $P := \{s\}$ and a new subsequence having the same dummy element as P is added to the solution unless it consists of $n/2$ subsequences. Resolve ties arbitrarily, preferring subsequences without dummy elements.

Note that in each iteration at least one of the at most $n/2$ subsequences contains one (dummy) element only. Therefore in each iteration an appropriate P exists. Unfortunately, for MONOTONE the extra effort made in BEST FIT in comparison to NEXT FIT does not improve the competitive ratio.

Lemma 8 BEST FIT *is exactly $n/4$-competitive for* MONOTONE.

Proof. If the input permutation is itself feasible, BEST FIT is optimal. Otherwise, by definition, it generates at most $n/2$ feasible subsequences and is at least $n/4$-competitive. To see that the upper bound is tight, we consider the permutation $S = [2, 1, 4, 3, \dots, 2k, 2k-1, \dots]$. The algorithm generates decreasing two-element subsequences $[2k, 2k-1]$ for all k, but the optimal partition contains only the two increasing subsequences $[2, 4, \dots]$, $[1, 3, \dots]$. □

The sequence defined in the last proof also gives a tight example for the $n/4$-competitiveness of the analogue of the bin packing algorithm FIRST FIT.

7 Computational Results

We generated random permutations (with the standard shuffling algorithm as described by Knuth) with 30, 60, 90, and 120 elements; 90–120 elements appear to be a reasonable size for rail car shunting. For each size we generated 100 instances. All experiments were conducted on a Linux PC (AMD Athlon MP 2800+) with 2.1 GHz and 3.5 GByte main memory. For solving linear and mixed integer programs we use the standard solver CPLEX 9.0 with default parameter settings. On every computation we impose a time limit of 900 CPU seconds.

Table 1 summarizes our results. Allowing ourselves a conservative judgment only, we see that we can optimally solve instances of the given size with the network design model within an acceptable time frame. For the set cover formulation we generated the *entire* integer program. It shows an exceptionally bad performance due to the fact that the lower bound only very slowly improves during the branch-and-bound process. To be fair at this point we remark that a true branch-and-price algorithm would probably outmatch the network design MIP in terms of computation time on larger instances.

The greedy algorithm, which iteratively extracts longest feasible subsequences, runs in a split second and yields very good solutions on the average as well as in the (empirically) worst case. The quality of solutions obtained with the LP Rounding is significantly better than the theoretically guaranteed approximation factors suggest. Next Fit empirically performs as poorly as predicted by competitive analysis, whereas Best Fit gives fair results on average.

8 Conclusions

We studied minimum partitioning of permutations into subsequences with certain monotonicity properties. Particularly interesting subsequences are increasing, monotone, upper unimodal, k-modal, and unimodec. Problems of this kind arise as subproblems in railroad logistics, see e.g., [2, 7, 18]. Theoretical hardness legitimates studying and applying computationally expensive approaches like solving (probably large scale) mixed integer programs. These also yield (small) constant factor approximation algorithms via LP rounding.

Several extensions can be incorporated in our models, e.g., a bounded track length, i.e., the length of a subsequence must not exceed a given number of elements. Then, solutions of our network flow based models become resource constrained shortest paths, which are $\mathcal{NP}$-hard to compute. The set partitioning model is most flexible in terms of extendibility. It is able to capture more "dirty" side constraints which do not fit into the context of this paper.

Among the remaining interesting open questions are:

- What is the exact approximability status of monotone and k-modal, in particular, does there exist a PTAS? Can our LP techniques lead to an improvement of the 1.71 approximation for monotone? Such a result would be quite fascinating since the algorithm in [9] already elegantly exploits the combinatorial structure of the problem.
- Considering the lower bound on the competitive ratio given in Theorem 6 one would be interested in an online algorithm matching this bound. Which competitive ratio is possible when look-ahead is allowed? In [6] some preliminary results are obtained.

n		Greedy	LP Rnd.	LP	Flow	(1)–(3)	Next	Best
monotone								
30	avg	1.066	1.060	.810	.092	.123	2.259	1.350
	worst	1.200	1.400	.726	.240	1.120	2.800	2.000
60	avg	1.127	1.111	.827	1.841	139.335	3.214	1.484
	worst	1.285	1.625	.742	35.340	900.060	3.857	1.875
90	avg	1.123	1.116	.821	19.953	869.234	3.853	1.581
	worst	1.222	1.333	.770	158.450	900.630	4.444	2.111
120	avg	1.151	1.133	.824	218.340	†	4.468	1.653
	worst	1.300	1.454	.769	900.040	†	5.000	2.100
upper unimodal								
30	avg	1.098	1.051	.804	.033	.386		
	worst	1.250	1.500	.678	.100	5.330		
60	avg	1.142	1.076	.803	.560	455.885		
	worst	1.400	1.333	.714	14.600	900.350		
90	avg	1.161	1.081	.812	19.009	†		
	worst	1.285	1.428	.717	407.680	†		
120	avg	1.193	1.090	.811	255.213	†		
	worst	1.375	1.250	.727	900.080	†		

Table 1: Average and worst case results for MONOTONE and UPPER UNIMODAL. The columns list (in that order): $|S| = n$; the quality (as a factor as compared to optimal = 1.000) for the greedy algorithm, LP ROUNDING, and the LP relaxation; CPU seconds for optimally solving the network design MIP and the set partitioning model; and the quality (as a factor) of the online algorithms NEXT FIT and BEST FIT. The † sign indicates that the time limit of 900 CPU seconds was almost always exceeded.

- The crucial property we use in the construction of the graphs underlying our MIP models, and which ensures that paths correspond to increasing or decreasing subsequences, is the transitivity of the ordering of elements. We would have liked to generalize our positive results for permutations to partially ordered sets (corresponding to comparability graphs). However, in general, the transitivity is lost for the complement of a comparability graph. Is there a network flow based model similar to ours which allows LP rounding, thus yielding a constant factor approximation?

Acknowledgment. We would like to thank Laura Heinrich-Litan for pointing us to the literature on cocoloring.

References

[1] R. Bar-Yehuda and S. Fogel. Partitioning a sequence into few monotone subsequences. *Acta Inform.*, 35(5):421–440, 1998.

[2] U. Blasum, M.R. Bussieck, W. Hochstättler, C. Moll, H.-H. Scheel, and T. Winter. Scheduling trams in the morning. *Math. Methods Oper. Res.*, 49(1):137–148, 1999.

[3] M. Bóna. A survey of stack-sorting disciplines. *Electronic Journal of Combinatorics*, 9(2):Article A1, 2002/2003.

[4] A. Brandstädt and D. Kratsch. On partitions of permutations into increasing and decreasing subsequences. *Elektron. Informationsverarb. Kybernet.*, 22(5/6):263–273, 1986.

[5] F.R.K. Chung. On unimodal subsequences. *J. Combin. Theory Ser. A*, 29:267–279, 1980.

[6] M. Demange and B. Leroy-Beaulieu. Online coloring of comparability graphs: Some results. ORWP 07/01, Institut de Mathématiques, EPFL Lausanne, 2007.

[7] G. Di Stefano and M.L. Koči. A graph theoretical approach to the shunting problem. *Electr. Notes Theor. Comput. Sci.*, 92:16–33, 2004.

[8] A. Fiat and G.J. Woeginger. *Online Algorithms—The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998.

[9] F.V. Fomin, D. Kratsch, and J.-C. Novelli. Approximating minimum cocolourings. *Inform. Process. Lett.*, 84(5):285–290, 2002.

[10] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Math. Programming Stud.*, 2:82–114, 1974.

[11] D.E. Knuth. *The Art of Computer Programming*, volume 1: Fundamental Algorithms. Addison-Wesley, third edition, 1997.

[12] M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.

[13] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, Berlin, 2003.

[14] J.M. Steele. Long unimodal subsequences: A problem of F.R.K. Chung. *Discrete Math.*, 33:223–225, 1981.

[15] J.M. Steele. Variations on the monotone subsequence theme of Erdős and Szekeres. In D. Aldous, P. Diaconis, J. Spencer, and J.M. Steele, editors, *Discrete Probability and Algorithms*, pages 111–131. Springer-Verlag, New-York, 1995.

[16] G. Di Stefano, S. Krause, M.E. Lübbecke, and U.T. Zimmermann. On minimum k-modal partitions of permutations. In J.R.Correa, A.Hevia, and M.Kiwi, editors, *Latin American Theoretical Informatics (LATIN 2006)*, volume 3887 of *LNCS*, pages 374–385. Springer-Verlag, Berlin, 2006.

[17] K. Wagner. Monotonic coverings of finite sets. *Elektron. Informationsverarb. Kybernet.*, 20(12):633–639, 1984.

[18] T. Winter and U.T. Zimmermann. Real-time dispatch of trams in storage yards. *Ann. Oper. Res.*, 96:287–315, 2000.