On Minimum k-Modal Partitions of Permutations

Gabriele Di Stefano[1], Stefan Krause[2],
Marco E. Lübbecke[3], and Uwe T. Zimmermann[2]

[1] Dipartimento di Ingegneria Elettrica, Universita dell'Aquila,
Monteluco di Roio, I-67040, L'Aquila
gabriele@ing.univaq.it
[2] Institut für Mathematische Optimierung, Technische Universität Braunschweig,
Pockelsstraße 14, D-38106, Braunschweig
{stefan.krause, u.zimmermann}@tu-bs.de
[3] Technische Universität Berlin, Institut für Mathematik, Sekr. MA 6-1,
Straße des 17. Juni 136, D-10623, Berlin
m.luebbecke@math.tu-berlin.de

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, that is, subsequences having at most k internal extrema. 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. From these models we derive an LP rounding algorithm which is a 2-approximation for minimum monotone partitions and a $(k+1)$-approximation for minimum (upper) k-modal partitions in general; this is the first approximation algorithm for this problem. In computational experiments we see that the rounding algorithm performs even better in practice. For the associated 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 findings immediately apply to online cocoloring of permutation graphs; they are the first results concerning online algorithms for this graph theoretical interpretation.

Keywords: Mixed integer program; approximation algorithm; LP rounding; online algorithm; $\mathcal{NP}$-hardness; monotone sequence; k-modal sequence; 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 (not necessarily consecutive elements in S) with particular monotony properties. Research in this direction dates back to the famous

J.R. Correa, A. Hevia, and M. Kiwi (Eds.): LATIN 2006, LNCS 3887, pp. 374–385, 2006.

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 [11]. Greedily extracting longest monotone subsequences in an iterative way yields a partition into at most $2\lfloor\sqrt{n}\rfloor$ monotone subsequences in $O(n^{1.5})$, see [2]. However, finding a minimum size partition into monotone subsequences is $\mathcal{NP}$-hard [12]. For fixed k and l (not part of the input), a partition into exactly k increasing and l decreasing subsequences can be computed in $O(n^{k+l})$, see [4]. A minimum monotone partition can be approximated within a factor of 1.71 in $O(n^{2.5})$, see [8].

A natural generalization asks for partitions into k-modal subsequences; that are sequences having 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 $\lceil\sqrt{3(n-1/4)}-1/2\rceil$. Chung 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. She refers to a simple proof obtained by Steele and Chvátal (among others, unpublished, but see [6] for a proof). For the guaranteed length of contained k-modal subsequences, Chung [5] gives the upper bound $\sqrt{(2k+1)n}$. Steele [10] 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 [6]. Even though a more general discussion is possible, we only consider k-modal sequences where the first internal extremum is a maximum, i.e., a generalization of *upper unimodal* sequences.

Our Contribution. We show that partitioning a permutation into a minimum number of k-modal (in particular: unimodal) subsequences is $\mathcal{NP}$-hard. 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. In fact, an easy observation allows us to derive a $1.71(k+1)$-approximation first. Not only because of the practical motivation described below, we are interested in actually computing optimum partitions. To this end we introduce mixed integer programming (MIP) formulations which can be easily extended to respect a variety of practical side constraints. We further give the first negative and (weakly) positive results concerning online algorithms for minimum monotone partitions. These findings immediately apply to cocoloring of permutation graphs, for which no online algorithms were known either.

Motivation and Application. 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 [3, 6, 13]. Our choice is limited by the fixed number of available tracks and by the mode 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. Our task is to choose a track for each unit, and the goal is to use as few tracks as possible.

The relation to our problems is that units on each track represent a subsequence of the incoming sequence of units. The different entry/exit combinations lead in particular to monotone and unimodal subsequences [6]. This relation may seem to be artificial, and we concede that the purpose of this paper primarily is to study the more theoretical background; however, the MIP models we propose can be tailored to fully capture the "real-world" situation, see our conclusions.

2 Preliminaries

Our results hold for any sequence $S = [s_1, s_2, \ldots, s_n]$ of n distinct reals, but we assume S to be a permutation of the first n integers. 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 is *k-modal* if is has at most k internal extrema; in particular in this paper, usually the first extremum should be a maximum, i.e., the first sequence is increasing (then we speak of *upper k-modal*). Particularly well known is the case of 1-modal (i.e., *unimodal*) sequences.

We use an intuitive set notation and language to work with sequences; e.g., when referring to all the elements contained in two sequences we speak of their union. 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. The type of subsequences allowed in P gives the name of the resulting minimization problem, that is, (monotone), (unimodal), or (upper k-modal). A *cover* of S is a collection of subsequences, the union of which contains each element in S *at least* once. Eliminating multiply covered elements, 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 [12].

Related Concepts. The easiest of our partitions are well studied in a graph theoretical context. The *permutation graph* $G = (S, E)$ associated with a permutation S has an edge (s_i, s_j) 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 in G.

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 can be given in $O(n \log n)$ (see e.g., [9]). *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 problem (monotone) we compute an optimal cocoloring of a permutation graph. Problem (k-modal) can be interpreted as a particular coloring problem on hypergraphs [6].

3 Complexity

In this extended abstract we present all statements for (upper k-modal), but for the sake of brevity proofs are given only for (upper unimodal). Restricting attention to this case essentially captures the necessary ideas needed for the generalization; all details are in the full paper.

Theorem 1. *Problem (k-modal) is strongly $\mathcal{NP}$-hard.*

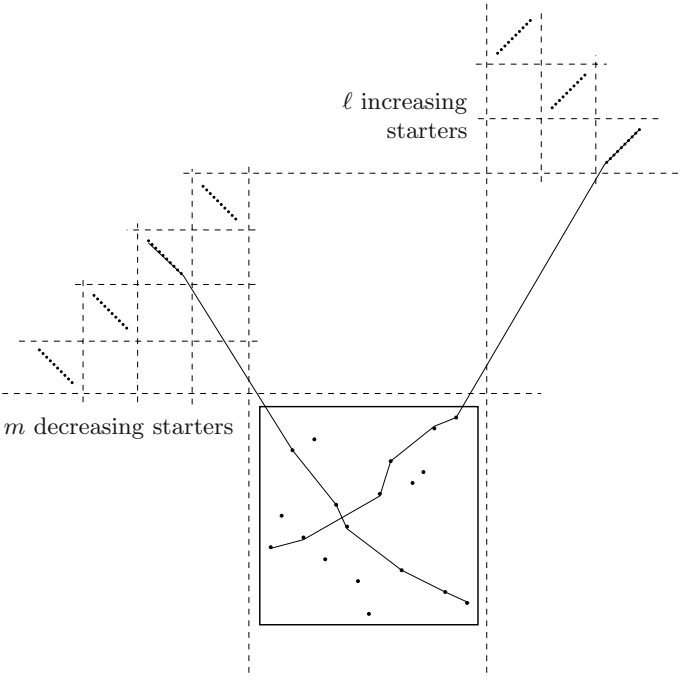

Fig. 1. Point map of the construction used in the proof of Theorem 1; $m = 4$, $\ell = 3$

Proof. Clearly, (upper unimodal) is in $\mathcal{NP}$; we will drop the attribute *upper* in the remainder. We use a reduction from (monotone) which is strongly $\mathcal{NP}$-hard [12]. In fact, one can solve (monotone) by solving a series of p restricted problems of partitioning S into at most $\ell = 1, \ldots, p$ increasing and at most $m = p - \ell$ decreasing subsequences. We reduce to this restricted version.

We represent elements and subsequences as points and lines. Having arranged the points corresponding to the elements of a given permutation S, we construct an extended arrangement of points which can be covered by p unimodal lines if and only if the original set of points can be covered by ℓ increasing and m decreasing lines. In fact, there always is an optimal solution to our construction which uses monotone lines only. We briefly use the notion *bounding rectangle*

for an axis-parallel rectangle containing the points corresponding to the given permutation, and no other points of our construction.

Above and to the left of the bounding rectangle we introduce m sets of points called the *decreasing starters*. Each of them contains $2p$ points which form a strictly decreasing line. The decreasing starters themselves are arranged in a chain going upwards and rightwards such that their respective ranges of x- and y-coordinates are disjoint. Above them and to the right of the bounding rectangle we introduce ℓ sets of points called the *increasing starters*. Each of them contains $2p$ points which form a strictly increasing line. The increasing starters themselves are arranged in a chain going downwards and rightwards such that their respective ranges of x- and y-coordinates are disjoint. Since $p \leq n$, this construction is polynomial.

If there is a cover of the given permutation's points with m decreasing and ℓ increasing lines, then these lines can be extended to $p = \ell + m$ unimodal, in fact monotone, lines as indicated in the figure such that all starters are covered.

On the other hand, assume that we are given a cover of p unimodal lines for the extended point set. The decreasing starters have to be covered by m distinct decreasing lines, and the increasing starters have to be covered by ℓ distinct increasing lines. Actually, since the increasing starters are above the decreasing starters, the arrangement enforces that all of these $p = \ell + m$ lines have to be distinct. These can pass through the bounding rectangle, and we obtain the claimed solution to the original problem. □

4 Exact Approaches: Mixed Integer Programs

In this section we develop mixed integer programs (MIPs) for computing optimal partitions (see e.g., [9] for background on linear and integer programming). We first solve the problem of partitioning into increasing subsequences via a linear program (LP) which in fact is a minimum cost flow model. We embark on this expensive approach because we can extend this model to monotone and k-modal covers by means of additional binary variables. We describe the construction of the respective directed graphs from which the MIP models can be easily derived. When we speak of inserting a directed edge $e = (i, j)$, we imply inserting the *tail* node i of e, and the *head* node j of e, if they are not already present. Unless otherwise stated, there are no capacity bounds on edges except non-negativity. We denote the source of the respective graph by s and denote the sink by t.

A Network Flow Linear Program. We construct a directed graph as follows. Corresponding to element s_i, $i = 1, \ldots, n$, we introduce an edge e_i with a lower capacity bound of 1 and zero cost. We connect the source s to the tail of each e_i with unit cost edges. The head of each e_i is connected to the sink t with zero cost edges. Additionally, we insert a zero cost edge going from the head of e_i to the tail of e_j if and only if $i < j$ and $s_i < s_j$ (that is, we model increasing subsequences; the decreasing case is similar).

We seek a minimum cost flow from s to t. Since our graph is acyclic, an optimal flow can be decomposed into s-t-paths [1]. By construction, each of these paths

uses exactly one edge incident to s, and the objective value is the number of paths. Each path uses a subset of the edges e_i. Our construction ensures that the sequence of the elements s_i corresponding to the edges e_i in each path is an increasing subsequence of S. Since the lower bound on the edges e_i is 1, all these edges must be contained in some s-t-path; the subsequences of S corresponding to the paths form a minimum partition into increasing subsequences.

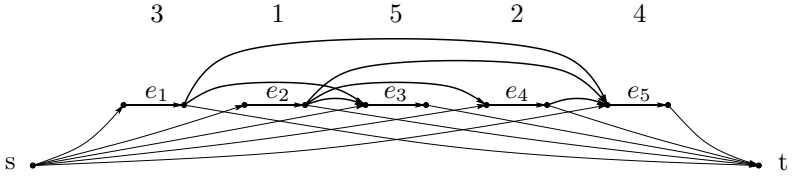

Fig. 2. The graph for the network flow model in the increasing case, $S = [3, 1, 5, 2, 4]$

A Flow Based MIP for Monotone Partitions. One can easily find a minimum monotone cover if we fix for each element whether it occurs in an increasing or in a decreasing subsequence: This results in two independent instances. We use this fact to model the monotone case. We use two complementary copies of the above network flow model, one part corresponding to increasing subsequences, and one complemented part for decreasing subsequences. For each e_i in the increasing part there is a corresponding copy e'_i in the decreasing part. The increasing part remains as before, and in the decreasing part there is an edge going from the head of e'_i to the tail of e'_j if and only if $i < j$ and $s_i > s_j$. The two parts share the source s and the sink t. We introduce binary variables x_i and x'_i and set the lower bound on the edges e_i to x_i and of e'_i to x'_i in the increasing and the decreasing part, respectively, where we require that $x_i + x'_i = 1$.

Again, an optimal flow decomposes into s-t-paths; these correspond to monotone subsequences of S, and the objective function value gives the number of paths. Since exactly one of e_i or e'_i has a lower bound of 1 these subsequences form a minimum monotone cover.

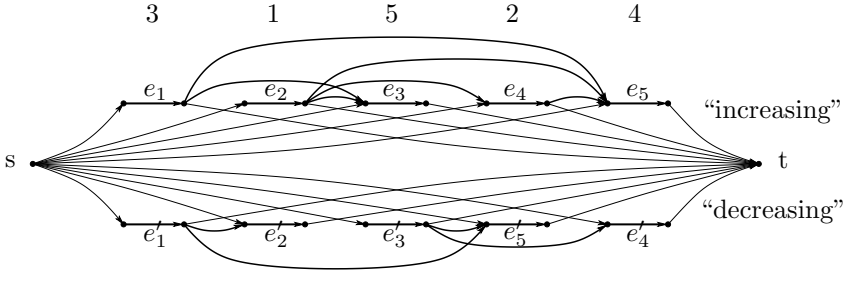

Fig. 3. The graph for the network flow based MIP model in the monotone case, $S = [3, 1, 5, 2, 4]$

A Flow Based MIP for (Upper) Unimodal Partitions. For the unimodal case, we start with the graph constructed for the monotone case. From the increasing part we omit the edges incident to t. From the decreasing part we omit the edges incident to s. For each i we add an edge connecting the head of e_i to the head of e'_i. Again, e_i and e'_i each get a lower bound of x_i and x'_i, respectively, where x_i and x'_i are binary variables with $x_i + x'_i = 1$.

In this graph an s-t-path uses exactly one edge incident to s, at least one edge e_i in the increasing part (corresponding to an increasing subsequence) and possibly some edges e'_i in the decreasing part (corresponding to a decreasing subsequence). Together, a path represents an upper unimodal subsequence. The variables x_i and x'_i control whether s_i occurs in the increasing part of such a sequence (including its maximum) or in its decreasing part. Note that also degenerate cases are considered, that is, monotone sequences are possible parts of a solution. The binary variables ensure that each s_i occurs in at least one unimodal sequence, therefore an optimal solution to this MIP gives a minimum unimodal cover. This construction generalizes to (upper k-modal) via the construction of an extended network of $k + 1$ *layers*.

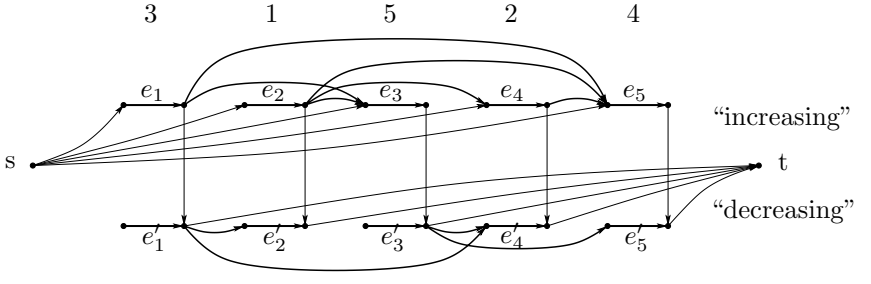

Fig. 4. The graph for the network flow based MIP model in the upper unimodal case, $S = [3, 1, 5, 2, 4]$, "upper unimodal" meaning—as always in this extended abstract—at most one internal maximum

5 Approximation Algorithms

Fomin, Kratsch, and Novelli [8] give a factor 1.71 approximation algorithm for finding a minimum partition of a partially ordered set into chains and antichains. In particular, this is a 1.71 approximation algorithm for the (monotone) problem. It is an open question whether there exists a polynomial time approximation scheme (PTAS). We derive a $1.71(k+1)$-approximation algorithm for (k-modal).

Lemma 1. *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

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

proving the first part of the lemma. 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. □

Using our network flow MIP models from the preceeding section, we are able to improve on this factor. We obtain a $(k+1)$-approximation algorithm for (upper k-modal). We state the result and the proof for (monotone) only.

Algorithm LP Rounding for (monotone)
Solve the LP relaxation of the MIP model for (monotone). For each element $i = 1, \ldots, n$, fix $x_i = 0$ if $x_i < 0.5$, and fix $x_i = 1$ if $x_i \geq 0.5$. Solve the resulting "fixed" LP again, and output the subsequences of S corresponding to the s-t-paths in an optimal solution.

Lemma 2. LP Rounding *is a 2-approximation algorithm for (monotone).*

Proof. For each $i = 1, \ldots, n$, if we fix $x_i = 1$ we increase the lower bound on e_i from at least 0.5 to 1.0. If we fix $x_i = 0$, this implies to fix $x'_i = 1$, and we increase the lower bound on e'_i from at least 0.5 to 1.0. The respective lower bound is at most doubled.

Denote by z the objective function value of an optimal solution x to the linear programming relaxation. Doubling the flow value of every s-t-flow in x gives a feasible solution to the fixed problem with objective function value at most $2z$. This is an upper bound for the optimal flow's objective function value in the fixed problem, yielding the claimed approximation factor.

This result generalizes to (upper k-modal) since we have $k+1$ variables per element, so at least one has fractional value at least $1/(k+1)$. Polynomial time solvability of linear programs follows from the ellipsoid method [9]. □

We note that the integrality gap of our MIP model for (monotone) is at least $\frac{3}{2}$ as is shown e.g., by the sequence $[6, 2, 1, 4, 3, 5]$: The optimal LP value is 2.0, the optimal integral objective is 3.0. From our computational experience we conjecture that the correct gap is smaller than 2, and that the analysis of the performance of LP Rounding can be improved.

6 Online Algorithms

Not only 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., [7] for background on online algorithms. For partitions into increasing subsequences the (optimal) greedy algorithm is in fact an online algorithm [6]. Already for (monotone) the situation is much worse.

Theorem 2. *There is no constant factor competitive online algorithm for (monotone).*

Proof. Consider any online algorithm $\mathcal{A}$. Depending on the decisions made by $\mathcal{A}$ we construct a sequence S with $n = 2^h - 1$ elements. We start with 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. 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 $a_i, \ldots, a_h$ 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 subsequences. Therefore, $\mathcal{A}$ is $\log_2(n + 1)/6$-competitive at best. □

Since we are not aware of any previous results on online algorithms for cocoloring, it is interesting in its own right to restate this result in graph theoretical terms.

Restatement of Theorem 2. *The problem of cocoloring a permutation graph does not allow an online algorithm with constant competitive ratio.*

We next discuss the performance of two online algorithms for (monotone) and (unimodal). Both are reminiscent of simple bin packing online algorithms.

Online algorithm Next Fit
Keep adding elements to one and the same subsequence as long as monotony (unimodularity) is not violated. Then start a new subsequence and leave the previous ones unchanged.

Lemma 3. NEXT FIT *is $n/4$-competitive for (monotone) and (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 $S = [n, 1, n-1, 2, \ldots]$. In the monotone and the unimodal case 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 $1, \ldots, n$ we have not yet seen in the input sequence. Interestingly, this does not help the competitive ratio.

Online algorithm Best Fit
We start with n increasing and n decreasing subsequences with an initial dummy element of 0 and $n+1$, respectively, that will be removed when the respective first element is added. An iteration is as follows. Let s be the current element of the input sequence and let t_i be the last element of the ith subsequence. Select an index i such that s can feasibly be added to the ith subsequence and such that the number of pending elements that are between s and t_i is minimum. Resolve ties arbitrarily but prefer already started subsequences. In the end, throw away all unused subsequences.

Lemma 4. BEST FIT *is $n/4$-competitive for (monotone) and (unimodal).*

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 thus at least $n/4$-competitive. To see that the upper bound is tight, we consider the permutation $S = [2, 1, 4, 3, \ldots, 2k, 2k-1, \ldots]$. The algorithm generates decreasing two-element subsequences $[2k, 2k-1]$ for all k, but the optimal partition contains only the two increasing subsequences $[2, 4, \ldots]$, $[1, 3, \ldots]$. □

7 Conclusions

We studied partitions of permutations into subsequences with particular monotony properties. The theoretical hardness legitimates applying computationally expensive algorithms like solving (probably large scale) mixed integer programs. These, in addition to their practical usefulness, yield (small) constant factor approximation algorithms via LP rounding.

In the full paper we computationally evaluate our proposals for random permutations. As a brief summary at this point, permutations of more than 100 elements can be partitioned optimally within a few seconds or minutes by solving our MIPs. The greedy algorithm, which iteratively extracts a longest subsequence of the requested type, runs in a split second and yields an acceptable

solution quality on the average and also in the (empirical) worst case. The quality of solutions obtained with the LP ROUNDING algorithm significantly stays below the theoretically guaranteed approximation factor. However, the simple NEXT FIT online algorithm also empirically performs as poorly as predicted by the competitive analysis, whereas the BEST FIT online algorithm gives somewhat better results on average, as was to be expected.

There are several extensions motivated from practice which we did not explicitly consider in this more theoretical study, but which can be easily incorporated in our models. One such extension is a bounded track length, that is, subsequences must not contain more than a fixed number of elements. Solutions to our network flow based models become resource constrained shortest paths in this case which may be of independent theoretical interest. In particular, we have developed a set covering model which is most flexible in terms of (practical) extendibility. It is able to capture more "dirty" side constraints which do not directly fit into the context of this extended abstract.

There remain several open questions, spawned by our work:

- 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 over the 1.71 approximation for (monotone)? Such a result would be quite fascinating since the known algorithm [8] already elegantly exploits the combinatorial nature of the problem.
- Considering the competitiveness lower bound of Theorem 2 one would be interested in an online algorithm matching this bound. Which competitiveness ratio is possible when look-ahead is allowed?
- 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, this property 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.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1993.
2. R. Bar-Yehuda and S. Fogel. Partitioning a sequence into few monotone subsequences. *Acta Inform.*, 35(5):421–440, 1998.
3. 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.

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. G. Di Stefano and M.L. Koči. A graph theoretical approach to the shunting problem. *Electr. Notes Theor. Comput. Sci.*, 92:16–33, 2004.
7. A. Fiat and G.J. Woeginger. *Online Algorithms—The State of the Art*, volume 1442 of *Lecture Notes in Computer Science*. Springer, 1998.
8. F.V. Fomin, D. Kratsch, and J.-C. Novelli. Approximating minimum cocolourings. *Inform. Process. Lett.*, 84(5):285–290, 2002.
9. A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin, 2003.
10. J.M. Steele. Long unimodal subsequences: A problem of F.R.K. Chung. *Discrete Math.*, 33:223–225, 1981.
11. 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.
12. K. Wagner. Monotonic coverings of finite sets. *Elektron. Informationsverarb. Kybernet.*, 20(12):633–639, 1984.
13. T. Winter and U.T. Zimmermann. Real-time dispatch of trams in storage yards. *Ann. Oper. Res.*, 96:287–315, 2000.