I N F O R M A T I K

Sum-Multicoloring on Paths

Annamária Kovács

MPI–I–2003–1-015 July 2003

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Stuhlsatzenhausweg 85 66123 Saarbrücken Germany

Author's Address

Annamária Kovács

Max-Planck-Institut für Informatik

Stuhlsatzenhausweg 85

66123 Saarbrücken

panni@mpi-sb.mpg.de

Abstract

The question, whether the *preemptive Sum Multicoloring* (pSMC) problem is hard on paths was raised by Halldórsson et al. in [7]. The pSMC problem is a scheduling problem where the pairwise conflicting jobs are represented by a conflict graph, and the time lengths of jobs by integer weights on the nodes. The goal is to schedule the jobs so that the *sum* of their finishing times is minimized. In the paper we give an $\mathcal{O}(n^3 p)$ time algorithm for the pSMC problem on paths, where n is the number of nodes and p is the largest time length. The result easily carries over to cycles.

Keywords

Preemptive Scheduling, Sum-Multicoloring

1 Introduction

In scheduling problems, one has a set of jobs, with time length for each job and pairwise conflicts between certain jobs. The conflicting jobs cannot be worked on at the same time, due to e.g. some non-sharable resource they use. Real-life situations of this kind can be found in operating systems, in areas like traffic intersection control, frequency assignment for mobile phones, compiler design, VLSI routing etc. (see [6]).

In the mathematical model the jobs are represented as nodes of a simple undirected graph $G = (V, E)$, where two nodes representing conflicting jobs are connected by an edge. The *demand* of $v \in V$ is a positive integer $x(v)$ modeling the number of time units – 'hours' – needed to carry out the job of v. A proper schedule $\Psi : V \to 2^{\mathbb{N}}$ of the jobs is done by assigning a set $\Psi(v)$ of positive integers to each $v \in V$ s.t. $|\Psi(v)| = x(v)$ and the sets assigned to adjacent vertices do not intersect (i.e. they are never scheduled at the same time).

In this way the scheduling problem becomes a graph *coloring* problem if $x(v) = 1$ for each $v \in V$, and graph *multicoloring* problem in the general case. (The name stems from regarding the $\Psi(v)$ as sets of colors. However, later in the paper we continue to view the problem as a scheduling problem and the $\Psi(v)$ as sets of hours as we will use colors for something else.)

A traditional optimization goal is to minimize the overall finishing time, respectively the number of colors used to color all the vertices. Another reasonable goal can be to minimize the average finishing time of the jobs. That is, if $f(v)$ denotes the largest integer assigned to v, we search for a schedule (multicoloring), such that $\sum_{v \in V} f(v)$ is minimum over all proper schedules. The latter is called the *sum multicoloring* (SMC) problem.

In the paper we consider *preemptive* schedulings, where the $\Psi(v)$ are arbitrary sets of positive integers (pSMC problem). There has been much related work done concerning the *non-preemptive* SMC (npSMC) problem, where the assigned $\Psi(v)$ sets must be contiguous, see e.g. [6, 7].

Our result. The question, if the pSMC problem is hard on paths, was raised as an open problem by Halldórsson et al. in [7]. In this paper we provide a pseudo-polynomial algorithm for the problem. Let $G = (V, E)$ be a path, $|V| = n$, and $p = \max_{v \in V} x(v)$. Our algorithm takes $\mathcal{O}(n^3 p)$ time. It is based on a technique that is interesting in its own right. With minor modifications the approach can be applied to the pSMC problem on cycles.

Related work. Here we just mention the most relevant results. For a more comprehensive history of the SMC and related problems see e.g. [2, 7].

The *sum coloring* problem, the special case of SMC with unit time requirements, was first raised by Kubicka in [3], where a polynomial algorithm was given for trees.

The sum coloring problem is NP-hard even on bipartite graphs [1], interval graphs [8], planar graphs [2] and line graphs [5]. These results imply the hardness of the corresponding SMC problems.

The general SMC problem was introduced by Bar-Noy et al. [6]. There a comprehensive study of the approximability of both the pSMC and npSMC was presented on different graph classes.

In [7] two efficient algorithms are provided for the *non-preemptive* (npSMC) problem on trees. They run in $\mathcal{O}(n^2)$ and in $\mathcal{O}(np)$ time respectively. On paths the first one runs in $\mathcal{O}(n \log p / \log\log p)$ time. For the *preemptive* (pSMC) problem on trees, a polynomial time approximation scheme is given.

Marx proved the hardness of the pSMC problem on trees in [4]. He has shown that pSMC is NP-hard even on binary trees, even when p is polynomially bounded. Thus, the SMC problem on trees turned out to be one of the few scheduling-type of problems in which the preemptive version is essentially harder than the non-preemptive version. It is natural to go on asking, on which graph-classes pSMC is efficiently solvable. In [7] the question is posed, whether pSMC is hard on paths. For this problem, an algorithm polynomial in n and p is given in this paper. It can serve as a first step towards characterizing these graph-classes.

Overview. Section 2 describes the basic notation we use and establishes a few elementary facts. In Section 3 we give the ingredients of a pseudo-polynomial algorithm. Section 4 contains the details of an improved algorithm of $\mathcal{O}(n^4 p)$ steps. Unfortunately we could not get rid of the factor p but Section 5 sketches a further improvement in the exponent of n and the possible modifications for the case when the graph is not a path but a cycle.

2 Notation, Definitions and Basic Facts

The nodes in the path are numbered from left to right by $1 \dots n$. If $i < j$, we denote the subpath of starting node i and ending node j, by $[i, j]$. If i is a node, then $x(i) \in \mathbb{N}^+$, is the *demand* of i. Let $p = \max_{1 \leq i \leq n} x(i)$ be the largest demand. $\Psi(i) \subset \mathbb{N}$ is the set of numbers or hours assigned to i in schedule Ψ; $|\Psi(i)| = x(i)$. $f_\Psi(i)$ is the largest number assigned to i. Most of the time we will simply write $f(i)$. We will also use $f(i, j) \stackrel{\text{def}}{=} \max(f(i), f(j))$. We add nodes 0 and $n + 1$ to the path with demands $x(0) = x(n + 1) = 0$.

Definition 1 *We call* Ψ *a* (proper) schedule, *if* $\Psi(i) \cap \Psi(i+1) = \emptyset \quad (1 \le i \le n-1)$. Ψ *is an* optimal schedule, *if* $\sum_{i=1}^{n} f_\Psi(i)$ *is minimum over all schedules.* Ψ *is a* square-optimal schedule, *if it is optimal, and the sum* $\sum_{i=1}^{n} f(i)^2$ *is maximum over all optimal schedules.*

Intuitively, in a square-optimal schedule small $f(i)$ values are as small as possible and large $f(i)$ are as large as possible.

We will give a pseudo-polynomial algorithm (polynomial in n and p) that computes an optimal schedule, for given demands on nodes of a path.

Definition 2 *Given a schedule* Ψ*, we say that:*

i *is a* local minimum, *if* $f(i-1) > f(i)$ *and* $f(i) < f(i+1)$*; we will also regard* 0 *and* $n+1$ *as local minima.*

i *is a* local maximum, *if* $f(i-1) < f(i)$ *and* $f(i) > f(i+1)$*;*

i *is a* stair-up, *if* $f(i-1) < f(i) < f(i+1)$ *and* i *is a* stair-down, *if* $f(i-1) > f(i) > f(i+1)$; *When no distinction is needed, we simply call them* stairs.

i *is* compact, *if* $f(i) = x(i)$.

Let us use the following visualizing expressions. We say that i is *black* on level a, if $a \in \Psi(i)$, and i is *white* on level a if $a \notin \Psi(i)$ (see Fig. 1). Trivially, if i is compact then it is not white on any level under $f(i)$.

Sometimes for $i < j$ we also say, that the ordered pair (i,j) is *black-white, black-black,... etc.* on level a. Note that $(i, i+1)$ cannot be black-black on any level.

Proposition 1 *In any optimal schedule, the number of levels where* (i,j) *can be black-black, white-black or black-white is bounded from above by* p *each. The number of levels under* $f(i,j)$ *where* (i,j) *might be white-white is bounded from above by* $2p$.

Proof. The first statement is obvious. In order to see the second statement, suppose, that $f(i,j) = f(i)$. On any level under $f(i)$, where i is white, at least one of $i-1$ and $i+1$ must be black. Therefore, there are at most $2p$ such levels. $\square$

Definition 3 *An* (i,j) *pair is* conflicting on level a, *if either*

$i \equiv j \pmod 2$ *and* (i,j) *is black-white, or white-black, or*

$i \not\equiv j \pmod 2$ *and* (i,j) *is black-black, or white-white.*

Proposition 2 *If* (i,j) *is conflicting on level* a*, then* $\exists k \in [i, j-1]$ *such that* $(k, k+1)$ *is white-white on level* a. $\square$

Definition 4 *Suppose that* $\forall k \in [i,j]$, $f(k) \geq \max(a,b)$ *in a schedule* Ψ. *In this case we will say that we* change the levels a and b on $[i,j]$, *if* $\forall k \in [i,j]$ *we make* k *white (black) on level* a *if and only if according to* Ψ *it was white (black) on level* b, *and we make* k *white (black) on level* b, *if and only if it was white (black) on level* a.

After carrying out this operation, we may have to make corrections to get a proper schedule again. Note, that we will have to check the pairs $(i-1,i)$ and $(j,j+1)$ on the levels a and b.

Proposition 3 *If* i *is a stair-up in a square-optimal schedule* Ψ, *then* $(i-1,i)$ *is either white-black or black-white on any level* $a \leq f(i)$. *A symmetric statement holds if* i *is a stair-down.*

Proof. Trivially, $(i-1,i)$ is white-black on the level $f(i)$. We have to show that it is not white-white on any level below $f(i)$. Suppose that on the contrary, $(i-1,i)$ is white-white on level $a < f(i)$. Let M be the first local maximum to the right of i, and let's change the levels a and $f(i)$ on $[i,M]$. This decreases $f(i)$ by at least one. Also, this is a proper schedule on $[i-1,i]$, since $i-1$ is white on the level a. If it is a proper schedule on $[M,M+1]$, then we decreased the optimum sum, contradiction; if it is not a proper schedule on $[M,M+1]$, then we make M white either on the level a or on the level $f(i)$, and make it black on the level $f(M)+1$ (increase $f(M)$ by one). Now we created another optimal schedule, but increased the sum of squares of finishing times, again a contradiction since Ψ was square-optimal. □

Now, let i be a local minimum in a square-optimal schedule and M be the first local maximum to the right of i. If we know $f(i)$, then for all stair-ups $i < k < M$ we know $f(k) = x(k-1) + x(k)$, we know the hours in $\Psi(k) \cap [f(i), f(k)]$, and we know how many black and how many white levels k has under the level $f(i)$. If we even know $\Psi(i)$, then we know $\Psi(k)$ for all of the above k.

Proposition 4 *If* k *is a local minimum in an optimal schedule then* $f(k) \leq 3x(k)$.

Proof. Suppose that $f(k) > 3x(k)$. Let M and M' be the closest local maximum to the left and right of k, respectively. By changing the first $x(k)$ levels to those where k is black, we could make k compact, gaining more than $2x(k)$ at k and losing at most $x(k)$ at both M and M'. □

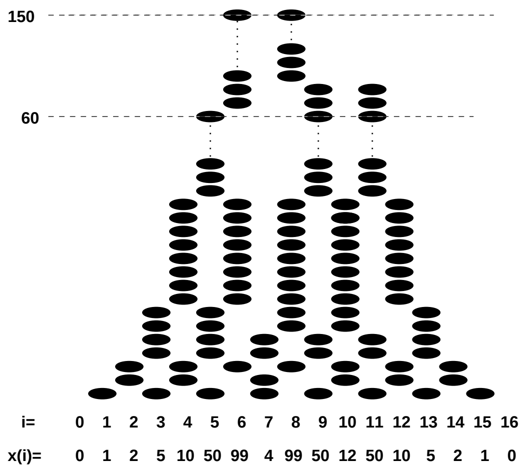

Figure 1: A square-optimal solution for the given pSMC problem on a path of length 15. The convex pairs are $(0, 16)$, $(0, 7)$, $(7, 16)$, $(7, 10)$ and $(10, 16)$. Other examples: $pit(7, 16) = 10$; $top(7, 10) = 8$; $\ell(10) = 7$ and $r(10) = 13$.

3 An Outline of the Algorithm and Further Definitions

Definition 5 *With respect to a given schedule Ψ, let $i < j$ be both local minima with the property that if $i < k < j$ is a local minimum between them, then $f(k) \geq \max(f(i), f(j))$. We will say that such an (i, j) pair is a* convex pair.

Definition 6 *For a convex pair (i, j), in a schedule Ψ, let $pit(i, j) = k$ if*

1. $i < k < j$

2. k is a local minimum, and

3. if $i < k' < j$ and k' is a local minimum, then $f(k) < f(k')$ or $(f(k) = f(k')$ and $k \leq k')$.

If there is no local minimum between i and j, then let $pit(i, j) = \emptyset$ and let $top(i, j)$ denote the unique local maximum between i and j.

In other words, $pit(i, j)$ is the leftmost local minimum of minimum finishing time between i and j. Note that if $k = pit(i, j)$, then both (i, k) and (k, j) are convex pairs.

The algorithm tests for every pair (i, j), whether it can be a convex pair in a square-optimal schedule. It proceeds from short distances to long ones – i.e. first it tests for each pair of the form $(i, i + 2)$ and at last $(0, n + 1)$. It proceeds dynamically by testing for each k between i and j, if it can be $pit(i, j)$, or $top(i, j)$. We will add a computed optimum on $[i + 1, k - 1]$ and on $[k+1, j-1]$, plus a computed $f(k)$, to obtain a possible sum of finishing times on $[i+1, j-1]$.

Of course, an optimum that we calculated this way is only realizable if we can indeed 'glue' two optimal schedules by the schedule of k. Therefore, in addition to testing k, we will also have to one by one fix and test some characteristic of the schedule on i, j and k.

For the time being, let us suppose that if $k = pit(i,j)$ then $f(k) < f(i+1)$ and $f(k) < f(j-1)$. That is, we disregard the fact that there might be several stairs on the two sides of $[i,j]$, having finishing time smaller than $f(k)$, or stair-ups finishing under $f(j)$ e.g. if $f(i) < f(j)$ (see Fig. 1).

Now, we will need to know in advance the number of black-black, white-black, black-white and white-white levels concerning the (i,j) pair. As we just need the number, and not the location of these levels, testing for all $\mathcal{O}(p^4)$ possibilities would still result in a pseudo-polynomial algorithm.

Definition 7 *For a convex pair $[i,j]$ in a fixed schedule Ψ, let $C(i,j) \in [0,2p]^4$ be the 4-tuple denoting the number of levels under $f(i,j)$, where (i,j) is black-black, white-black, black-white and white-white, respectively. We will call $C(i,j)$ the* color scheme *of (i,j). For the triple $i < k = pit(i,j) < j$, we will talk about the color scheme $C(i,k,j) \in [0,2p]^8$ under the level $f(k)$, in the same sense.*

We denote by $C(i,k,j) \Rightarrow C(i,j)$, when they are consistent with each other, i.e., the number of black-white-black and the number of black-black-black levels of (i,k,j) sum up to the number of black-black levels of (i,j), and so on for the remaining cases. Similarly, we use the notations $C(i,k,j) \Rightarrow C(i,k)$ and $C(i,k,j) \Rightarrow C(k,j)$.

Remark 1 *Note that the four numbers in $C(i,j)$ sum up to $f(i,j)$ and the eight numbers in $C(i,k,j)$ sum up to $f(k)$. Note also, that $C(i,k,j) \Rightarrow C(i,j)$ implies that the number of white-white-white plus the number of white-black-white levels in $C(i,k,j)$ equals the number of white-white levels in $C(i,j)$ plus $f(k) - f(i,j)$.*

Here is where we implicitly exploit that (i,j) is supposed to be a convex pair, and therefore $f(k) \geq f(i,j)$. For a fixed $C(i,j)$ we will want to test for all possible $C(i,k,j)$, for which $C(i,k,j) \Rightarrow C(i,j)$ holds. Suppose, we wanted to calculate the optimum by taking also $f(k) < f(i,j)$ values into consideration. For this calculation we would need in advance the color scheme of (i,j) under each possible $f(k)$ level. On the other hand, knowing $C(i,j)$ under all of the levels, boils down to having $\Psi(i)$ and $\Psi(j)$ and that is exactly what we tried to avoid, as testing all that, would lead beyond polynomial time.

For each pair $0 < i < j < n+1$ and each possible color scheme $C(i,j)$ of a pair, the algorithm computes an optimum sum of finishing times $F(i,j,C(i,j)) = \sum_{l=i+1}^{j-1} f(l)$ supposing that (i,j) is a convex pair. In particular, we test for each $k \in [i+1, j-1]$ to be $pit(i,j)$ and we test for each color scheme $C(i,k,j) \Rightarrow C(i,j)$.

For $C(i,k,j) \Rightarrow C(i,k)$ and $C(i,k,j) \Rightarrow C(k,j)$ we have a previously computed $F(i,k,C(i,k))$ and a $F(k,j,C(k,j))$ value. We obtain the sum of finishing times for this k and $C(i,k,j)$ by $F(i,k,C(i,k)) + F(k,j,C(k,j)) + f(k)$.

We also test if this k can be $top(i,j)$. In that case the sum of finishing times is computed easily based on Proposition 3 and on $C(i,j)$. Finally we choose the smallest sum of finishing times to be $F(i,j,C(i,j))$ and remember one $k = pit(i,j)$ and one $C(i,k,j)$, or a $k = top(i,j)$ that yielded this value.

In the end of this process we will have all the local minima and maxima in an optimal schedule and we will know the color scheme of convex pairs of minima. This is enough information to create such a schedule, starting with a minimum of smallest finishing time:

Suppose we have the schedule on i and j : $\Psi(i)$ and $\Psi(j)$, and they imply a color scheme $C(i,j)$. We take e.g. the best $k = pit(i,j)$ and $C(i,k,j)$ reported by the algorithm. Then we insert the 'blacks' and 'whites' of k (give the schedule of k) according to $\Psi(i)$, $\Psi(j)$ and $C(i,k,j)$. This is possible – though in general not uniquely determined –, because $C(i,k,j)$ is consistent with $C(i,j)$. We proceed with the scheduling in this way, until there is a convex pair where the optimum was yielded by a $k = top(i,j)$, in which case the scheduling on $[i,j]$ is obvious.

The algorithm is still a bit more complicated than this, because we may have stairs on the two sides of $[i,j]$ that we did not yet consider. For talking about this, first we need one more definition:

Definition 8 *Let i be a local minimum in a schedule Ψ. Let $i < r(i) \leq n+1$ be the node with the property $f(r(i)) < f(i)$ and $\forall k, i < k < r(i)$ $f(k) \geq f(i)$. We define $0 \leq \ell(i) < i$ symmetrically.*

Now, with these terms in hand, suppose we are testing the pair (i,j) and e.g. $f(i) > f(j)$ holds. Then we will need $C(i,r(i))$ instead of $C(i,j)$ and if $f(k) \gneq f(i)$, then we will need $C(\ell(k),k,r(k))$ instead of $C(i,k,j)$ (see Fig. 1). This won't make so much difference, because $[i+1,\ell(k)]$ consists of stair-ups as well as $[r(k), j-1]$ consists of stair-downs only, and there the schedule and finishing times easily follow. In particular, *the color schemes $C(i,r(i))$ and $C(\ell(k),k,r(k))$ can be defined like before, and the consistency $C(\ell(k),k,r(k)) \Rightarrow C(i,r(i))$ has a well defined meaning.*

Remark 2 *Note that $f(i,j)$, $F(i,j)$, $C(i,j)$, $pit(i,j)$, $r(i)$ etc. should all have $_\Psi$ as a subscript when they are relative to the schedule Ψ. We will only use subscripts when needed. We will use the subscript $_A$ when a value was yielded by the algorithm and is not necessarily realized by a schedule.*

4 An $\mathcal{O}(n^4p)$ Algorithm

In Section 3 the main idea of the algorithm was presented. Turning to the exact description we will make use of two technical theorems – Theorems 8 and 12 – which will help us simplify the algorithm. We do so not only in order to reduce the factor of p^7 of the previous rough version to p : The two theorems and the lemmas may as well be helpful during further investigation of the problem, and on the whole they also simplify our discussion.

In the main part we present an $\mathcal{O}(n^4p)$ time algorithm. After that we give a short argument on how it reduces to time $\mathcal{O}(n^3p)$.

Theorems 8 and 12 are based on the following Lemmas:

Lemma 5 *Let i be a local minimum in a square-optimal schedule, such that it is not compact, and let $\ell(i) \leq k \leq r(i)$. Then the following hold:*

The (i,k) pair is not conflicting on the level $f(i)$;

The (i,k) pair is not conflicting on any level where i is white.

Proof. Suppose for example, that $i < k \leq r(i)$, and (i,k) is conflicting on the level $f(i)$. Furthermore, let i be white on the level $a < f(i)$.

According to Proposition 2, there is an $(s, s+1)$ pair of nodes that is white-white on level $f(i)$, and $i \leq s < k$. Let M be the first local maximum to the left of i. Now all the finishing times on $[M,s]$ are not smaller than $f(i)$.

Let's change the levels a and $f(i)$ on $[M,s]$. Since $(s,s+1)$ was white-white, this remains a proper schedule on $[s,s+1]$ and it reduces $f(i)$ by at least 1. If it is not a proper schedule on $[M-1,M]$, we can correct it by increasing $f(M)$ by 1 (like in the proof of Proposition 3). This remains an optimal schedule, but improves the former schedule concerning square-optimality, which is a contradiction.

If (i,k) is conflicting on the level a, then we can make the same changing of levels and obtain a contradiction. $\Box$

Corollary 6 *If $f(i) > x(i)$ then $i \not\equiv r(i) (\mathrm{mod} 2)$, and $i \not\equiv \ell(i) (\mathrm{mod} 2)$.*

Proof. Otherwise $(\ell(i), i)$ or $(i, r(i))$ would be conflicting on the level $f(i)$. $\Box$

Lemma 7 *If i is a non compact local minimum in a square-optimal schedule, then for any level $a \leq f(i)$ there is at most one $i \leq k < r(i)$ and at most one $\ell(i) \leq k < i$ such that $(k, k+1)$ is white-white on the level a.*

Proof. Suppose that on the contrary, we have $i \leq k < k' < r(i)$ such that $(k, k+1)$ and $(k', k'+1)$ are both white-white on level a. Note that by Lemma 5 i must be black on the level a.

Let i be white on the level b. If $i \equiv k \pmod 2$ then k is white on the level b (see Lemma 5). Then we change the levels a and b on $[k+1, k']$; If $i \not\equiv k \pmod 2$ then k is white on the level $f(i)$. Then we change the levels a and $f(i)$ on $[k+1, k']$.

In both cases we obtained a proper square-optimal schedule (note that finishing times did not change), that contradicts Lemma 5, because $(i, k+1)$ is conflicting on level $f(i)$ or on level b. □

Now, Theorem 8 shows that we don't have to test for all $\mathcal{O}(p^4)$ possible tuples of $C(i, r(i))$ (resp. $C(\ell(j), j)$), because it is uniquely determined by $f(i,j)$.

Definition 9 *Let (i, j) be a convex pair. According to $f(i) > f(j)$, $f(i) < f(j)$ or $f(i) = f(j)$ we will use the name* relevant scheme of (i, j) *for the color scheme $C(i, r(i))$, $C(\ell(j), j)$ or $C(i, j)$, respectively.*

Theorem 8 *Let (i, j) be a convex pair in a square-optimal schedule. Either having $f(i, j) \neq \max(x(i), x(j))$, or having $f(i, j) = \max(x(i), x(j))$, and $k = top(i, j)$, the relevant scheme of (i, j) can be computed in $\mathcal{O}(n)$ time.*

The proof will make use of the following lemmas:

Lemma 9 *Let (i, j) be a convex pair in a square-optimal schedule. If $f(i) = f(j)$ then i and j are compact and $f(i) = x(i) = x(j) = f(j)$.*

Proof. We show that if $f(i) = f(j)$ then i and j are compact. Were i or j non-compact, then according to Lemma 5, $i \equiv j (mod 2)$ must hold, otherwise they would be conflicting on the level $f(i)$. Therefore, i and j must be white on exactly the same levels, again by Lemma 5. Now, if there is a level $a < f(i)$, where both of them are white, then let's change the levels a and $f(i)$ on $[M, M']$, where the latter are the first local max to the left of i and to the right of j, respectively. If necessary, let's increase $f(M)$ and $f(M')$ by one each. $f(i)$ and $f(j)$ were decreased, so what we get is still an optimal solution, but we could increase the sum of squares of finishing times, a contradiction. □

In the following two lemmas $f(i, j) = f(i) > f(j)$. Symmetric counterparts of the lemmas can be stated when $f(i) < f(j)$.

Lemma 10 *Let (i,j) be a convex pair and $f(i) > f(j)$. Then there is a node $i < s \leq j$ with the property that $x(s-1)+x(s) \geq f(i) > x(s)+x(s+1) > \ldots > x(j-1)+x(j)$.*

If $f(i) > x(i)$ then $r(i) = s$.

If $f(i) = x(i)$ and $k = top(i,j)$, then $r(i) = \max(k+1, s)$.

Proof. First let $f(i,j) = f(i) > x(i)$ that is, i is non-compact.

Note that $r(i) \leq j$ and $r(i) \ldots (j-1)$ is a (possibly empty) series of step-downs. Let $r = r(i)$. We show that $x(r-1)+x(r) \geq f(i) > x(r)+x(r+1) > \ldots > x(j-1)+x(j)$. Since there is at most one node with this property, this proves the first part of the lemma.

If $r-1$ is also a step-down, then the statement trivially holds due to the greedy schedule on the stairs.

Let $r-1$ be a local maximum. We claim that under the level $f(i)$, $r-1$ is colored as if it were still a step-down. Suppose that on the contrary, $(r-1,r)$ is white-white on the level $a \leq f(i)$. But that would imply that (i,r) is conflicting on this level (see Lemmas 5 and 7). Recall however, that i and $r = r(i)$ have opposite parity, and therefore (i,r) might only be conflicting on a level where it is black-black, so r cannot be white on level a, contradiction. From this the inequality $x(r-1)+x(r) \geq f(i)$ follows.

Second, let $f(i) = x(i)$. Now we can't say anything about the parity of $r(i)$. Let s be a node, for which $x(s-1)+x(s) \geq f(i) > x(s)+x(s+1) > \ldots > x(j-1)+x(j)$ holds. If $r(i)-1$ is a step-down, then $r(i) = s$ must hold, like before. If $r(i)-1$ is a local maximum, then $r(i)-1 = k$. Therefore, $r(i) = \max(k+1, s)$, or $r(i) = k+1$ if s does not exist. $\square$

Note that there is at most one such s as in the statement of Lemma 10, and it can be found in $\mathcal{O}(n)$ steps.

Lemma 11 *Let (i,j) be a convex pair and $f(i) > f(j)$. If we have $f(i)$ and $r(i)$ then $C(i, r(i))$ can be computed in $\mathcal{O}(1)$ time.*

Proof. We prove that if i, $f(i)(= f(i,j))$ and $r(i)$ are given, then in $\mathcal{O}(1)$ time $C(i, r(i))$ can be calculated.

Let first $i \not\equiv r(i) (\mathrm{mod} 2)$. Now, Lemma 5 implies that $(i, r(i))$ is not white-white on any level. We get that the number of white-black levels is $f(i) - x(i)$, the number of black-white levels is $f(i) - x(r(i))$ and the number of black-black levels is $x(r(i)) + x(i) - f(i)$.

Second, let $i \equiv r(i) (\mathrm{mod} 2)$. By Corollary 6 this can only happen when i is compact. Consequently, the number of black-black levels is $x(r(i))$ and the number of black-white levels is $x(i) - x(r(i))$. $\square$

Proof of Theorem 8. Note that knowing if $f(i)$ or $f(j)$ is larger, is not a condition in the theorem. Namely, it is either obvious or irrelevant: If $x(i) + x(i+1) < f(i,j)$ then $f(i,j) = f(j) > f(i)$. Otherwise for i as a local minimum $f(i) > x(i) + x(i+1)$ would hold, and there would be white-white conflicts between $(i, i+1)$, contradicting Lemma 5. Similarly, if $x(j) + x(j-1) < f(i,j)$ then $f(i,j) = f(i) > f(j)$. In the first case we compute $\ell(j)$ (Lemma 10) and then $C(\ell(j), j)$ (Lemma 11), respectively in the second we compute $r(i)$ and $C(i, r(i))$.

Finally, if $\max(x(i), x(j)) \leq f(i,j) \leq \min(x(i)+x(i+1), x(j)+x(j-1))$ then exactly one of $f(i) = f(j)$, $\ell(j) = i$ or $r(i) = j$ holds. So in any case we compute $C(i,j)$ (Lemma 9 or Lemma 11) independently of which case we are in. □

From the next theorem it follows that $f(k)$, $\ell(k)$, $r(k)$ and $C(\ell(k), k, r(k))$ are in 'most' cases uniquely determined by $f(i,j)$ and k, and in just one remaining case possible values of $f(k)$ must be tested for as well.

Theorem 12 *Let (i,j) be a convex pair, $k = pit(i,j)$ and k be non-compact in a square-optimal schedule.*

If $f(i,j) > \max(x(i), x(j))$ then $(\ell(k), k, r(k))$ can only be black-white-black, black-black-white, white-black-black or white-black-white.

If $f(i,j) = \max(x(i), x(j))$ then in addition to the previous four, a black-black-black triple is also possible.

Proof. Let k be a non-compact local minimum. By Corollary 6 then $\ell(k) \not\equiv k \not\equiv r(k) \pmod 2$.

It follows directly from Lemma 5 that k is black on the levels where at least one of $\ell(k)$ and $r(k)$ is white.

If moreover $k = pit(i,j)$ then $i \leq \ell(k)$ and $r(k) \leq j$ because k is non-compact and therefore $f(i,j) < f(k)$ (see Lemma 9). Let $f(i) = f(i,j) > x(i)$, let i be white on the level b, and suppose that $(\ell(k), k, r(k))$ is black-black-black on the level $a < f(k)$. Then $\exists s, s'$ such that $\ell(k) < s < k < s' < r(k)$ and $(s, s+1)$ and $(s', s'+1)$ are white-white on the level a. Now we can change the levels a and $f(i)$ or a and b on $[s+1, s']$ like in the proof of Lemma 7 and obtain a square-optimal schedule that contradicts Lemma 5. □

For examples of Theorem 12 see Fig. 1. The first case is valid if e.g. $(i,k,j) = (7, 10, 16)$, and the second, if $(i,k,j) = (0, 7, 16)$. Compare the levels 2 and 3 of $(\ell(k), k, r(k)) = (2, 7, 14)$.

Now we describe two basic steps of the algorithm by the following simple lemmas.

Lemma 13 *Let (i,j) be a convex pair of minima in a square-optimal solution Ψ, and let $k = top_\Psi(i,j)$. It is possible to compute $f_\Psi(k)$ in $\mathcal{O}(n)$ steps if $f_\Psi(i,j)$ is known.*

Proof. If $k = top(i,j)$, then the – possibly empty – subpaths $[i+1, k-1]$ and $[k+1, j-1]$ consist of stair-ups and stair-downs, respectively. By Proposition 3, the schedule on these stairs is determined by $\Psi(i)$ and $\Psi(j)$ in a kind of 'greedy' way. Finally, $\Psi(k)$ is also greedy, in that k is black on a level if and only if $(k-1, k+1)$ is white-white.

Let e.g. $f(i) > f(j)$. Now $i < r(i) \le j$. According to Theorem 8, $C(i, r(i))$ can be computed in $\mathcal{O}(n)$ time. We proceed along the levels from the bottom up, and sum up the number of levels, where k is black. Under the level $f(i,j)$ we calculate this number from $C(i, r(i))$, above the level $f(i,j)$ it is easy to exactly calculate the schedule on $[i+1, j-1]$.

Suppose e.g. that $i \equiv k \not\equiv r(i) \pmod 2$. Now we have the number of levels under $f(i) = f(i,j)$ where $(i, r(i))$ is black-white, and these are the same levels where $(k-1, k+1)$ is white-white. We start with this value, and go on summing as follows.

Above the level $f(i)$, the coloring of levels changes at finishing times of the stairs on the two sides, so we will have to merge the increasing number series

$$x(i) + x(i+1), \quad x(i+1) + x(i+2), \quad \ldots \quad , x(k-2) + x(k-1)$$

and

$$x(r(i)) + x(r(i)-1), \quad x(r(i)-1) + x(r(i)-2),$$
$$\ldots \quad , x(k+2) + x(k+1)$$

and sum up the number of levels when we passed – the finishing time of – a stair of opposite parity to k on both sides and did not yet finish the next stair on either of the sides. These are the levels where k is black. Merging and summing takes $\mathcal{O}(n)$ time

In the end the remaining demand – $x(k)$ lessened by the resulting sum – is added to $\max(x(k-1) + x(k-2), x(k+1) + x(k+2))$ and this yields the finishing time of k. □

Lemma 14 *Let (i,j) be a convex pair of minima in a square-optimal solution Ψ, and let $k = pit_\Psi(i,j)$. If $f_\Psi(i,j)$ is known and $f_\Psi(i,j) > \max(x(i), x(j))$, then it is possible to compute $f_\Psi(k)$ in $\mathcal{O}(n)$ steps.*

Proof. The argument of the proof is much the same as in the proof of Lemma 13.

Let e.g. $f_\Psi(i) > f_\Psi(j)$ and let $f_\Psi(i) > x(i)$. Now we can also suppose $f(k) \gneq f(i)$ because the case $f(k) = f(i)$ would imply that i is compact. Now $i \leq \ell(k) < k < r(k) \leq r(i) \leq j$. Also, since $k = pit(i,j)$, it is easy to see that $[i+1, \ell(k)]$ consists of zero or more stair-ups as well as $[r(k), j-1]$ consists of zero or more stair-downs.

We proceed from the bottom and sum up the levels where k is black, until the sum reaches $x(k)$. Under $f(i,j)$ we calculate from $C(i, r(i))$, above that we know the schedule on $[i+1, \ell(k)]$ and on $[r(k), j-1]$.

According to Theorem 8, $C(i, r(i))$ can be computed in $\mathcal{O}(n)$ steps. Now, if e.g. $i \equiv k \not\equiv r(i) (\mathrm{mod} 2)$, then the levels under $f(i) = f(i,j)$ where k is black, are those where $(i, r(i))$ is black-white, black-black or white-white (compare Theorem 12). The number of these levels is the starting value in the summation. Note again, here is where we used that (i,j) is a convex pair and $f(k) \geq f(i,j)$.

Above the level $f(i)$ we go on summing as follows. As before, we merge the increasing series of finishing times on the two sides:

$$x(i) + x(i+1), \quad x(i+1) + x(i+2), \quad \ldots$$

and

$$x(r(i)) + x(r(i)-1), \quad x(r(i)-1) + x(r(i)-2), \quad \ldots$$

Here we *don't* include in the sum those levels when we passed a stair of the *same* parity as of k on both sides and did not yet finish the next stair on either of the sides. These are the levels where k is *not* black because $(\ell(k), r(k))$ is black-black (compare Theorem 12).

All the other levels we sum up until we get $x(k)$ and there we stop. The level where we stopped is $f_\Psi(k)$. The stair-up on the left and the stair-down on the right whose finishing time we passed for last, are $\ell(k)$ and $r(k)$, respectively. Merging and summing takes $\mathcal{O}(n)$ time. □

We will run the algorithms of Lemma 13 and Lemma 14 on all possible inputs i, j, k and $f = f(i,j)$ to test what $f(k)$ it would result if (i,j) were a convex pair and k were $top(i,j)$ or $pit(i,j)$. The potential finishing time of k, computed this way will be denoted by $fmax_A(i,j,f,k)$ and $fmin_A(i,j,f,k) = fmin_A(k)$, respectively. It may happen that we don't get any numeric result, if e.g. the s of Lemma 10 does not exist or the merged series are not increasing, etc. This means that the tested setting is impossible and this we denote by $fmax_A(i,j,f,k) = \infty$ or $fmin_A(i,j,f,k) = \infty$, respectively.

Now an exact description of the algorithm can follow. The algorithm has two phases.

```
for $d = 2 \ldots n+1$ do
  for $0 \le i < j \le n+1,\ j - i = d$ do
    for $f = \max(x(i), x(j)) \ldots 3\max(x(i), x(j))$ do
      for $k = i+1 \ldots j-1$ do
        compute $fmax(i,j,f,k)$ in time $\mathcal{O}(n)$ (Lemma 13)
      end for
      $Fmax := \min_{i<k<j}(\sum_{s=i+1}^{k-1}(x(s-1)+x(s)) + fmax(i,j,f,k) +$
                $+ \sum_{s=k+1}^{j-1}(x(s+1)+x(s)))$
      $top(i,j,f)$ := the $k$ providing the minimum value
      for  $k = i+2 \ldots j-2$ do
        if $f = \max(x(i), x(j))$ then
          compute $\min_{x(k) \le fk \le 3x(k)}(F(i,k,fk) + fk + F(k,j,fk))$
          $fmin(i,j,f,k)$ := the $fk$ providing the minimum
        else
          compute $fmin(i,j,f,k)$ in time $\mathcal{O}(n)$ (Lemma 14)
        end if
      end for
      $Fmin := \min_{i<k<j}(F(i,k,fmin(i,j,f,k)) + fmin(i,j,f,k) +$
                $+F(k,j,fmin(i,j,f,k)))$
      $pit(i,j,f)$ := the $k$ providing the minimum value
      if $Fmax \le Fmin$ then
        $F(i,j,f) := Fmax$ and $pit(i,j,f) := \emptyset$
      else
        $F(i,j,f) := Fmin$ and $top(i,j,f) := \emptyset$
      end if
    end for
  end for
end for
```

Figure 2: First phase of the $\mathcal{O}(n^4 p)$ algorithm.

First phase. In this phase we compute an optimal structure: locations and finishing times of local minima, and locations of local maxima. We won't do the scheduling here, but we will get the optimum sum of finishing times as a result.

For all $0 \le i < j \le n+1$ where $i+2 \le j$ and all f where $\max(x(i), x(j)) \le f \le 3\max(x(i), x(j))$, we will compute $F_A(i, j, f)$ with the following meaning: if in a Ψ square-optimal solution (i, j) is a convex pair of minima, and $f_\Psi(i, j) = f$, then $\sum_{s=i+1}^{j-1} f_\Psi(s) = F_A(i, j, f)$.

We proceed from short $[i, j]$ subpaths to longer ones, using a dynamic programming strategy.

First compute

$$Fmax = \min_{k\in[i+1,j-1]} \Big(\sum_{s=i+1}^{k-1} (x(s-1) + x(s)) + fmax_A(i, j, f, k) +$$

$$+ \sum_{s=k+1}^{j-1} (x(s+1) + x(s)) \Big)$$

in $\mathcal{O}(n^2)$ time (see Lemma 13).

Second, if $f > \max(x(i), x(j))$ then compute

$$Fmin = \min_{k\in[i+2,j-2]} (F_A(i, k, fmin_A(i, j, f, k)) + fmin_A(i, j, f, k)+$$

$$+F_A(k, j, fmin_A(i, j, f, k)))$$

in $\mathcal{O}(n^2)$ time (see Lemma 14).

If $f = \max(x(i), x(j))$, then we have to test for the possible values of the finishing time of $k = pit(i, j)$; we will denote the values put to test by fk. Thus, we compute

$$Fmin = \min_{k\in[i+2,j-2], fk\in[x(k),3x(k)]} (F_A(i, k, fk) + fk + F_A(k, j, fk))$$

in $\mathcal{O}(n^2 p)$ time. If k and fk provided the minimum sum, then we assign the fk value to $fmin_A(i, j, f, k)$.

We obtain

$$F_A(i, j, f) = \min(Fmax, Fmin).$$

Again, $F_A(i, j, f) = \infty$ means that (i, j) has lost the chance to be a convex pair of minima with $f(i, j) = f$ in any square-optimal schedule. If $F_A(i, j, f) \ne \infty$ then together with the value $F_A(i, j, f)$ we also store which $k \in [i + 1, j - 1]$ resulted the minimum. If it was a unique local maximum (that is if $Fmax \leq Fmin$) then we record it by $top_A(i,j,f) = k$. Otherwise we record $pit_A(i,j,f) = k$.

The computation for one (i,j) pair takes $\mathcal{O}(n^2p)$ time. Overall computation of the First phase takes $\mathcal{O}(n^4p)$ time.

Theorem 15 *If Φ is a valid schedule of the given demands on the path $[1,n]$, then $\sum_{s=1}^{n} f_\Phi(s) \geq F_A(0,n+1,0)$.*

Proof. Suppose that on the contrary, the optimum sum of finishing times is strictly smaller than $F_A(0,n+1,0)$. Then there exists a Ψ square-optimal schedule such that $\sum_{s=1}^{n} f_\Psi(s) < F_A(0,n+1,0)$.

Let's consider all the convex pairs (i,j) in Ψ, such that $\sum_{s=i+1}^{j-1} f_\Psi(s) < F_A(i,j,f_\Psi(i,j))$, e.g. $(0,n+1)$ is one of them. We proceed by induction on the length of subpaths, and suppose (i,j) is one of minimum distance $j-i$ among all these pairs. Let $f = f_\Psi(i,j)$.

First let $pit_\Psi(i,j) = \emptyset$, that is, we have a $k = top_\Psi(i,j)$. The algorithm also tested this k to be $top(i,j)$.

Now, according to Proposition 3, the finishing times on $[i+1,k-1]$ and $[k+1,j-1]$ are $f_\Psi(i+1) = x(i)+x(i+1), \ldots, f_\Psi(k-1) = x(k-2)+x(k-1)$ and $f_\Psi(k+1) = x(k+2)+x(k+1), \ldots, f_\Psi(j-1) = x(j)+x(j-1)$.

Also, above the level $f_\Psi(i,j)$ the scheduling is uniquely determined and obviously, k will be black where $(k-1,k+1)$ is white-white. Under $f_\Psi(i,j)$ $C_\Psi(i,j)$ (or $C_\Psi(i,r(i))$ or $C_\Psi(\ell(j),j)$) must be as determined by Theorem 8, and this again determines the number of levels where $(k-1,k+1)$ is white-white. This implies that $f_\Psi(k)$ is uniquely determined by $f_\Psi(i,j)$ and equals $fmax_A(i,j,f_\Psi(i,j),k)$.

But then

$$\sum_{s=i+1}^{j-1} f_\Psi(s) =$$

$$= \sum_{s=i+1}^{k-1} (x(s-1)+x(s)) + fmax_A(i,j,f,k) + \sum_{s=j-1}^{k+1} (x(s+1)+x(s)) \geq$$

$$\geq F_A(i,j,f),$$

a contradiction.

Second, let $pit_\Psi(i,j) = k$. Now again $C_\Psi(i,j)$ (or $C_\Psi(i,r(i))$ or $C_\Psi(\ell(j),j)$) are as determined by Theorem 8. This either uniquely determines $f_\Psi(k)$ according to Theorem 12 and consequently $f_\Psi(k) = fmin_A(i,j,f_\Psi(i,j),k)$. In this case, since $j-i$ was minimum,

$$\sum_{s=i+1}^{j-1} f_\Psi(s) = \sum_{s=i+1}^{k-1} f_\Psi(s) + f_\Psi(k) + \sum_{s=k+1}^{j-1} f_\Psi(s) \geq$$

$$\geq F_A(i,k,f_\Psi(k)) + f_\Psi(k) + F_A(k,j,f_\Psi(k)) =$$

$$= F_A(i,k,fmin_A(i,j,f,k)) + fmin_A(i,j,f,k) + F_A(k,j,fmin_A(i,j,f,k)) \geq$$

$$\geq F_A(i,j,f),$$

a contradiction.

Or, we tested this $k = pit(i,j)$ with all possible fk finishing times, so with $fk = f_\Psi(k)$ as well. Then

$$\sum_{s=i+1}^{j-1} f_\Psi(s) \geq F_A(i,k,f_\Psi(k)) + f_\Psi(k) + F_A(k,j,f_\Psi(k)) =$$

$$= F_A(i,k,fk) + fk + F_A(k,j,fk) \geq F_A(i,j,f_\Psi(i,j)),$$

a contradiction. □

procedure $greedy(i,j)$
if $i < j$ **then**
  for $k := i+1 \ldots j$ **do**
    if k is not scheduled **then**
      assign k the smallest $x(k)$ hours that are not assigned to $k-1$
    end if
  end for
end if
if $j < i$ **then**
  for $k := i-1 \ldots j$ **do**
    if k is not scheduled **then**
      assign k the smallest $x(k)$ hours that are not assigned to $k+1$
    end if
  end for
end if

procedure $greedy(k)$
assign k the smallest $x(k)$ hours neither assigned to $k-1$ nor to $k+1$

procedure $minschedule(i,j,k,fk)$
$\ell(k) := \min\{s \geq i | x(s+1) + x(s) \geq fk\}$
$r(k) := \max\{s \leq j | x(s-1) + x(s) \geq fk\}$
$greedy(i,\ell(k))$
$greedy(j,r(k))$
assign k the hours below fk that don't conflict with at least one of $\Psi(\ell(k))$ or $\Psi(r(k))$; the missing hours from $x(k)$ should be the smallest possible, not yet assigned hours

procedure $schedule(i,j,f)$
if $top(i,j,f) \neq \emptyset$ **then**
  $k := top(i,j,f)$
  $greedy(i,k-1)$
  $greedy(j,k+1)$
  $greedy(k)$
else
  $k := pit(i,j,f)$
  $fk := fmin(i,j,f,k)$
  $minschedule(i,j,f,fk)$
  $schedule(i,k,fk)$
  $schedule(k,j,fk)$
end if

procedure $main()$
$schedule(0,n+1,0)$

Figure 3: Second phase.

Second phase. In this phase we give an optimal schedule Φ. Now we proceed from long subpaths – starting with $[0, n+1]$ – to shorter ones. The First phase provided the local minima in the form of $pit_A()$ values and their finishing times as $fmin_A()$ and the local maxima in the form of $top_A()$ values. While computing the schedule of these nodes we will basically follow the same algorithm as in Lemma 13 and 14 when we calculated their finishing times. Along the way we will also prove that Φ is valid schedule and optimal, because $\sum_{s=1}^{n} f_\Phi(s) = F_A(0, n+1, 0)$.

First we take $pit_A(0, n+1, 0)$ or $top_A(0, n+1, 0)$. One of them must exist, because there exist valid schedules on the path.

If we have a $k = top_A(0, n+1, 0)$ that means that there is just one local maximum in our optimal schedule. Then we do the scheduling in the greedy way – see Proposition 3 – on the stair-ups of $[1, k-1]$ and on the stair-downs of $[k+1, n]$. Finally we do the greedy scheduling on k, that is, we assign the hours where $(k-1, k+1)$ is white-white. Now we are ready with all the nodes, and the finishing times are the same as those resulting $F_A(0, n+1, 0)$, because only then is $k = top_A(0, n+1, 0)$ possible. Obviously, Φ is a valid schedule.

If on the other hand we have $k = pit_A(0, n+1, 0)$, that means that k is the (leftmost) smallest local minimum in our schedule. We have the finishing time $f_\Phi(k) := fmin_A(0, n+1, 0, k)$ as well. If k is compact then the schedule on k is trivial.

If k is not compact, then we recompute $\ell(k)$ and $r(k)$, while we are doing the greedy scheduling on the stairs $[1, \ell(k)]$ and $[Right(k), n]$. In the meantime we schedule k : if it is not compact then $\ell(k) \not\equiv k \not\equiv r(k) \pmod 2$, and we assign the hours to k where $(\ell(k), r(k))$ is white-white, white-black or black-white. If $x(k)$ is not completely used up till we reach $fmin_A(0, n+1, 0, k)$, then we are free to assign the rest of hours to any of the levels under $fmin_A(0, n+1, 0, k)$ where $(\ell(k), r(k))$ is black-black. In particular, we will fill the empty hours from the bottom up (see node 7 on Fig. 1).

Now it is easy to see that if $\sum_{s=1}^{k-1} f_\Phi(s) = F_A(0, k, fmin_A(k))$ and $\sum_{s=k+1}^{n} f_\Phi(s) = F_A(k, n+1, fmin_A(k))$ then $\sum_{s=1}^{n} f_\Phi(s) = F_A(0, n+1, 0)$.

Now we proceed recursively, first make the schedule on $[0, k]$ then on $[k, n+1]$. Exactly one of $top_A(0, k, fmin_A(k))$ or $pit_A(0, k, fmin_A(k))$, exists, in the first case we make the schedule on the whole subpath, in the latter case on $k' = pit(0, k)$, on $[\ell(k)+1, \ell(k')]$ and on $[r(k'), k-1]$. And so on...

The way we computed the $pit_A()$, $fmin_A()$ and $top_A()$ etc. values in the First phase, corresponds to creating the schedule in the Second phase. This guarantees that we will make ends meet, that is the $fmin_A()$ values are neither smaller than the demand, nor too high for the stairs on both sides, and all these details.

Analysis. We will call a a *preemption level*, if there is a node $i \in [0, n+1]$, such that i is black on a and white on $a+1$, or the other way round. The way we did the scheduling, any preemption level is either the finishing time of a node, or the last of the extra black levels in a local minimum (e.g. level 2 of node 7 on Fig. 1). Therefore, there are $\mathcal{O}(n)$ preemption levels.

We could make the schedule of any node through fixing its preemption levels, by way of merging or revisiting preemption levels of at most two other nodes. Thus, the Second phase requires $\mathcal{O}(n^2)$ time.

5 The $\mathcal{O}(n^3p)$ Algorithm

Let us fix an (i,j) pair and an $f = f(i,j) > \max(x(i), x(j))$ and suppose we are just testing this (i,j,f) triple. We claim that we can compute $fmax_A(i,j,f,k)$ and $fmin_A(i,j,f,k)$ overall for all $k \in [i+1, j-1]$ in $\mathcal{O}(n)$ time. This is enough to reduce the time to $\mathcal{O}(n^3p)$ (compare Fig. 2).

First note that in the computations of Theorem 8, the part that takes $\mathcal{O}(n)$ time is finding s in Lemma 10 and this is independent of k, therefore it can be done once for (i,j,f).

Second, in the algorithms of Lemma 13 and Lemma 14, the merging and summing procedure is exactly the same for $k \neq k'$ if $k \equiv k' (\mathrm{mod} 2)$. Suppose that the $x(k)$ demands of e.g. all even $k \in [i+1, j-1]$ values are ordered in non-decreasing order. Now, parallel to the merging of the two sides and summing the black levels, we just need to record the $fmax/fmin(i,j,f,.)$ finishing times of the k values by the given order of demands.

Let's do the ordering of the $x()$ values separately for the odd and even nodes at the beginning of the First phase, and for an (i,j) pair let's restrict the ordered series to the nodes between i and j.

The $\mathcal{O}(n^3p)$ Algorithm on Rings. We start the First phase like before: we compute the $F_A(i,j,f)$ values for all (i,j) paths not longer than n nodes and each possible f. Instead of computing $F_A(0, n+1, 0)$ in the end, we do the following: On a cycle, a node of minimum finishing time is a compact local minimum. We can test for each node to be such a node: the optimum sum will be $\min_{1 \leq i \leq n}(F_A(i,i,x(i)) + x(i))$, where $F_A(i,i,x(i))$ is the computed optimum on $i+1, i+2, \ldots, i+n-1 = i-1 (\mathrm{mod} n)$. Finally, we can start the scheduling with the compact local minimum.

6 Future work

It remains to be a challenging question, whether the pSMC on paths can be solved in time polynomial in n. We firmly believe that the answer to this question is yes. There doesn't seem to be an obvious way to exploit our idea on any graph class with maximum node degree $\Delta \geq 3$. It might be interesting to examine graph-classes with just a small number of nodes of degree ≥ 3.

Acknowledgements

I would like to thank Dániel Marx for the idea of searching for square-optimal schedules. Special thanks to Katalin Friedl, for turning my attention to the problem, and for her advice and plenty of help during the writing of this paper.

References

[1] A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *Journal of Algorithms*, 28:339–365, 1998.

[2] M. M. Halldórsson and G. Kortsarz. Tools for multicoloring with applications to planar graphs and partial k-trees. *Journal of Algorithms*, 42(2):334–366, 2002.

[3] E. Kubicka. *The Chromatic Sum of a Graph*. PhD thesis, Western Michigan University, 1989.

[4] D. Marx. The complexity of tree multicolorings. In *Proc. 27th Intl. Symp. Math. Found. Comput. Sci. (MFCS)*, LNCS. Springer, 2002.

[5] A. Bar-Noy M. Bellare M. M. Halldórsson H. Shachnai and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140:183–202, 1998.

[6] A. Bar-Noy M. M. Halldórsson G. Kortsarz H. Shachnai and R. Salman. Sum multicoloring of graphs. *Journal of Algorithms*, 37:422–450, 2000.

[7] M. M. Halldórsson G. Kortsarz A. Proskurowski R. Salman H. Shachnai and J. A. Telle. Multi-coloring trees. *Information and Computation*, 180(2):113–129, 2002.

[8] T. Szkaliczki. Routing with minimum wire length in the dogleg-free Manhattan model is NP-complete. *SIAM Journal on Computing*, 29(1):274–287, 1999.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2003-NWG2-002	F. Eisenbrand	Fast integer programming in fixed dimension
MPI-I-2003-NWG2-001	L.S. Chandran, C.R. Subramanian	Girth and Treewidth
MPI-I-2003-4-009	N. Zakaria	FaceSketch: An Interface for Sketching and Coloring Cartoon Faces
MPI-I-2003-4-008	C. Roessl, I. Ivrissimtzis, H. Seidel	Tree-based triangle mesh connectivity encoding
MPI-I-2003-4-007	I. Ivrissimtzis, W. Jeong, H. Seidel	Neural Meshes: Statistical Learning Methods in Surface Reconstruction
MPI-I-2003-4-006	C. Roessl, F. Zeilfelder, G. Nrnberger, H. Seidel	Visualization of Volume Data with Quadratic Super Splines
MPI-I-2003-4-005	T. Hangelbroek, G. Nrnberger, C. Roessl, H.S. Seidel, F. Zeilfelder	The Dimension of C^1 Splines of Arbitrary Degree on a Tetrahedral Partition
MPI-I-2003-4-004	P. Bekaert, P. Slusallek, R. Cools, V. Havran, H. Seidel	A custom designed density estimation method for light transport
MPI-I-2003-4-003	R. Zayer, C. Roessl, H. Seidel	Convex Boundary Angle Based Flattening
MPI-I-2003-4-002	C. Theobalt, M. Li, M. Magnor, H. Seidel	A Flexible and Versatile Studio for Synchronized Multi-view Video Recording
MPI-I-2003-4-001	M. Tarini, H.P.A. Lensch, M. Goesele, H. Seidel	3D Acquisition of Mirroring Objects
MPI-I-2003-2-003	Y. Kazakov, H. Nivelle	Subsumption of concepts in $DL\ \mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSPACE-complete
MPI-I-2003-2-002	M. Jaeger	A Representation Theorem and Applications to Measure Selection and Noninformative Priors
MPI-I-2003-2-001	P. Maier	Compositional Circular Assume-Guarantee Rules Cannot Be Sound And Complete
MPI-I-2003-1-014	G. Schfer	Average Case and Smoothed Competitive Analysis of the Multi-Level Feedback Algorithm
MPI-I-2003-1-013	I. Katriel, S. Thiel	Fast Bound Consistency for the Global Cardinality Constraint
MPI-I-2003-1-012	D. Fotakis, R. Pagh, P. Sanders, P. Spirakis	Space Efficient Hash Tables with Worst Case Constant Access Time
MPI-I-2003-1-011	P. Krysta, A. Czumaj, B. Voecking	Selfish Traffic Allocation for Server Farms
MPI-I-2003-1-010	H. Tamaki	A linear time heuristic for the branch-decomposition of planar graphs
MPI-I-2003-1-009	B. Csaba	On the Bollobás – Eldridge conjecture for bipartite graphs

MPI-I-2003-1-008	P. Sanders	Soon to be published
MPI-I-2003-1-007	H. Tamaki	Alternating cycles contribution: a strategy of tour-merging for the traveling salesman problem
MPI-I-2003-1-006	M. Dietzfelbinger, H. Tamaki	On the probability of rendezvous in graphs
MPI-I-2003-1-005	M. Dietzfelbinger, P. Woelfel	Almost Random Graphs with Simple Hash Functions
MPI-I-2003-1-004	E. Althaus, T. Polzin, S.V. Daneshmand	Improving Linear Programming Approaches for the Steiner Tree Problem
MPI-I-2003-1-003	R. Beier, B. Vcking	Random Knapsack in Expected Polynomial Time
MPI-I-2003-1-002	P. Krysta, P. Sanders, B. Vcking	Scheduling and Traffic Allocation for Tasks with Bounded Splittability
MPI-I-2003-1-001	P. Sanders, R. Dementiev	Asynchronous Parallel Disk Sorting
MPI-I-2002-4-002	F. Drago, W. Martens, K. Myszkowski, H. Seidel	Perceptual Evaluation of Tone Mapping Operators with Regard to Similarity and Preference
MPI-I-2002-4-001	M. Goesele, J. Kautz, J. Lang, H.P.A. Lensch, H. Seidel	Tutorial Notes ACM SM 02 A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2002-2-008	W. Charatonik, J. Talbot	Atomic Set Constraints with Projection
MPI-I-2002-2-007	W. Charatonik, H. Ganzinger	Symposium on the Effectiveness of Logic in Computer Science in Honour of Moshe Vardi
MPI-I-2002-1-008	P. Sanders, J.L. Trff	The Factor Algorithm for All-to-all Communication on Clusters of SMP Nodes
MPI-I-2002-1-005	M. Hoefer	Performance of heuristic and approximation algorithms for the uncapacitated facility location problem
MPI-I-2002-1-004	S. Hert, T. Polzin, L. Kettner, G. Schfer	Exp Lab A Tool Set for Computational Experiments
MPI-I-2002-1-003	I. Katriel, P. Sanders, J.L. Trff	A Practical Minimum Scanning Tree Algorithm Using the Cycle Property
MPI-I-2002-1-002	F. Grandoni	Incrementally maintaining the number of l-cliques
MPI-I-2002-1-001	T. Polzin, S. Vahdati	Using (sub)graphs of small width for solving the Steiner problem
MPI-I-2001-4-005	H.P.A. Lensch, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing and Interactive Display of High Quality 3D Models
MPI-I-2001-4-004	S.W. Choi, H. Seidel	Linear One-sided Stability of MAT for Weakly Injective Domain
MPI-I-2001-4-003	K. Daubert, W. Heidrich, J. Kautz, J. Dischler, H. Seidel	Efficient Light Transport Using Precomputed Visibility
MPI-I-2001-4-002	H.P.A. Lensch, J. Kautz, M. Goesele, H. Seidel	A Framework for the Acquisition, Processing, Transmission, and Interactive Display of High Quality 3D Models on the Web
MPI-I-2001-4-001	H.P.A. Lensch, J. Kautz, M. Goesele, W. Heidrich, H. Seidel	Image-Based Reconstruction of Spatially Varying Materials
MPI-I-2001-2-006	H. Nivelle, S. Schulz	Proceeding of the Second International Workshop of the Implementation of Logics
MPI-I-2001-2-005	V. Sofronie-Stokkermans	Resolution-based decision procedures for the universal theory of some classes of distributive lattices with operators
MPI-I-2001-2-004	H. de Nivelle	Translation of Resolution Proofs into Higher Order Natural Deduction using Type Theory
MPI-I-2001-2-003	S. Vorobyov	Experiments with Iterative Improvement Algorithms on Completely Unimodel Hypercubes
MPI-I-2001-2-002	P. Maier	A Set-Theoretic Framework for Assume-Guarantee Reasoning
MPI-I-2001-2-001	U. Waldmann	Superposition and Chaining for Totally Ordered Divisible Abelian Groups
MPI-I-2001-1-007	T. Polzin, S. Vahdati	Extending Reduction Techniques for the Steiner Tree Problem: A Combination of Alternative-and Bound-Based Approaches
MPI-I-2001-1-006	T. Polzin, S. Vahdati	Partitioning Techniques for the Steiner Problem