Improved Bounds for Scheduling Conflicting Jobs with Minsum Criteria

RAJIV GANDHI
Department of Computer Science
Rutgers University - Camden
and
MAGNÚS M. HALLDÓRSSON
Department of Computer Science
University of Iceland
and
GUY KORTSARZ
Department of Computer Science
Rutgers University - Camden
and
HADAS SHACHNAI
Department of Computer Science
The Technion

We consider a general class of scheduling problems where a set of *conflicting* jobs needs to be scheduled (preemptively or non-preemptively) on a set of machines so as to minimize the weighted sum of completion times. The conflicts among the jobs are formed as an *arbitrary* conflict graph.

Building on the framework of Queyranne and Sviridenko (*J. of Scheduling, 5:287-305, 2002*), we present a general technique for reducing the weighted sum of completion times problem to the classical makespan minimization problem. Using this technique, we improve the best known results for scheduling conflicting jobs with minsum objective, on several fundamental classes of graphs, including line graphs, $(k+1)$-claw free graphs and perfect graphs. In particular, we obtain the first constant factor approximation ratio for non-preemptive scheduling on interval graphs. We also improve the results of Kim (*SODA 2003, 97–98*) for scheduling jobs on line graphs and for resource-constrained scheduling.

1. INTRODUCTION

1.1 The problems

We consider a general class of problems in which jobs that utilize non-sharable resources need to be scheduled (preemptively or non-preemptively) on multiple machines. A pair of jobs, i and j, cannot be scheduled simultaneously if they compete over resources. The conflicts among the jobs are modeled by an arbitrary conflict graph, in which the vertices represent the jobs, and there is an edge between two vertices if the corresponding jobs cannot be scheduled simultaneously. An instance of a problem in the class is a pair (G, x), where $G = (V, E)$ is a conflict graph and x is a vector of *job lengths*. The value of x_v equals the length of the corresponding job. We may also be given vertex weights, with w_v denoting the weight of v.

A proper (or feasible) schedule for G is an assignment $\psi : V \to 2^{\mathbf{N}}$ of x_v integers to each $v \in V$, such that for each pair $(v, u) \in E$, $\psi(v) \cap \psi(u) = \emptyset$. Intuitively, the set of integers $\psi(v)$ is the set of unit-time *rounds* (i.e., time intervals of length one) in which the job v is being processed. In each of these rounds we say that v is *active*.[1]

In this work, we focus on the *weighted sum of completion times* measure. Denote by $f_v(\psi) = \max_{i \in \psi(v)} i$ the largest integer assigned to v by the schedule ψ: this is the round in which the processing of job v is completed. The goal is to find a schedule ψ minimizing $\mathtt{SMC}(G, \psi) = \sum_{v \in V} w_v f_v(\psi)$.

1.1.0.1 *Relation to coloring.* The above problems can also be described as multicoloring problems, where colors are positive integers. A schedule ψ is a multicoloring of G, assigning x_v colors to each vertex v so that adjacent vertices receive non-intersecting sets of colors. In this terminology, the sum of completion times of the jobs equals to the sum of highest colors assigned to the corresponding vertices. Minimizing this sum is known as the *sum multicoloring* problem, which we abbreviate as $\mathtt{SMC}$. When the schedule is required to be non-preemptive we get an instance of *non-preemptive* $\mathtt{SMC}$ ($\mathtt{npSMC}$), and otherwise preemptive $\mathtt{SMC}$ ($\mathtt{pSMC}$). In the special case where all jobs have unit lengths we get an instance of the *sum coloring* ($\mathtt{SC}$) problem.

1.2 Applications

Scheduling conflicting jobs, and the resulting variants of $\mathtt{SMC}$, have numerous applications, in particular, on interval graphs and line graphs. The following practical

---

[1] In all of our solutions, the total number of times in which preemptions take place is polynomial in n; therefore, they admit a strongly polynomial representation.

scenarios yield instances of our problems on these natural classes of graphs.

1.2.0.2 *Session scheduling on a path.* In a path network, pairs of nodes need to communicate, for which they need use of the intervening path. If two paths intersect, the corresponding sessions cannot be held simultaneously. In this case, it would be natural to expect the sessions (i.e., "jobs") to be of different lengths, leading to the SMC problem on interval graphs.

1.2.0.3 *Storage allocation.* Storage allocation in a warehouse involves minimizing the total distance traveled by a robot [Woeginger 1997]. Goods are checked in and out at known times; thus, goods that are not in the warehouse at the same time can share the same location. We represent each of the goods by an interval on the line, which gives the time interval in which it is stored at the warehouse. We number the storage locations by their distance from the counter, and charge each of the goods that distance, being half of the distance traveled by the robot to either store it or fetch it. The sum of the charges for the goods corresponds to scheduling under minsum criteria of the intervals formed by the goods. In particular, minimizing the sum of charges is equivalent to SC on interval graphs.

1.2.0.4 *Scheduling on dedicated processors.* In the dedicated multiprocessor scheduling problem, we are given a set of jobs to be executed on a collection of processors, such that each processor can work on at most one task at a time, and each task needs to be processed simultaneously by a prespecified set of processors without interruption. We consider here the weighted completion time objective, $\sum w_j C_j$. In the commonly used three-field notation of scheduling problems, these problems are denoted by $P|\text{fix}_j = k|\sum_v w_j C_j$, where the second field indicates that each job requires up to k processors. The case of $k = 2$ with biprocessor tasks is equivalent to the npSMC problem in line graphs. More generally, we obtain the npSMC problem in intersection graphs of k-sets (see in Section 6.3).

Examples of multiprocessor tasks include file transfers, which require two corresponding processors simultaneously: the sender and the receiver [Coffman et al. 1985], programs executed on several processors in parallel, which vote for a reliable final result, and mutual testing of processors in biprocessor diagnostic links (see in [Kubale 1996; Giaro et al. 2002]).

More generally, scheduling on dedicated processors is a form of a resource constrained scheduling, where the processors are the main resource. One class of such problems is *data migration* in storage systems. When the objective is to minimize the average completion times of the data files (corresponding to jobs), we obtain the npSMC problem in line graphs. Another similar measure that has also been considered is the sum of completion times of the *resources*, which are in this problem the storage devices. It was observed in [Gandhi et al. 2006] that the problem of minimizing resource completion times in bipartite graphs generalizes the classical open shop scheduling problem under the weighted sum of completion times measure.

Other applications of scheduling conflicting jobs under minsum criteria include traffic intersection control [Bell 1992; Bullock and Hendrickson 1994], wire minimization in VLSI design [Nicoloso et al. 1999], session scheduling in local-area networks [Chen et al. 1993], and compiler design [Nicoloso et al. 1999] (a comprehensive survey appears in [Bar-Noy et al. 2000]). Our results apply also to

permutation graphs, which model, e.g., train scheduling problems.

1.3 Our Results

We present (in Sections 2−4) a general technique for reducing SMC to the classic problem of scheduling conflicting jobs so as to minimize the makespan. Using the technique, we improve (in Sections 5 and 6) the best known results for pSMC and npSMC on several fundamental classes of graphs, including line graphs, $(k + 1)$-claw free graphs and perfect graphs. In particular, we obtain the first constant factor approximation ratio for npSMC on interval graphs, which admits numerous applications. Our improved bound of 7.682 for npSMC of line graphs is achieved by a combination of two simple greedy algorithms within our basic framework (see in Section 6.3), while the previous best ratio of 10 by Kim [2005] involved solving an LP with an exponential number of constraints. We also use the technique to improve (in Section 7) the best known bound for SC on perfect graphs.

It is implicitly assumed in our formulation that the number of machines is "unbounded". This is automatically satisfied in the dedicated scheduling situation by the nature of its formulation. More generally, we show (in Section 8) how the technique can be applied in a system with any given number of machines, with slightly weaker performance ratios.

Finally, we show (in Section 9) how our technique can be applied to handle instances with release times, generalized objective functions and possibly precedence constraints. For the problem of resource constrained scheduling, with the objective of minimizing resource completion times, our technique yields a ratio of $2e \cdot k \approx 5.437k$ to the optimal, thereby improving the best known ratio of $8k - 7$ due to Kim [2005], for any $k \geq 3$.

Table 1 summarizes the new and previous upper and lower-bound results for SC, and upper bounds for pSMC and npSMC, in various classes of graphs. New bounds given in this paper are shown in boldface, with the previous best known bound given in parenthesis. When omitted, [Bar-Noy et al. 1998] is the reference for SC and [Bar-Noy et al. 2000] for SMC. Also, c represents some positive constant. Some of the earlier results apply to restricted subclasses of graphs (e.g., Marx [2003] developed a PTAS for line graphs of trees). A detailed bibliography can be found in [Marx 2004].

    1.3.0.5 *Relation to Min-sum Set Cover.* Our improvement of the previous ratio of 4 for SC on perfect graphs is of particular interest, due to the relation of SC to the *min-sum set cover* problem.

The input to min-sum set cover consists of a universe $\mathcal{U}$ and a collection of subsets $\mathcal{S} = \{S_i\}$, $S_i \subseteq \mathcal{U}$. A feasible solution is an ordered sub-collection of subsets $\mathcal{S}' = \{S'_1, S'_2, \ldots\}$, $\mathcal{S}' \subseteq S$ such that $\bigcup_i S'_i = \mathcal{U}$. We say that $u \in \mathcal{U}$ has cover time i if S'_i is the first subset in the order of $\mathcal{S}'$ to include u. The goal is to minimize the sum of cover times over all the elements of $\mathcal{U}$. Feige et al. [2004] showed that min-sum set cover admits a 4-approximation and that, unless P=NP, for any constant $\epsilon > 0$, there is no $(4 - \epsilon)$-approximation. Observe that SC can be viewed as an instance of the min-sum set cover problem by setting $\mathcal{S}$ as the collection of all independent sets in the conflict graph G. Hence, our 3.591-approximation implies

	SC		SMC	
	u.b.	*l.b.*	`pSMC`	`npSMC`
General graphs	$n/\log^2 n$	$n^{1-\epsilon}$	$n/\log^2 n$	$n/\log n$
Perfect graphs	**3.591** (4)	$c>1$ [BK]	**5.436** (16)	$O(\log n)$
Interval graphs	1.796 [HKS]	$c>1$ [G]	**5.436** (7.184)	**11.273** ($O(\log n)$)
Bipartite graphs	27/26 [M]	$c>1$ [BK]	1.5	2.8
Planar graphs	PTAS [HK]	NPC [HK]	PTAS [HK]	PTAS [HK]
Trees	1 [K]		PTAS [HKP]	1 [HKP]
$k+1$-claw free	k		k	**1.796k^2+.5** ($4k^2$–$2k$) [HKS]
k-sets	k		k	**3.591k+.5** ($6k$–2) [YK]
Line graphs	2	NPC	2	**7.682** (10) [YK]

Table I. Known results for scheduling conflicting jobs under minsum criteria. In the above table [BK] stands for [Bar-Noy and Kortsarz 1998], [G] stands for [Gonen 2001], [HK] stands for [Halldórsson and Kortsarz 2002], [HKP] stands for [Halldórsson, Kortsarz, Proskurowski *et al.*2003], [HKS] stands for [Halldórsson, Kortsarz, and Shachnai 2003], [K] stands for [Kubicka 1989], [YK] stands for [Kim 2005], and [M] stands for [Malafiejski et al. 2004].

that despite this relation, the min-sum set cover problem in its full generality is provably harder to approximate than `SC` on perfect graphs.

1.3.0.6 *Techniques.* Our general approximation technique builds on the framework of Queyranne and Sviridenko [2002b] for scheduling jobs with release times on parallel machines. As in [Queyranne and Sviridenko 2002b], we divide the time line into intervals of geometrically increasing size (see also [Hall et al. 1996; Hall et al. 1997]), using randomized starting points (as introduced in [Chakrabarti et al. 1996]), and approximate the classic makespan problem on each block. Note, however, that the results in [Queyranne and Sviridenko 2002b] do not apply to *arbitrary* conflict graphs. The class of problems studied in [Queyranne and Sviridenko 2002a; 2002b] include shop scheduling (open shop and job shop) and entail a different optimization criteria than `SMC`.

1.4 Related Work

The `SC` problem was introduced by Kubicka [1989] and the `SMC` problems by Bar-Noy et al. [2000].

There is a wide literature on parallel machine scheduling, with the objective of minimizing the sum of completion times/makespan. Most of these works deal with scheduling *independent* jobs, or allow for *precedence constraints* which are directed conflicts. The problem of scheduling *dependent* jobs on m identical machines, so as to minimize the makespan (also called *mutual exclusion scheduling*), is known to be hard to solve, already in the case of unit length jobs, and for graph classes which are easy to color. This includes, e.g., interval graphs and bipartite graphs. A comprehensive survey is given in [Bodlaender and Fomin. 2004]. Afrati et al. [2000] gave a polynomial time approximation scheme for the problem of minimizing the sum of completion times of dedicated tasks (i.e., scheduling dependent jobs on a line graph), in the case where m is a fixed constant.

Some work has been done on resource-constrained scheduling, while the majority has focused on the makespan objective. Hoogeveen et al. [1994] showed that `npSMC` on line graphs of a restricted class of trees is NP-hard in the weak sense. Marx

[2003] showed that `npSMC` of line graphs of trees is strongly NP-hard, even if all tasks have length 1 or 2. Further hardness results on restricted classes are given by Giaro et al. [2002]. Coffman et al. [1985] analyzed the makespan version of `npSMC` of line graphs, under the name of *file transfer problem*. They showed that a class of greedy algorithms yields a 2-approximation and gave a $(2 + \epsilon)$-approximation for a version with more general resource constraints. Kim [2005] gave an LP formulation of the npSMC problem on line graphs and intersection graphs of k-sets, improving the earlier bounds of Halldórsson et al. [2003][2]. She also showed a ratio of $8k - 7$ for the resource completion times minimization problem with k resources.

2. THE TECHNIQUE AND A META-ALGORITHM

Our Meta-algorithm provides a very general tool for scheduling conflicting jobs under the minsum criteria. The algorithm uses as a procedure a makespan algorithm $\mathcal{A}$, which needs to be provably "good" for the meta-algorithm to work well.

In Section 3.1 we describe specific properties for $\mathcal{A}$ that ensure good approximation ratio. In Section 4 we discuss ways to design such a useful makespan algorithm $\mathcal{A}$, which satisfies the required properties.

2.1 Partition into blocks

The basic idea is to divide V into disjoint subsets, i.e., $V = \bigcup_{i=0}^{L} V_i$. We process first the jobs in V_0 to completion, then the jobs in V_1, and so on. The jobs in each V_i must demonstrate some "uniformity" (e.g., being of roughly equal processing time). We elaborate on that below.

Observe that for $i \geq 1$, V_i needs to wait until all the jobs in V_j, $j \leq i-1$, complete before it can start processing. Hence, the job length of V_i should be typically larger than the length of jobs in V_j, $j < i$. More precisely, the jobs in V_i are crucially affected by the *number of rounds* (or makespan) required for processing V_j, $j < i$, as it causes a *delay* in the start time of V_i. It follows that we need a good *makespan* approximation algorithm in order to obtain a good minsum approximation.

We now describe how V is partitioned into the subsets $V_0, V_1, \ldots, V_L$. We assume that a value f_v^* is associated with each job v. V is partitioned into blocks of increasing f_v^* values. Let α be a value chosen uniformly at random from $[0, 1)$ and $\beta > 1$ a constant (to be optimized). Let L be the smallest value such that $\beta^{\alpha+L} \geq \sum_v x_v$. Let $c_i = \beta^{\alpha+i}$, for $i = -1, 0, 1, \ldots, L$. Define $V_i = \{v \in V : c_{i-1} < f_v^* \leq c_i\}$, $i = 0, \ldots, L$, to be the subset of vertices whose f_v^* values fall in the interval $(c_{i-1}, c_i]$.

Remark: Halldórsson and Kortsarz [2002] used a similar method of partitioning V into subsets and fully handling V_j before V_i, $j < i$. However, the partition used $f_v^* = x_v$ and also other types of blocks $[c_i, c_{i+1}]$.[3] In this paper we need to use different f_v^* values; for example, f_v^* may be the optimum fractional solution for a linear programming relaxation of our problem.

---

[2]Henceforth, the reference [Halldórsson et al. 2003] would always stand for the citation [Halldórsson, Kortsarz, and Shachnai 2003].
[3]These blocks are defined throughout the operation of algorithm *Assign Color Sets* presented in [Halldórsson and Kortsarz 2002].

2.2 A meta-algorithm

The (meta-)algorithm can thus be described as follows. We assume an approximation algorithm, $\mathcal{A}$, for the makespan problem on the graph induced by each $V_\ell \subseteq V$. The algorithm uses the values f_v^* and the assumed algorithm $\mathcal{A}$.

Algorithm ALG

(i) Partition the vertices in the graph into blocks $V_0, V_1, \ldots, V_L$, by their f_v^* values.

(ii) Schedule the blocks in sequence using the makespan algorithm $\mathcal{A}$.

---

3. USEFUL PROPERTIES OF $\mathcal{A}$ AND $\{F_V^*\}$

The performance of Algorithm ALG depends on the specific properties of $\mathcal{A}$ and the $\{f_v^*\}$ values. We present three properties which guarantee a "good" approximation.

3.1 The properties

Let OPT be the cost of an optimal solution, and $OPT^* = \sum_v w_v \cdot f_v^*$. In order for the f_v^* values to be useful they have to satisfy two properties. The first requirement is that they give a lower bound on the optimal solution:

(**P1**) $OPT^* \leq OPT$.

For example, this inequality holds if f_v^* is an optimal (fractional) completion time of $v \in V$ in the LP relaxation of our problem.

Recall that V_i has to "wait" for $V_\ell, \ell < i$ to complete before it can start processing. Hence, it is important to process V_ℓ using a good approximation algorithm for the makespan. Considering the lengths x_v as *weights* on the vertices, define $\omega(V_\ell, x)$ as the weight of a maximum weight clique in the graph induced by V_ℓ. It is easy to see that $\omega(V_\ell, x)$ is a lower bound on the optimum makespan for V_ℓ. Let $d \geq 1$ be a parameter. We require that:

(**P2**) $\max_{v \in V_\ell} f_v^* \geq \dfrac{\omega(V_\ell, x)}{d}$, for all $0 \leq \ell \leq L$.

Intuitively, property (P2) tells us that when considering (only) V_ℓ, the maximum fractional finishing time given to vertices is *not much smaller* than the obvious lower bound on the makespan: the value of the largest weighted clique. Observe that if, say, the f_v^* values are obtained from the solution of a linear program, such a property is not obvious, since the goal in the LP is to minimize the *sum* of f_v^* values. The parameter d tells us how well the f_v^* values relate to the clique size.

Thus, if (P2) applies, the clique size is closely related to the f_v^* values. However, we also need to assume that we are able to find a solution of low makespan:

(**P3**) The algorithm $\mathcal{A}$ has makespan bounded by $\rho \cdot \omega(V_\ell, x)$, for some constant ρ. Namely,

$$\mathcal{A}(V_\ell, x) \leq \rho \cdot \omega(V_\ell, x), \text{ for } \ell = 0, 1, \ldots, L. \tag{1}$$

Observe that (P3) is a stronger requirement than asking for a performance ratio of ρ to the optimum makespan of V_ℓ. Our approximation ratios will be proportional to d and ρ.

3.2 The approximation ratio based on (P1)-(P3)

We analyze the approximation ratio of algorithm ALG assuming (P1)-(P3). We first obtain a per-vertex bound of the expected completion time $\tilde{f}_v$ of each vertex v under our algorithm schema ALG.

THEOREM 3.1. *For each vertex $v \in V$, $\mathbf{E}[\tilde{f}_v] \leq d \cdot \rho \cdot \dfrac{\beta}{\ln \beta} f_v^*$.*

PROOF. Recall that $c_\ell = \beta^{\alpha+\ell}$, for $\ell = -1, 0, \ldots, L$, where $c_L \geq \sum_v x_v$ and $V_\ell = \{v \in V : c_{\ell-1} < f_v^* \leq c_\ell\}$, $\ell = 0, \ldots, L$. Denote by ℓ_v the block into which vertex v falls (as a function of α).

The completion of vertex v is bounded above by the sum of the makespans of all blocks up to and including its own block, V_{ℓ_v}. By properties (P2) and (P3), the makespan on each block V_ℓ is bounded by $\mathcal{A}(V_\ell, x) \leq \rho \cdot \omega(V_\ell, x) \leq \rho \cdot dc_\ell$. Hence, we get for each vertex independently that

$$\tilde{f}_v \leq \sum_{r=0}^{\ell_v} \mathcal{A}(V_r, x) \leq \frac{d \cdot \rho \beta^{\alpha+\ell_v+1}}{\beta - 1} = d\rho \frac{\beta}{\beta - 1} c_{\ell_v}. \tag{2}$$

Recall that we select α uniformly at random from $[0, 1)$. Then ℓ_v and c_{ℓ_v} are also random variables. The theorem now follows from the following lemma. □

LEMMA 3.2. *For any $\beta > 1$ and $v \in V$, $\mathbf{E}[c_{\ell_v}] = \dfrac{\beta - 1}{\ln \beta} f_v^*$, where the expectation is over the random choices of α.*

PROOF. Let $z = \log_\beta f_v^*$, so $f_v^* = \beta^z$. Recall that ℓ_v is the smallest integer such that $\alpha + \ell_v \geq z$. Since z and α are both non-negative, so is ℓ_v. Then, $\ell_v = \lceil z - \alpha \rceil = -\lfloor \alpha - z \rfloor$. Define $y_v = \ell_v + \alpha - z$ and note that y_v is the fractional part of $\alpha - z$. Namely, $y_v = \alpha - z - \lfloor \alpha - z \rfloor = (\alpha - z) \bmod 1$. Note that z is a fixed value, independent of α. Thus, since α is uniformly distributed on $[0, 1)$, so is y_v. The random variable β^{y_v} then has expected value

$$\mathbf{E}[\beta^{y_v}] = \int_0^1 \beta^t dt = \frac{\beta - 1}{\ln \beta}.$$

Hence,

$$\mathbf{E}[c_{\ell_v}] = \mathbf{E}[\beta^{\ell_v+\alpha}] = \mathbf{E}[\beta^{\ell_v+\alpha-z}] \cdot \beta^z = \frac{\beta - 1}{\ln \beta} f_v^*.$$

□

Recall that the cost of the algorithm is $\mathsf{ALG}(V, x) = \sum_v w_v \tilde{f}_v$, while the cost of the lower bound to the optimum is $OPT^* = \sum_v w_v f_v^*$. Using linearity of expectation, we have by Theorem 3.1 that

$$\mathbf{E}[\mathsf{ALG}(V, x)] = \sum_v w_v \mathbf{E}[\tilde{f}_v] \leq d\rho \frac{\beta}{\ln \beta} OPT^*. \tag{3}$$

The function $f(\beta) = \beta/\ln \beta$ is minimized when $\beta = e \approx 2.718$. This gives the following:

THEOREM 3.3. ALG *with an algorithm $\mathcal{A}$ and $\{f_v^*\}$ values that satisfy (P1)-(P3) yields a $(d \cdot e\rho)$-approximation for the corresponding minsum problem.*

4. OBTAINING USEFUL F_V^* VALUES VIA LP

One way to obtain the f_v^* values is by solving the LP relaxation of an integer programming formulation of the problem. (Such LP relaxations have been used in the past in scheduling *independent* jobs; see, e.g., [Wolsey 1985; Queyranne 1987; 1993; Schulz 1996; Hall et al. 1996]).

Let $C_1, \ldots, C_N$ be the set of all maximal cliques in G. For a subset U of vertices, let $x(U) = \sum_{u \in U} x_u$. The following integer program is valid both for the preemptive and non-preemptive case.

$$(LP_1) \ \text{minimize} \ \sum_v w_v f_v$$

$$\text{subject to:} \quad \forall i \text{ and } \forall C \subseteq C_i \quad : \sum_{v \in C} x_v f_v \geq \frac{(x(C))^2 + \sum_{v \in C} x_v^2}{2} \qquad (4)$$

$$\forall v \in V \qquad : f_v \in \{1, 2, \ldots\}$$

The constraints (4) follow from the requirement that the vertices (=jobs) in any clique C are processed in *disjoint* sets of rounds. Thus, if the vertices in C are scheduled in the order $v_1, v_2, \ldots, v_t$, then $f_{v_j} \geq \sum_{r=1}^{j} x_r$. In the linear relaxation of LP_1, we allow f_v to take non-integral values ≥ 1. Let f_v^* denote the value of f_v in an optimal (fractional) solution for LP_1. We show below how to implement this linear program more efficiently, and in most cases with a polynomial-size program.

The next lemma shows that LP_1 satisfies property (P2) with $d = 2$. It is based on a result of Kim [2005] (Lemma 2.3), attributed to Hall et al. [1996].

LEMMA 4.1. *For any* $1 \leq \ell \leq L$, $\max_{v \in V_\ell} f_v^* \geq \frac{\omega(V_\ell, x)}{2}$.

PROOF. Let C_ℓ be a maximum weight clique in V_ℓ, and let v_ℓ be the vertex in C_ℓ with the largest completion time in V_ℓ, $f_{v_\ell}^*$. From the constraints (4), we have that $\sum_{u \in C_\ell} x_u f_u \geq x(C_\ell)^2/2 = \omega(V_\ell, x)^2/2$. We also have that $\sum_{u \in C_\ell} x_u f_u \leq f_{v_\ell}^* \sum_{u \in C_\ell} x_u = f_{v_\ell}^* x(C_\ell) = f_{v_\ell}^* \omega(V_\ell, x)$. $\square$

Since $\max_{v \in V_\ell} f_v^* \leq c_\ell$, this implies that $c_\ell \geq \omega(V_\ell, x)/2$, for $\ell = 0, 1, \ldots, L$.

4.1 An Efficient Implementation

Note that, since the number of constraints in (4) is exponential, LP_1 can be solved in polynomial time only on restricted sets of graphs (e.g., graphs in which the number of maximal cliques is polynomial). We now show that an optimal fractional solution can be obtained by using an alternative formulation of the linear program, which can be solved efficiently on the classes of graphs that we study here. For some of these graph classes (e.g. line graphs, interval graphs), the linear program becomes of polynomial size. We can formulate pSMC (as well as npSMC) as an integer program that uses *linear ordering* variables (see, e.g. in [Potts 1980; Hall et al. 1996]). For each edge $uv \in E$, there is a variable $\delta_{uv} \in \{0, 1\}$, such that $\delta_{uv} = 1$ if u precedes v in the schedule, and 0 otherwise. Let $N(v)$ denote the set of neighbors of v in G. We denote by $C_1, \ldots, C_{N_v}$ the set of maximal cliques in $N(v)$.

$$(LP) \quad \text{minimize} \ \sum_{v \in V} w_v f_v$$

$$\text{subject to:} \quad \forall v \in V, \ \forall r, 1 \le r \le N_v \ : \ f_v \ge x_v + \sum_{u \in C_r} x_u \delta_{uv} \tag{5}$$

$$\forall uv \in E \qquad \qquad : \ \delta_{uv} + \delta_{vu} = 1$$

In the linear relaxation of LP we allow $\delta_{uv} \ge 0$. We now show that the fractional solution obtained for LP is a feasible solution for LP_1.

LEMMA 4.2. *The optimal completion times obtained from the solution of the program LP satisfy the constraints in (4).*

PROOF. Let C be a clique in G. Let f_v be the completion time of $v \in C$ in the solution for LP. Indeed, $C \setminus \{v\} \subseteq N(v)$. From (5), we get that

$$
\begin{aligned}
\sum_{v \in C} x_v f_v &\ge \sum_{v \in C} x_v \left(x_v + \sum_{u \in C, u \ne v} x_u \delta_{uv} \right) \\
&= \sum_{v \in C} x_v^2 + \sum_{u,v \in C} (x_v x_u \delta_{uv} + x_v x_u \delta_{vu}) \\
&= \frac{1}{2} \sum_{v \in C} x_v^2 + \sum_{u \in C} x_u \sum_{v \in C} x_v \\
&= \frac{1}{2} \sum_{v \in C} x_v^2 + x(C)^2.
\end{aligned}
$$

□

Combining Lemmas 4.1 and 4.2, we get the following.

COROLLARY 4.3. *The values of f_v in the optimal solution for LP satisfy property (P2) with $d = 2$.*

5.   APPLICATIONS FOR PREEMPTIVE SCHEDULING

In this section we use the developed tools to derive approximations for pSMC on perfect graphs. Applying our technique, we first find a fractional solution for the LP of value OPT^*. We note that on perfect graphs, the number of constraints in LP may be exponential. Yet, it can be solved in polynomial time, since we have a polynomial time separation oracle: given an optimal solution for LP, we can test in polynomial time whether all the constraints are satisfied.

A separation oracle is obtained as follows. Given a solution for LP and a vertex $v \in V$, we set for each vertex $u \in N(v)$, $x_u' = x_u \delta_{uv}$. We can now find a maximum weight clique in $N(v)$ with respect to the x'-values (since any subgraph of G is perfect). Then, we can test in polynomial time whether f_v satisfies the constraint (5), by checking whether the inequality holds for this maximum weight clique. If the inequality does not hold, then it provides a violated constraint. Hence, LP can be solved in polynomial time.[4]

The solution for LP yields a (possibly infeasible) schedule that satisfies (P1) and (P2) with $d = 2$. The problem of preemptive scheduling dependent jobs to minimize the makespan (denoted by pMC, from the relation to preemptive multicoloring) is

---

[4]For more details, see e.g., in [Queyranne 1993].

solvable in polynomial time on perfect graphs, within arbitrary desired precision, as shown in [Grötschel et al. 1993] (also known as *weighted coloring*). Thus, $\rho = 1 + o(1)$, and using Theorem 3.3, we improve on the previous best factor of 16 [Bar-Noy et al. 2000].

THEOREM 5.1. ALG *gives a* $2e + o(1) \approx 5.436$-*approximation for* pSMC *on perfect graphs.*

This also improves on the previous 7.184-ratio for interval graphs [Halldórsson et al. 2003].

6.   THE NON-PREEMPTIVE CASE

6.1   Improving the bound of Theorem 3.3

In the non-preemptive case, $\mathcal{A}$ must be an algorithm for *non-preemptive* scheduling to minimize makespan. Assuming that, we may use the schedule output by algorithm $\mathcal{A}$ for V_ℓ either directly or reversed. In the reverse order, vertices scheduled in round i by $\mathcal{A}$ on V_ℓ will be scheduled in round $\mathcal{A}(V_\ell, x) - i + 1$. Job v completed at time f in forward order will then be completed at time $\mathcal{A}(V_\ell, x) - f + x_v$ in the reverse order, and the average of the two is $(\mathcal{A}(V_\ell, x) + x_v)/2$. We can select the order that yields the better weighted average for the jobs within V_ℓ, or choose one at random. Thus, the expected completion time of vertex v, over the two orderings of its block, is bounded by

$$
\begin{aligned}
\tilde{f}_v &\le \sum_{r=0}^{\ell-1} \mathcal{A}(V_r, x) + \frac{\mathcal{A}(V_\ell, x)}{2} + \frac{x_v}{2} \\
&\le d \cdot \rho \left(\frac{\beta^{\alpha+\ell}}{2} + \sum_{r=0}^{\ell-1} \beta^{\alpha+r} \right) + \frac{x_v}{2} \qquad (6) \\
&\le d \cdot \rho \beta^{\alpha+\ell} \left(\frac{1}{2} + \frac{1}{\beta-1} \right) + \frac{x_v}{2} \\
&= d \cdot \rho \cdot c_\ell \left(\frac{\beta+1}{2(\beta-1)} \right) + \frac{x_v}{2} \qquad (7)
\end{aligned}
$$

Using Lemma 3.2, the expected completion time of v, over the random choices of α, is bounded by

$$
\mathbf{E}[\tilde{f}_v] \le d \cdot \rho \frac{\beta+1}{2 \ln \beta} f_v^* + \frac{x(v)}{2}.
$$

Hence,

$$
\begin{aligned}
\mathbf{E}[\mathsf{ALG}(V, x)] &= \sum_v w_v \mathbf{E}[\tilde{f}_v] \le d \cdot \rho \frac{\beta+1}{2 \ln \beta} OPT^* + \frac{\sum_v w_v x_v}{2} \\
&\le \left(d\rho \frac{\beta+1}{2 \ln \beta} + \frac{1}{2} \right) OPT^* \qquad (8)
\end{aligned}
$$

The function $f(\beta) = (\beta+1)/\ln \beta$ is minimized when $\beta = \gamma \approx 3.59112$, for a ratio of $d\gamma\rho/2 + 0.5$. We summarize in the next result.

THEOREM 6.1. *There is a* $(1.796 d\rho + 0.5)$-*approximation algorithm for* npSMC.

6.2 An alternative for obtaining the f_v^* values

An alternative way of obtaining the infeasible solution f_v^* in the non-preemptive case is to use the preemptive solution when solving the non-preemptive problem. In this case, we replace (P2) and (P3) by the following properties.

(**P2′**) There is a d-approximation algorithm for pSMC, for some $d \geq 1$.
(**P3′**) There is a non-preemptive scheduling algorithm, $\mathcal{A}$, that approximates the makespan of any graph in the given graph class within a ρ factor of the number of rounds used by an optimal preemptive schedule, i.e.,

$$\mathcal{A}(V_\ell, x) \leq \rho \cdot \texttt{pMC}(V_\ell, x), \text{ for } \ell = 0, 1, \ldots, L. \tag{9}$$

We now summarize the steps of the algorithm, based on the approximation for pSMC. The algorithm gets as parameters the values β, α.

Algorithm ALG$_{\text{PRE}}$

(i) Apply to G a d-approximation algorithm for pSMC. Let f_v^{pre} be the completion time of $v \in V$. Set for each $v \in V$, $f_v^* = f_v^{pre}/d$.

(ii) Partition the vertices into blocks $V_0, V_1, \ldots$ by their f_v^* values.

(iii) Schedule the blocks in sequence using a non-preemptive makespan algorithm $\mathcal{A}$.

Theorem 6.1 now applies (with the same parameters) here as well.

6.3 Approximation Results

6.3.0.7 *Line graphs.* Here we can apply both the LP and the preemptive relaxations with equal performance ratio, but the latter approach is more efficient. A greedy 2-approximation algorithm for pSMC on line graphs is presented in [Bar-Noy et al. 2000] (that holds also in the weighted case). Thus, (P2′) is satisfied, and we can apply algorithm ALG$_{\text{PRE}}$, with $d = 2$.

For non-preemptive scheduling to minimize makespan on line graphs, we can use the greedy algorithm of [Coffman et al. 1985], which schedules each job as early as possible, breaking ties arbitrarily. This ensures that each vertex is always waiting for a neighbor until it is scheduled to completion. The completion time of a vertex is then at most the sum of the lengths of its neighbors, which is bounded by twice the length of the larger clique involving the vertex. Thus, in this case, we have $\rho = 2$, and using Theorem 6.1, we get:

THEOREM 6.2. *There is a 7.68224-approximation algorithm for* npSMC *on line graphs.*

This improves on the factor of 10 by Kim [2005] and on the factor of 12 given by a combinatorial (greedy) algorithm of Halldórsson et al. [2003]. Recall that this implies the same ratio for the biprocessor scheduling problem $P|\text{fix}_j = 2| \sum w_j C_j$.

Observe that the non-preemptive algorithms are all measured in terms of the preemptive optimum, so the ratio may actually be better.

6.3.0.8 *Intersection graphs of k-sets.* Resource-bounded scheduling, where each job uses at most k resources, is modeled by intersection graphs of sets of size at

most k. These are the clique graphs of hypergraphs in which each element occurs in at most k sets. For each resource r, the vertices using that resource form a clique C_r. Then, for any $v \in V$, $N(v)$ can be partitioned into at most k maximal cliques.

We can extend the LP-based strategy for line graphs to intersection graphs of k-sets. In this case, the non-preemptive greedy makespan scheduling algorithm of Coffman et al. [1985] uses at most $k\omega$ rounds, where ω is the maximal size of any of the resource cliques. Note that it suffices to consider only cliques induced by individual resources, and not those cliques formed by interplay of a collection of resources. In other words, the clique constraints in LP need only involve the resource-cliques, therefore the number of constraints is polynomial. This gives a non-preemptive solution with $d = 2$ and $\rho = k$, and by Theorem 6.1, we get

THEOREM 6.3. *There is a $(3.591k + 0.5)$-approximation algorithm for* npSMC *on intersection graphs of k-sets.*

This improves on the ratio of $6k - 2$ of Kim [2005].

6.3.0.9 *$(k+1)$-claw free graphs.* The combinatorial strategy for line graphs can be generalized for $(k + 1)$-claw free graphs, albeit with a ratio function worse than the ratio obtained by the LP-based algorithm for intersection graphs of k-sets. The sorted greedy algorithm of Bar-Noy et al. [2000] yields a ratio of k for pSMC in $(k + 1)$-claw free graphs, resulting in a preemptive relaxation with $d = k$ in our schema. Also, as above, the makespan algorithm has performance ratio $\rho = k$. Thus, we get:

THEOREM 6.4. *There is a combinatorial $(1.796k^2 + 0.5)$-approximation algorithm for* npSMC *on $(k + 1)$-claw free graphs.*

6.3.0.10 *Interval graphs.* The nondeterministic makespan problem, npMC, on interval graphs is better known as *dynamic storage allocation*. Gergov gave an algorithm whose makespan is at most $3\omega(G)$ [Gergov 1999]. Hence, we have an algorithm $\mathcal{A}$ with $\rho = 3$. The number of maximal cliques in an interval graph is at most n. Thus, LP has a polynomial number of constraints and we can use it to obtain a schedule satisfying (P1) and (P2), with $d = 2$. We can also use the approximation of the preemptive solution of Halldórsson et al. [2003] as a relaxation with $d = 7.184$. Applying Theorem 6.1, we obtain the first constant approximation factor for this problem.

THEOREM 6.5. *There is an 11.273-approximation algorithm and a combinatorial 38.7-approximation algorithm for* npSMC *on interval graphs.*

7. SCHEDULING UNIT LENGTH JOBS

For sum coloring (SC) problems, where vertices have unit length, we obtain a slight improvement of the results in Theorem 6.1.

THEOREM 7.1. *There is a $1.796d\rho$-approximation algorithm for* SC.

PROOF. Continuing from (6), we have

$$\tilde{f}_v \ \leq \ d\rho \left(\frac{\beta^{\alpha+\ell_v}}{2} + \sum_{r=0}^{\ell_v-1} \beta^{\alpha+r} \right) + \frac{1}{2} x_v$$

$$= d\rho \left(\frac{\beta^{\alpha+\ell_v}}{2} + \beta^\alpha \frac{\beta^{\ell_v} - 1}{\beta - 1} \right) + \frac{1}{2}$$

$$= d\rho \left(c_{\ell_v} \frac{\beta + 1}{2(\beta - 1)} - \frac{\beta^\alpha}{\beta - 1} \right) + \frac{1}{2}$$

Thus, the expected completion time of vertex v is bounded by

$$\mathbf{E}[\tilde{f}_v] \leq d\rho \left(\frac{\beta + 1}{2 \ln \beta} f_v^* - \frac{1}{\ln \beta} \right) + \frac{1}{2} \leq d\rho \frac{\beta + 1}{2 \ln \beta} f_v^*,$$

using that in all reasonable scenarios we would choose β to be less than e^2. Hence,

$$\mathbf{E}[\mathsf{ALG}(V, x)] = \sum_{v \in V} w_v \mathbf{E}[\tilde{f}_v] \leq d\rho \cdot \frac{\beta + 1}{2 \ln \beta} OPT^*.$$

Setting $\beta = \gamma \approx 3.59112$ yields the theorem. □

In particular, we obtain a ratio of 3.59112 for perfect graphs. Namely, there is an optimal coloring algorithm (makespan) for perfect graph [Grötschel et al. 1993] (satisfying (P3) with $\rho = 1 + o(1)$), and as argued earlier, we can obtain a solution for LP which satisfies properties (P1) and (P2) with $d = 2$. Recall that this result is to be contrasted with the lower bound of 4 for the min-sum set cover problem [Feige et al. 2004], that can model SC problem, albeit, by a reduction that is not always polynomial.

8. SCHEDULING DEPENDENT JOBS ON IDENTICAL PARALLEL MACHINES

In the following we describe how our technique can be applied for scheduling a set of n dependent jobs on m identical machines. As before, we get as input the conflict graph G of the jobs. The problem of minimizing the sum of completion times can be now formulated as the following integer program.

$$(LP_1(m)) \quad \text{minimize} \quad \sum_v w_v f_v$$

subject to:

$$\forall S \subseteq V : \ \sum_{v \in S} x_v f_v \geq \frac{(x(S))^2 + \sum_{v \in S} x_v^2}{2m} \tag{10}$$

$$\forall i \text{ and } \forall C \subseteq C_i : \ \sum_{v \in C} x_v f_v \geq \frac{(x(C))^2 + \sum_{v \in C} x_v^2}{2} \tag{11}$$

$$\forall v \in V : \ f_v \in \{1, 2, \ldots\}$$

For a subset of vertices $S \subseteq V$ and a vertex $v \in S$, we denote by $P_v(S)$ the set of vertices in S whose processing is completed no later than f_v in the solution for $LP_1(m)$; that is, $P_v(S) = \{u \in S \mid f_u \leq f_v\}$. Thus, we have

$$\sum_{v \in S} x_v f_v \geq \frac{1}{m} \sum_{v \in S} x_v \sum_{u \in P_v(S)} x_u.$$

In the linear programming relaxation, we allow $f_v \geq 1$. An optimal solution $\{f_v^*\}$ of the program $LP_1(m)$ satisfies the next lemma, due to Hall et al. [1996]).

LEMMA 8.1. *For any $v \in V$ and a subset of vertices $S \subseteq V$, $f_v^* \geq \dfrac{x(P_v(S))}{2m}$.*

As before, we can replace $LP_1(m)$ by the following program

$$(LP(m)) \quad \text{minimize} \sum_{v \in V} w_v f_v$$

$$\text{subject to:} \qquad \forall S \subseteq V: \qquad \sum_{v \in S} x_v f_v \geq \frac{(x(S))^2 + \sum_{v \in S} x_v^2}{2m} \qquad (12)$$

$$\forall v \in V,\ 1 \leq r \leq N_v: \ f_v \geq x_v + \sum_{u \in C_r} x_u \delta_{uv}$$

$$\forall uv \in E: \qquad \delta_{uv} + \delta_{vu} = 1 \qquad (13)$$

We note that on the classes of graphs that we study, $LP(m)$ can be solved in polynomial time. This follows from the fact that, given a vector of fractional values for the variables, we can use the separation algorithm of Queyranne [1993] to test whether all the constraints in (12) are satisfied; the other set of constraints may be either of polynomial size, or exponential, in which case we apply the separation algorithm described in Section 5. We now describe our algorithm schema, ALG_m, distinguishing between the preemptive and non-preemptive case.

8.0.0.11 *Preemptive scheduling.* In the preemptive case, we solve $LP(m)$ and partition the time axis, as before, to the intervals $(c_{\ell-1}, c_\ell]$. For each $\ell \geq 0$, we schedule V_ℓ using a ρ-approximation algorithm $\mathcal{A}$ for the preemptive makespan problem, pMC. That is, we initially assume that we have an *unbounded* number of machines. We then 'fix' the preemptive schedule of V_ℓ by partitioning each subset of independent jobs I_g (that are scheduled to run simultaneously at $(t, t+1]$) to $\lfloor |I_g|/m \rfloor$ sets of size m, and at most one set of size smaller than m. By that, we ensure that at most m jobs are processed at any given time. Let t_ℓ be the total number of rounds used after we fix the schedule of V_ℓ. We first upper bound t_ℓ. The fixed schedule has rounds corresponding to the makespan schedule of V_ℓ under $\mathcal{A}$, and additional rounds in which all m machines are busy. Note that if the initial number of rounds is t_ℓ', then in the worst case, the remaining set of jobs for each independent set, I_g, is of size one. Hence, using properties (P2) and (P3), we get that

$$t_\ell \leq t_\ell' + \frac{x(V_\ell) - t_\ell'}{m} \leq \rho\,\omega(V_\ell, x)\left(1 - \frac{1}{m}\right) + \frac{x(V_\ell)}{m} \leq d \cdot \rho \beta^{\alpha + \ell}\left(1 - \frac{1}{m}\right) + \frac{x(V_\ell)}{m}. \quad (14)$$

Let $V_\ell^- = \cup_{r=0}^\ell V_r$ denote the set of jobs scheduled up to (and including) the ℓ-th block, and let $\tilde{f}_v$ be the completion time of v under ALG_m. Then,

$$\tilde{f}_v \leq \sum_{r=0}^{\ell_v} t_r \leq d \cdot \rho c_{\ell_v} \frac{\beta}{\beta - 1}\left(1 - \frac{1}{m}\right) + \frac{x(V_{\ell_v}^-)}{m} \qquad (15)$$

By Lemma 8.1,

$$\frac{x(V_\ell^-)}{m} \leq 2 \cdot \max_{v \in V_\ell} f_v^* \leq 2c_{\ell_v}.$$

Randomizing on α and using Lemma 3.2, we have that

$$\mathbf{E}[\mathsf{ALG}_m(V,x)] = \sum_v \mathbf{E}[\tilde{f}_v] \leq \left(d \cdot \rho \frac{\beta}{\ln \beta}(1 - \frac{1}{m}) + \frac{2(\beta - 1)}{\ln \beta} \right) OPT^*, \qquad (16)$$

and taking $\beta = e$, we get the next result.

THEOREM 8.2. *There is a $(d \cdot e\rho(1 - \frac{1}{m}) + 4.436)$-approximation algorithm for the problem of preemptive minsum of completion times scheduling of dependent jobs on m identical machines, where d is as given in (P2) and ρ is the approximation ratio of algorithm A for* pMC.

In particular, for the class of perfect graphs, we solve $LP(m)$, and then apply the makespan algorithm of Grötschel et al. [1993], which gives a ratio arbitrarily close to 1 for unbounded number of machines. Hence, we have:

COROLLARY 8.3. *There is a $(5.436(1 - \frac{1}{m}) + 3.436 + o(1))$-approximation algorithm for the problem of preemptive minsum of completion times scheduling of dependent jobs on m identical machines for perfect graphs.*

8.0.0.12   *Non-preemptive scheduling.* In the non-preemptive case, it may not be possible to 'fix' the schedule of V_ℓ, i.e., transform a schedule with 'unbounded' number of machines to one that uses at most m machines at any time. Thus, when scheduling the jobs in V_ℓ, we assume that $\mathcal{A}$ is an algorithm for the makespan problem on m machines. For deriving an approximation ratio for the non-preemptive case, we slightly modify the properties of the minsum solution and the makespan algorithm used in ALG_m; that is, we replace (P2) and (P3) by the following properties.

(P2(m)) For all $1 \leq \ell \leq L$, $\max_{v \in V_\ell} f_v^* \geq \max(\omega(V_\ell, x), x(V_\ell)/m)/d$.

(P3(m)) The makespan of algorithm $\mathcal{A}$ satisfies

$$\mathcal{A}(V_\ell, x) \leq \rho \cdot \max(\omega(V_\ell, x), x(V_\ell)/m), \text{ for } \ell = 0, 1, \ldots, L. \qquad (17)$$

We note that, from Lemmas 4.1 and 8.1, it follows that the f_v^* values output by $LP(m)$ satisfy property (P2(m)), with $d = 2$.

THEOREM 8.4. *There is a $(1.796d\rho + 0.5)$-approximation algorithm for the problem of non-preemptive minsum of completion times scheduling, where d and ρ are given in (P2(m)) and (P3(m)), respectively.*

PROOF. By properties (P2(m)) and (P3(m)), for any $0 \leq \ell \leq L$,

$$t_\ell \leq \rho \max(\omega(V_\ell, x), x(V_\ell)/m) \leq \rho d \cdot c_\ell.$$

Note that we apply here ALG_m with possible reverse of the schedule. (We decide on reversing the schedule for each machine separately.) Hence, the analysis of Section 6.1 holds, and we obtain the same ratio as in Theorem 6.1. □

Consider, for example, the class of line graphs. Here, we can use the greedy algorithm of Coffman et al. [1985], allowing at most m jobs to be processed at any time. Then, each job waits either for a neighbor (that is being processed), or for one of the machines to become available. Thus, property (P3(m)) is satisfied with $\rho = 3$.

COROLLARY 8.5. *There is an* 11.276*-approximation algorithm for the problem of non-preemptive minsum of completion times scheduling on line graphs, in a system of m identical machines.*

For intersection of k-sets and $(k+1)$-claw free graphs, the algorithm of [Coffman et al. 1985] satisfies property (P3(m)) with $\rho = k+1$. Hence, we have

COROLLARY 8.6. *There is a* $(3.591(k+1)+0.5)$*-approximation algorithm for the problem of non-preemptive minsum of completion times scheduling of the intersection of k-sets, on m identical machines.*

For $(k+1)$-claw free graphs, we can apply the sorted greedy algorithm of Bar-Noy et al. [2000], which yields the ratio $(k+1)$ for pSMC with bounded number of machines. Thus, we have

COROLLARY 8.7. *There is a combinatorial* $(1.796(k+1)^2 + 0.5)$*-approximation algorithm for* npSMC *on m identical machines, where the conflict graph is $(k+1)$-claw free.*

9. EXTENSIONS

9.0.0.13 *Release times.* Our technique can be applied also in the case where each job v has a release time, r_v. In this case, we add the constraint $f_v \geq r_v + x_v$, for each vertex v, to the LP formulation. By setting $\beta = 2$, we can ensure that the block V_{ℓ_v} containing vertex v does not start to get executed until time $\sum_{i < \ell_v} c_i = c_{\ell_v} \geq r_v$. Thus, using (3) and (8) with $\beta = 2$, we get

THEOREM 9.1. ALG *attains a ratio of* $1.5d\rho/\ln 2 + \frac{1}{2} \approx 2.16d\rho + \frac{1}{2}$ *for* npSMC *and $d\rho2/\ln 2 \approx 2.89d\rho$ for* pSMC *instances with release times.*

9.0.0.14 *Order-constrained schedules.* When the input contains precedence constraints (i.e., some of the edges in G are *directed*), or for some other reason we are constrained in the way we order the jobs, we may not be able to use the improvement in the non-preemptive case obtained by possibly reversing the order of jobs within a block (as applied in Section 6.1). We can then trivially bound the completion time of vertex v by the number of rounds used up to and including the block in which v is processed (as we did in the preemptive case). We naturally assume that the makespan algorithm in question can handle these order constraints. We then get:

THEOREM 9.2. *There is a $(d \cdot e\rho)$-approximation algorithm for order constrained scheduling.*

9.0.0.15 *General objective functions.* We can handle considerably more general objectives function. Suppose we are given a collection $\mathcal{S}$ of subsets of vertices, along with their weights. The completion time of a set S in $\mathcal{S}$ is the largest completion time of a vertex in S. The objective is now to minimize the weighted sum of completion times of the sets. As before, for any $v \in V$, we denote by $C_1, \ldots, C_{N_v}$ the set of maximal cliques in $N(v)$. The linear relaxation of the problem is the following linear program.

$$\text{(LP-g)} \quad \text{minimize} \quad \sum_{S \in \mathcal{S}} w_S f_S$$

$$\text{subject to:} \quad \forall v \in V, \ \forall r, 1 \le r \le N_v \ : \ f_v \ge x_v + \sum_{u \in C_r} x_u \delta_{uv}$$

$$\forall S \in \mathcal{S}, \ \forall v \in S \qquad : \ f_S \ge f_v \tag{18}$$

$$\forall uv \in E \qquad\qquad : \ \delta_{uv} + \delta_{vu} = 1 \tag{19}$$

This can, e.g., involve makespan, by including in $\mathcal{S}$ the single set $S = V$. Also, the jobs could be divided into groups, each with a different "owner", and it may be desirable to bound the average completion times of the owners of the jobs.

In our analysis of the preemptive case (Thm. 3.1), and order-constrained version of the non-preemptive case, we considered each vertex independently and separately, bounding its expected cost only by the last round used in its block. Thus, similar arguments will give us a bound matching Theorem 3.3.

Let w denote the vertex of largest f^* value in S. From (2),

$$\max_{v \in S} \tilde{f}_v \le \max_{v \in S} d\rho \frac{\beta}{\beta - 1} c_{\ell_v} = d\rho \frac{\beta}{\beta - 1} c_{\ell_w}.$$

Now, by Lemma 3.2, $\mathbf{E}[c_{\ell_w}] = \frac{\beta - 1}{\ln \beta} f_w^*$, giving a bound on the expected completion time of set S of

$$\mathbf{E}[\max_{v \in S} \tilde{f}_v] \le d \cdot e\rho \max_{v \in S} f_v^*.$$

THEOREM 9.3. *There is a $(d \cdot e\rho)$-approximation algorithm for* npSMC *and/or* pSMC *under generalized objective functions.*

Recall that in resource-constrained scheduling, the resources are represented as cliques in the conflict graph G. We seek to minimize the average completion time of the resources (cliques). This is a case of a generalized objective function, where the sets are given by the resource cliques.

COROLLARY 9.4. ALG *attains an approximation ratio of $2e \cdot k$ for the weighted resource completion times measure in a resource-constrained scheduling with up to k resources per job.*

This improves on the previous ratio of $8k - 7$ presented by Kim [2005], for any $k \ge 3$. For $k = 2$, the ratio of 10.88 is worse than the best known approximation ratio of 5.055 [Gandhi et al. 2006]; however, it applies to more general objective functions, and is achieved by solving a polynomial-size linear program.

9.0.0.16 *Derandomization.* Note that all the above schemes can be derandomized, by partitioning the interval $[0, 1)$ to smaller intervals; we can then search for the best value of α in these intervals to within desired precision. In particular, to bound the error in the approximation ratio by $(1 + \epsilon)$, we need only to break the interval $[0, 1)$ into a number of intervals whose quantity depends on ϵ. For any constant $\epsilon > 0$, this only adds a multiplicative constant to the running time. See [Halldórsson et al. 2003] for a detailed analysis in the sum coloring case, and [Queyranne and Sviridenko 2002a] on related scheduling problems.

paper.

REFERENCES

AFRATI, F., BAMPIS, E., FISHKIN, A., JANSEN, K., AND KENYON, C. 2000. Scheduling to minimize the average completion time of dedicated tasks. In *Proceedings of the Twentieth Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 454–464.

BAR-NOY, A., BELLARE, M., HALLDÓRSSON, M. M., SHACHNAI, H., AND TAMIR, T. 1998. On chromatic sums and distributed resource allocation. *Information and Computation 140*, 183–202.

BAR-NOY, A., HALLDÓRSSON, M. M., KORTSARZ, G., SHACHNAI, H., AND TAMIR, T. 2000. Sum multicoloring of graphs. *Journal of Algorithms 37*, 422–450.

BAR-NOY, A. AND KORTSARZ, G. 1998. The minimum color-sum of bipartite graphs. *Journal of Algorithms 28*, 339–365.

BELL, M. 1992. Future directions in traffic signal control. *Transportation Research Part A 26*, 303–313.

BODLAENDER, H. L. AND FOMIN., F. V. 2004. Equitable colorings of bounded treewidth graphs. In *Proceedings of the Twenty-Ninth International Symposium on Mathematical Foundations of Computer Science*. Springer, 180–190.

BULLOCK, D. AND HENDRICKSON, C. 1994. Roadway traffic control software. *IEEE Transactions on Control Systems Technology 2*, 255–264.

CHAKRABARTI, S., PHILLIPS, C. A., SCHULZ, A. S., SHMOYS, D. B., STEIN, C., AND WEIN, J. 1996. Improved scheduling problems for minsum criteria. In *Proceedings of the Twenty-Third International Colloquium on Automata, Languages, and Programming*. Springer, 646–657.

CHEN, J., CIDON, I., AND OFEK, Y. 1993. A local fairness algorithm for gigabit lans/mans with spatial reuse. *IEEE Journal on Selected Areas in Communications 11*, 1183–1192.

COFFMAN, E. G., GAREY, M. R., JOHNSON, D. S., AND LAPAUGH, A. S. 1985. Scheduling file transfers. *SIAM Journal on Computing 14, 3*, 744–780.

FEIGE, U., LOVÁSZ, L., AND TETALI, P. 2004. Approximating min-sum set cover. *Algorithmica 40, 4*, 219–234.

GANDHI, R., HALLDÓRSSON, M. M., KORTSARZ, G., AND SHACHNAI, H. 2006. Improved results for data migration and openshop scheduling. *ACM Transactions on Algorithms 2, 1*, 116–129.

GERGOV, J. 1999. Algorithms for compile-time memory allocation. In *Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 907–908.

GIARO, K., MALAFIEJSKI, M., KUBALE, M., AND PIWAKOWSKI, K. 2002. Dedicated scheduling of biprocessor tasks to minimize mean flow times. In *Proceedings of the Fourth International Conference on Parallel Processing and Applied Mathematics*. Springer, 87–96.

GONEN, M. 2001. Coloring problems on interval graphs and trees. M.Sc. Thesis, School of Computer Science, The Open University, Tel-Aviv.

GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. 1993. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag.

HALL, L., SCHULZ, A. S., SHMOYS, D. B., AND WEIN, J. 1997. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research 22*, 513–544.

HALL, L., SHMOYS, D. B., AND WEIN, J. 1996. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. In *Proceedings of the Seventh ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 142–151.

HALLDÓRSSON, M. M. AND KORTSARZ, G. 2002. Tools for multicoloring with applications to planar graphs and partial k-trees. *Journal of Algorithms 42, 2*, 334–366.

HALLDÓRSSON, M. M., KORTSARZ, G., AND SHACHNAI, H. 2003. Sum coloring interval graphs and k-claw free graphs with applications for scheduling dependent jobs. *Algorithmica 37*, 187–209.

HOOGEVEEN, H., VAN DER VELDE, S. L., AND VELTMAN, B. 1994. Complexity of scheduling multi-processor tasks with prespecified processor allocations. *Discrete Applied Mathematics 55*, 259–272.

KIM, Y. 2005. Data migration to minimize the average completion time. *Journal of Algorithms 55*, 42–57.

KUBALE, M. 1996. Preemptive versus non-preemptive scheduling of biprocessor tasks on dedicated processors. *European Journal of Operations Research 94*, 242–251.

KUBICKA, E. 1989. The chromatic sum of a graph. Ph.D. Thesis, Western Michigan University.

MALAFIEJSKI, M., GIARO, K., JANCZEWSKI, R., AND KUBALE, M. 2004. Sum coloring of bipartite graphs with bounded degree. *Algorithmica 40,* 4, 235–244.

MARX, D. 2003. Minimum sum multicoloring on the edges of trees. In *Proceedings of the First Workshop on Approximation and Online Algorithms*. Springer, 214–226.

MARX, D. 2004. Chromatic sum and minimum sum multicoloring. `http://www.cs.bme.hu/~dmarx/sum.html`.

NICOLOSO, S., SARRAFZADEH, M., AND SONG, X. 1999. On the sum coloring problem on interval graphs. *Algorithmica 23*, 109–126.

POTTS, C. N. 1980. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Stud. 13*, 78–87.

QUEYRANNE, M. 1987. Polyhedral approaches to scheduling problems. Seminar presented at RUCOR, Rutgers University.

QUEYRANNE, M. 1993. Structure of a simple scheduling polyhedron. *Mathematical Programming 58*, 263–285.

QUEYRANNE, M. AND SVIRIDENKO, M. 2002a. A $(2 + \epsilon)$-approximation algorithm for generalized preemptive open shop problem with minsum objective. *Journal of Algorithms 45*, 202–212.

QUEYRANNE, M. AND SVIRIDENKO, M. 2002b. Approximation algorithms for shop scheduling problems with minsum objective. *Journal of Scheduling 5*, 287–305.

SCHULZ, A. S. 1996. Scheduling to minimize total weighted completion time: Performance guarantees of lp-based heuristics and lower bounds. In *Proceedings of the Fifth International Conference on Integer Programming and Combinatorial Optimization*. Springer, 301–315.

WOEGINGER, G. 1997. Private Communication.

WOLSEY, L. 1985. Mixed integer programming formulations for production planning and scheduling problems. Invited talk at the Twelfth International Symposium on Mathematical Programming, MIT, Cambridge.