Approximating Minimum Sum Coloring with Bundles

Seyed Parsa Darbouy

Department of Computing Science, University of Alberta, Canada

Zachary Friggstad[1]

Department of Computing Science, University of Alberta, Canada

Abstract

In the MINIMUM SUM COLORING WITH BUNDLES problem, we are given an undirected graph $G = (V, E)$ and (not necessarily disjoint) bundles $V_1, V_2, \ldots, V_p \subseteq V$ with associated weights $w_1, \ldots, w_p \geq 0$. The goal is to give a proper coloring of G using positive integers to minimize the weighted average/total completion time of all bundles, where the completion time of a bundle is the maximum integer assigned to one of its nodes. This is a common generalization of the classic MINIMUM SUM COLORING problem, i.e. when all bundles are singleton nodes, and the classic CHROMATIC NUMBER problem, i.e. the only bundle is all of V.

Despite its generality as an extension of MINIMUM SUM COLORING, only very special cases have been studied with the most common being the line graph $L(H)$ of a graph H (also known as COFLOW SCHEDULING). We provide the first constant-factor approximation in perfect graphs and, more generally, graphs whose chromatic number is within a constant factor of the maximum clique size in any induced subgraph. For example, we obtain constant-factor approximations for graphs that are well-studied in minimum sum coloring such as interval graphs and unit disk graphs.

Next, we extend our results to get constant-factor approximations for a general model where the bundles are disjoint (i.e. can be thought of as jobs brought by the corresponding client) and we are only permitted to color/schedule a bounded number of jobs from each bundle at any given time. Specifically, we get constant-factor approximations for this model if the nodes of graph G have an ordering $v_1, v_2, \ldots, v_n$ such that the *left-neighborhood* $N_\ell(v_i) := \{v_j : j < i, v_i v_j \in E\}$ can be covered by $O(1)$ cliques. For example, this applies to chordal graphs, unit disc graphs, and circular arc graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Scheduling algorithms

Keywords and phrases Approximation Algorithms, Scheduling, Coloring

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.21

Funding *Zachary Friggstad*: Supported by an NSERC Discovery Grant and an Accelerator Supplement.

1 Introduction

The MINIMUM SUM COLORING (**MSC**) problem is a well-studied problem lying at the intersection of scheduling theory and graph coloring. In it, we are given an undirected graph $G = (V, E)$ on n nodes. The goal is to find a **proper coloring** $\chi : V \to \{1, 2, \ldots\}$ of nodes of V, i.e. $\chi(u) \neq \chi(v)$ for all $uv \in E$, with minimum total color $\sum_{v \in V} \chi(v)$. Viewed as a scheduling problem, this models settings where unit-length jobs may be completed in parallel but resource conflicts prevent certain pairs of jobs from being completed at the same time.

While different than standard graph coloring where the goal is to simply minimize the number of distinct colors of the coloring, it is essentially just as hard to approximate. That is, unless $\mathbf{P} = \mathbf{NP}$, there is no $n^{1-\delta}$-approximation for any constant $\delta > 0$ [2]. However, in

[1] Corresponding Author

19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024).
Editor: Hans L. Bodlaender; Article No. 21; pp. 21:1–21:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

certain cases, it is possible to get improved approximations. For example, if it is possible to efficiently compute a maximum independent set in G and any of its induced subgraphs then greedily coloring by computing a maximum independent set I, coloring I with the next unused integer, and then removing I from G yields a 4-approximation [2]. There is a large body of work on getting improved constant-factor approximation in more structured special cases or obtaining constant-factor approximations in other graph classes, see the related works section below for further discussion.

In this work, we extend this model in two ways. The first extension is given as follows.

▶ **Definition 1.** *In the* Minimum Sum Coloring with Bundles *problem (***MSCB***), we are given an undirected graph $G = (V, E)$ and collection of bundles $V_1, \dots, V_p \subseteq V$ with associated weights $w_1, \dots, w_p \geq 0$. The goal is to find a proper coloring $\chi : V \to \{1, 2, \dots, |V|\}$ of V in a way that minimizes the total weighted makespan of all bundles. That is, the coloring should minimize $\sum_{i=1}^{p} w_i \cdot \max_{v \in V_i} \chi(v)$.*

If one views bundles as being associated with clients, then each client is interested in the time it takes for their bundle to be complete. The objective of **MSCB** is then equivalent to minimizing the average weighted completion time of all clients' bundles. If all bundles are singletons, then **MSCB** is the same as **MSC**. If the partition has only a single bundle, i.e. $p = 1$, then **MSCB** is the same as computing the chromatic number of G. Thus, **MSCB** is a common generalization of both problems.

It should be noted that **MSCB** has been studied previously in certain special cases. For example, when G is the line graph of an undirected graph the problem is known as Coflow Scheduling. See Section 1.1 for a discussion of this and other related problems.

Many techniques that are successful for **MSC** fail in **MSCB**. For example, while iteratively computing a maximum independent set in G to color the nodes is a 4-approximation for **MSC** [2], it could be as bad as an $\Omega(\log n)$-approximation for computing the chromatic number even in trees (see Appendix A). Further difficulties in designing **MSCB** approximations will be discussed in Section 1.3.

Our first main result is a constant-factor approximation for **MSCB** in perfect graphs. Recall a graph G is **perfect** if the maximum clique size in $G[U]$ equals the chromatic number of $G[U]$ for any $U \subseteq V$ where $G[U] = (U, \{uv \in E : u, v \in U\})$ denotes the subgraph of G induced by U. Examples of perfect graphs include bipartite graphs, line graphs of bipartite graphs, interval graphs, comparability graphs, split graphs, permutation graphs, and chordal graphs as well as the edge-complements of these graphs. For a broader discussion of perfect graphs, we refer the reader to Golumbic's excellent book [8].

▶ **Theorem 2.** **MSCB** *on perfect graphs admits a polynomial-time* 10.874*-approximation.*

Our presentation will first focus on proving Theorem 2 in the case of interval graphs. These are graphs $G = (V, E)$ where each $v \in V$ is associated with an interval $[s_v, t_v] \subseteq \mathbb{R}$ and we have an edge $uv \in E$ whenever the corresponding intervals intersect, i.e. $[s_u, t_u] \cap [s_v, t_v] \neq \emptyset$. The linear programming (LP) relaxation we use in this case is simpler than in the general case of perfect graphs. After presenting the algorithm for interval graphs, we discuss the few necessary changes to extend it to perfect graphs in general.

Our techniques extend further to classes of graphs for which the chromatic number is approximately equal to the maximum clique number and these quantities can be approximated in polynomial time. Namely, we get the following extension. Recall a graph class $\mathcal{G}$ is **hereditary** if for any $G \in \mathcal{G}$ we have $G[U] \in \mathcal{G}$ as well for all induced subgraphs of G.

▶ **Corollary 3.** *Let $\mathcal{G}$ be any hereditary graph class with the following properties: (a) the maximum clique size is within a constant factor of the chromatic number of any $G \in \mathcal{G}$, (b) given vertex weights $w_v, v \in V$ for a graph $G \in \mathcal{G}$, there is a constant-factor approximation algorithm for the maximum-weight clique of G, and (c) there is a constant-factor approximation for coloring any $G \in \mathcal{G}$ with the minimum number of colors. Then there is a constant-factor approximation for* **MSCB** *for graphs in $\mathcal{G}$.*

For example, we get $O(1)$-approximations for the following graph classes:

- Unit disc graphs: when vertices are associated with discs of radius 1 in the plane and edges indicate when two discs intersect. The classic algorithm in [4] for computing a maximum-size clique easily generalizes to compute a maximum-weight clique in polynomial time. It is possible to color any unit disc graph with maximum clique size K using fewer than $3 \cdot K$ colors in polynomial time [13].
- Circular-arc graphs: when vertices are associated with arcs of a circle and edges indicate when two such arcs intersect. A maximum weight clique can be found in polynomial time, e.g. [3]. One can efficiently $2 \cdot K$-color a circular arc graph with maximum clique size K by first coloring the arcs spanning one particular point with at most K colors. After removing these arcs, we are left with an interval graph which can also be colored with K additional colors since interval graphs are perfect.

Our general definition of **MSCB** allows bundles to overlap. A natural special case is where the bundles constitute a partition $V_1, \ldots, V_p$ of V, i.e. each client brings their own jobs V_i to be processed. In this case, it is natural to imagine that clients themselves are somewhat limited on how quickly they can interact with the service provider that is processing the jobs. For example, perhaps clients can only deliver or take away a bounded number of jobs from the processing center at any given time. We consider the following generalization of **MSCB**

▶ **Definition 4.** *In the* MINIMUM SUM COLORING WITH BUNDLES AND TASK CONCURRENCY *problem (***MSCB-TC***), we are given a graph $G = (V, E)$ and a* **partition** *$V_1, \ldots, V_p$ of V with associated weights $w_1, \ldots, w_p \geq 0$. Further, for each $1 \leq i \leq p$ we are given a bound $d_i \geq 1$ on the number of jobs from bundle i that may be processed at any moment. The goal is still to find a proper coloring χ that minimizes the total weighted completion time of all bundles, but we further require $|\{v \in V_i | \chi(v) = t\}| \leq d_i$ for each $1 \leq i \leq p$ and each time step/color t.*

To the best of our knowledge, this extension of **MSCB** has not been previously studied even in special cases. We obtain constant-factor approximations for this problem, though in more restricted settings. Recall that a graph G is chordal if every cycle of length ≥ 4 has a chord. That is, if $v_1, v_2, \ldots, v_\ell, v_1$ is a cycle of length $\ell \geq 4$ then we also have $v_i v_j \in E$ for some $1 \leq i, j \leq \ell$ where i, j are not consecutive indices around the cycle, i.e. $1 \leq i \leq j - 1$ and if $i = 1$ then $j \neq \ell$. Interval graphs, an important class of graphs in scheduling, are chordal.

▶ **Theorem 5.** **MSCB-TC** *in chordal graphs admits a polynomial-time 16.31-approximation.*

The key property of chordal graphs that drives our algorithm is that they admit **perfect elimination orderings**. That is, it is possible to compute an ordering $v_1, v_2, \ldots, v_n$ of all nodes V such that the left-neighborhood $N_\ell(v_i) := \{v_j : j < i \text{ and } v_i v_j \in E\}$ is a clique for all $1 \leq i \leq n$. In fact, a graph is chordal if and only if it admits such an ordering and this ordering can be computed in linear time [8]. Our techniques extend more generally to other classes of graphs.

▶ **Corollary 6.** *Let $\mathcal{G}$ be a hereditary graph class with the same properties as in Corollary 3. Further, for any $G \in \mathcal{G}$ suppose we can compute an ordering $v_1, \ldots, v_n$ of its nodes in polynomial time such that the left-neighborhood $N_\ell(v_i)$ can be covered by a constant number of cliques in polynomial time. Then* **MSCB-TC** *has a constant-factor when restricted to inputs whose graphs lie in $\mathcal{G}$.*

For example, such an ordering exists for unit disk graphs (with each left-neighborhood being covered by ≤ 3 cliques) [13]. Such an ordering can be also found for circular arc graphs with each left-neighborhood being covered by ≤ 2 cliques, i.e. consider the coloring algorithm mentioned above: if one orders the arcs spanning the selected point and then orders the remaining arcs according to a perfect elimination ordering in the resulting interval graph.

The algorithm from Theorem 5 requires one additional structural result about coloring than the algorithm from Theorem 2, namely Lemma 10 in Section 3. Unfortunately this structural result fails to hold in perfect graphs, which is why Theorem 5 is only for chordal graphs. Still, we are able to recover the following.

▶ **Theorem 7.** *There is an $O(\sqrt{n})$-approximation for* **MSCB-TC** *in perfect graphs.*

While the ratio is quite large, it is at least better than the lower bound in general graphs of $n^{1-\delta}$ for any constant $\delta > 0$, which is inherited from the same hardness for **MSC** [2]. Perhaps it is possible to design a constant-factor approximation for **MSCB-TC** in perfect graphs, we leave this as an open problem.

1.1 Related Work

MSCB has been studied in certain special cases. The most notable example is COFLOW SCHEDULING, which is equivalent to **MSCB** when the input graph is the line graph[2] of a bipartite graph (i.e. at any given time step a matching of the edges/jobs is scheduled). This problem was first introduced in [14] where a 22.34-approximation was given. Later, 4-approximations followed for COFLOW SCHEDULING and generalizations with release times for the jobs [1, 16, 6].

In a matroid setting, the jobs are given as items in the ground set X of a matroid rather than as vertices in a graph and bundles are subsets of items in X. The set of jobs scheduled at any given time must form an independent set of the matroid. A 2-approximation for the problem of minimizing the total weighted completion time of all bundles was given in [12].

MSC itself is much more well studied. As mentioned earlier, a 4-approximation is known in settings where the maximum independent set of the graph (and any induced subgraph) can be computed in polynomial time [2]. Special attention has been given to particular graph classes, in particular a 1.796-approximation is known in interval graphs [10] which was recently extended to an algorithm with the same approximation guarantee for chordal graphs [5]. In the more general setting of perfect graphs, a 3.591-approximation is known [7]. A broader summary of approximation algorithms for **MSC** in special graph classes can be found in the survey article by Halldórsson and Kortsarz [9].

1.2 Organization

After discussing some challenges in adapting previous **MSC** and COFLOW SCHEDULING algorithms to **MSCB**, Section 2 presents our algorithm for **MSCB** and the proofs of Theorem 2 and Corollary 3. Section 3 extends these techniques to **MSCB-TC** and proves Theorem 5

[2] Recall the line graph of $G = (V, E)$ is the graph $L(G) = (E, F)$ where two edges $e, f \in E$ are considered adjacent in $L(G)$ if they share a common endpoint.

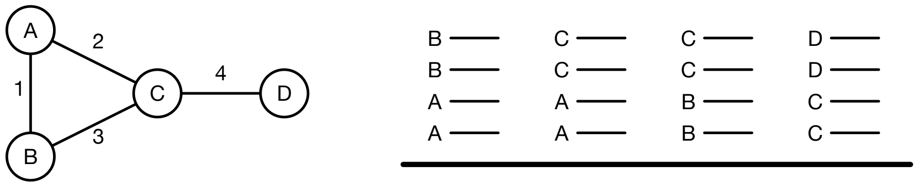

Figure 1 An example of the reduction from an instance of the MAXIMUM INDEPENDENT SET problem to a collection of intervals using $t = 2$. It is straightforward to verify a subset of nodes is independent if and only if each point on the underlying line is spanned by at most t intervals in the union of their bundles.

and Corollary 6. We also discuss why our **MSCB-TC** algorithm does not extend to perfect graphs in general, point out that we can at least get an $O(\sqrt{n})$-approximation in perfect graphs for **MSCB-TC**, and leave further improvements for future work.

1.3 Challenges

While **MSCB** is a common generalization of **MSC** and COFLOW SCHEDULING, it turns out most techniques used to address these problems fail. For example, a 4-approximation for **MSC** follows by iteratively finding a maximum independent set in the graph of unscheduled tasks in each time step. But for the classic problem of computing the chromatic number of a graph, i.e. the special case of **MSCB** where all nodes are in a single bundle, this can be as bad as an $\Omega(\log n)$-approximation even if the underlying graph is a tree, see Appendix A for a simple example.

Another strategy that is used to get refined approximations for **MSC** is to compute maximum t-colorable subgraphs of G for a geometric sequence of values for t as in [10, 5]. For **MSCB**, one could consider an algorithm that for a geometric sequence of values t will compute a maximum-size subset of bundles $\mathcal{P} \subseteq \{1, 2, \ldots, p\}$ such that all nodes in these bundles, i.e. $\cup_{j \in \mathcal{P}} V_j$ can be scheduled without conflict in t steps. That is, we do not just compute a maximum-size t-colorable induced subgraph of G itself, rather we are concerned with how many clients can have their bundles completed within t steps.

Unfortunately, even in the special case where G is an interval graph, this seems impossible to approximate well.

▶ **Lemma 8.** *Let $G = (V, E)$ be an interval graph and $V_1, V_2, \ldots, V_p$ a partition of V. For any $t \leq |V|$ and any constant $\delta > 0$, there is no $O(|V|^{1/3-\delta})$-approximation algorithm for computing the maximum size $\mathcal{P} \subseteq \{1, 2, \ldots, p\}$ such that $\cup_{j \in \mathcal{P}} V_j$ can be scheduled within t steps unless* **P** $=$ **NP**.

Proof. We reduce from the MAXIMUM INDEPENDENT SET problem. Let $H = (U, F)$ be an undirected graph. Order F arbitrarily as $e_1, e_2, \ldots, e_{|F|}$. We build an interval graph over the line $[1, 2 \cdot |F|]$. Namely, for each $v \in U$ and each e_i having v as an endpoint, add t copies of the interval $[2 \cdot i - 1, 2 \cdot i]$. Let U_v denote all intervals created from $v \in U$ this way. Note, there are exactly $2 \cdot t$ intervals of the form $[2 \cdot i - 1, 2 \cdot i]$ for each $1 \leq i \leq |F|$, one for each endpoint of e_i. Let G be the resulting interval graph whose nodes/intervals are partitioned as $\{U_v : v \in U\}$. Figure 1 illustrates this reduction.

Let $\mathcal{I}$ be a subset of bundles. It is straightforward to verify the intervals in $\cup_{U_v \in \mathcal{I}} U_v$ can be t-colored (i.e. can be scheduled within t time steps) if and only if $\{v \in U : U_v \in \mathcal{I}\}$ is an independent set in H.

Finally, recall for any constant $\delta > 0$, there is no $|U|^{1-\delta}$-approximation for the maximum independent set problem unless $\mathbf{P} = \mathbf{NP}$ [17]. Since $t \leq n$, then G has $2 \cdot |F| \cdot t \leq O(|U|^3)$ vertices, as required. Thus, an $\alpha = o(|V|^{1/3-\delta/3})$-approximation for the largest number of parts that can be scheduled in t time steps would yield a $o(|U|^{1-\delta})$-approximation for the maximum independent set problem in H. Replacing δ by 3δ (again a constant) yields the result. ◀

Since **MSC** techniques seem to fail for **MSCB**, one could look to techniques successfully used for approximating Coflow Scheduling. Recall Coflow Scheduling is the special case of **MSCB** when the graph G is the line graph $L(H)$ of an undirected graph. A property of Coflow Scheduling that is commonly leveraged to design approximation algorithms is that for each edge e in H (i.e. a job), all other edges that conflict with e share one of two endpoints with e.

For example, the algorithm in [1] solves a time-indexed LP relaxation and uses the familiar trick of greedy scheduling jobs according to their fractional completion time. Their analysis relies on the fact that there are only 2 "reasons" an edge may not be scheduled at a time step (i.e. one of the two endpoints already has an edge at that time step). Also, in [6] a hypergraph matching result by Haxell [11] is used to demonstrate that a good schedule of jobs exists. However, the way this matching result is used in [6] crucially relies on the fact that the formed hyperedges only have 3 nodes, which comes from the fact that each edge of H has only two endpoints.

2 Approximating MSCB in Perfect Graphs

Despite the challenges pointed out in Section 1.3, we are still able to design a constant-factor approximation for **MSCB** in perfect graphs. Our techniques can be seen as a workaround to the problem highlighted in Lemma 8, though we also need to use LPs to do so. That is, we use an LP-relaxation to fractionally schedule the jobs. For a geometrically-increasing sequence of values t, we consider the jobs that are completed to an extent of at least $1/2$ by time t in the fractional solution. The LP constraints will witness that the size of the largest clique among these jobs is $O(t)$. The fact the graph is perfect then allows us to color these jobs with $O(t)$ colors, i.e. to schedule them all within $O(t)$ time steps.

For simplicity of presentation, we suppose $G = (V, E)$ is an interval graph with $n = |V|$ nodes and that $V = \{V_1, V_2, \ldots, V_p\}$. This allows us to write a polynomial-size LP relaxation. The straightforward extension to perfect graphs (albeit with an exponential-size LP) will be discussed at the end of this section. Thus, we suppose each vertex $v \in V$ is associated with an interval $[s_v, t_v] \subseteq \mathbb{R}$. It is a folklore result that we may further assume, without loss of generality, that each endpoint of each interval lies in the set $\{1, 2, \ldots, 2 \cdot n\}$. For each $1 \leq i \leq 2 \cdot n$ the set $C_i := \{v \in V : i \in [s_v, t_v]\}$ is easily seen to be a clique in G. It is also known [8] that every clique of G is a subset of C_i for some $1 \leq i \leq 2 \cdot n$.

Like some previous work in **MSC** [5] and Coflow Scheduling [1, 6], we consider a time-indexed LP relaxation for **MSCB**. For each $v \in V$ and $1 \leq t \leq n$ we let $x_{v,t}$ be a variable indicating v should be colored t and for each $1 \leq k \leq p$ we let f_k be a variable that is intended to be the largest color used to color nodes in V_k (i.e. when all jobs for bundle k are completed). Throughout this section, we adhere to the following indexing conventions: k will be used for bundles, t for time steps/colors, i for points on the underlying line $\{1, 2, \ldots, 2 \cdot n\}$ in the interval graph, and j for indexing geometric groupings of jobs defined in the algorithm.

$$\begin{array}{rrcll} \textbf{minimize}: & \sum_{k=1}^{p} w_k \cdot f_k & & & \\ \textbf{subject to}: & f_k & \geq & \sum_{t=1}^{n} t \cdot x_{v,t} & \forall\, 1 \leq k \leq p, v \in V_k \\ & \sum_{v \in C_i} x_{v,t} & \leq & 1 & \forall\, 1 \leq t \leq n, \forall\, 1 \leq i \leq 2 \cdot n \\ & \sum_{t=1}^{n} x_{v,t} & = & 1 & \forall\, v \in V \\ & x, f & \geq & 0 & \end{array} \qquad (\textbf{LP-MSCB})$$

The first constraint says that bundle k is considered finished only after each $v \in V_k$ is completed, and the second constraint ensures that at most one vertex from any clique in the interval graph can be processed at a time. The third ensures each job is completed at some point. Clearly, the optimum value of (**LP-MSCB**) is at most the optimum value of the **MSCB** instance since the natural integer solution corresponding to the optimal **MSCB** solution is feasible for this LP.

2.1 Rounding Algorithm

After computing an optimal solution to **LP-MSCB**, for each job $v \in V$ we let τ_v denote the smallest time t such that $\sum_{t' \leq t} x_{v,t'} \geq 1/2$, i.e. when the LP solution has completed v to an extent of at least $1/2$. Notice by minimality of τ_v we also have $\sum_{t' \geq t} x_{v,t'} = 1 - \sum_{t' < t} x_{v,t'} > 1 - 1/2 = 1/2$.

For a bundle k, we then let $\hat{f}_k := \max_{v \in V_k} \tau_v$ be the minimum time when all jobs in V_k are completed to an extent of at least $1/2$ by the LP solution. As we show below, it is not hard to see $\sum_k \hat{f}_k$ is within a constant factor of the optimal LP solution.

We then employ geometric grouping of the jobs $v \in V$. That is, for each time t in a geometric sequence we form a group with all jobs v having $\tau_v \leq t$. Using properties of the LP solution and interval graphs, we show we can properly color all jobs in each such group with $2 \cdot t$ colors. Concatenating these schedules for the various groups in this geometric sequence completes the algorithm.

To optimize our final ratio, we carefully choose the geometric growth rate and also pick an initial random geometric offset, as has been done in many previous works in minimum-latency problems, e.g. [10]. We could also try a different parameter than $1/2$ as the choice of threshold for the defining values τ_v, but it turns out that $1/2$ is the optimal value for our approach. Also, readers with experience in **MSC** algorithms may wonder about another optimization. Namely, with **MSC** once one has colored a geometric group one may get a slight improvement in the approximation guarantee by optimally ordering the colors so that larger color classes are finished earlier. However, this optimization does not work in our setting since we are concerned with the completion times of bundles and reordering color classes within a group's coloring may not affect the completion time of a bundle.

We let $q > 1$ be a constant. It turns out the optimal setting for q in our algorithm is just e, the base of the natural logarithm. We leave q as an unspecified constant for now and only set it at the point in the analysis where it is apparent that this was the best choice of constant. The precise description of our rounding technique is presented in Algorithm 1.

2.2 Analysis

Recall the sets U_j described in Algorithm 1.

▷ Claim 9. For each j, the jobs in U_j can be scheduled without any conflicts using at most $2 \cdot q^{j+\alpha}$ time steps.

Algorithm 1 **MSCB** Scheduling.

Compute an optimal solution (x, f) to (**LP-MSCB**).
Set τ_v to the smallest integer such that $\sum_{t \leq \tau_v} x_{v,t} \geq 1/2$ for each $v \in V$.
Sample $\alpha \sim [0, 1)$ uniformly.
Let $U_j = \{v \in V : \lfloor q^{j-1+\alpha} \rfloor < \tau_v \leq \lfloor q^{j+\alpha} \rfloor\}$ for $0 \leq j \leq \log_q n$.
for each U_j in increasing order of j **do**
  Schedule all jobs in U_j within the next $2 \cdot \lfloor q^{j+\alpha} \rfloor$ unused time steps. $\{$Claim 9$\}$

Proof. We claim the size of the largest clique contained in U_j is at most $2 \cdot q^{j+\alpha}$. If so, then we can properly color all of U_j using at most $2 \cdot q^{j+\alpha}$ colors because interval graphs are perfect.

First, consider any $v \in U_j$ and say $v \in V_k$. Then $\tau(v) \leq \hat{f}_k \leq q^{j+\alpha}$, so

$$\sum_{t \leq q^{j+\alpha}} x_{v,t} \geq \sum_{t \leq \tau(v)} x_{v,t} \geq \frac{1}{2}.$$

Now consider any point $i \in \{1, 2, \ldots, 2 \cdot n\}$ on the interval, our goal is to show $|U_j \cap C_i| \leq 2 \cdot q^{j+\alpha}$. Letting $X_{i,j} := \sum_{v \in U_j \cap C_i} \sum_{t \leq q^{j+\alpha}} x_{v,t}$, summing the above bound over $v \in |U_j \cap C_i|$ shows $X_{i,j} \geq |U_j \cap C_i|/2$.

On the other hand, by the LP constraints we also have

$$X_{i,j} = \sum_{t \leq q^{j+\alpha}} \sum_{v \in U_j \cap C_i} x_{v,t} \leq \sum_{t \leq q^{j+\alpha}} \sum_{v \in C_i} x_{v,t} \leq \sum_{t \leq q^{j+\alpha}} 1 = q^{j+\alpha}.$$

From these two bounds on $X_{i,j}$ we have $|U_j \cap C_i|/2 \leq X_{i,j} \leq q^{j+\alpha}$ so $|U_j \cap C_i| \leq 2 \cdot q^{j+\alpha}$.

Finally, consider any clique $C \subseteq U_j$. Any clique of G is contained in a clique of the form C_i so $C \subseteq U_j \cap C_i$. Thus, we have $|C| \leq |U_j \cap C_i| \leq 2 \cdot q^{j+\alpha}$, that is the bound holds for all cliques $C \subseteq U_j$. ◁

Next, we show each bundle k finishes within time $O(\hat{f}_k)$. Fix any such bundle k and pick any $v_k \in V_k$ with $\tau_{v_k} = \hat{f}_k$. For any value α sampled by the algorithm, the completion time of bundle k is upper bounded by the completion time of all jobs in the bundle U_j that contains v_k. This is because no job of V_k will be placed in a bundle $U_{j'}$ having $j' > j$ and because we concatenated the schedules for the various buckets in increasing order of j.

Since $0 \leq \alpha \leq 1$ then there is some integer j_k such that $v_k \in U_{j_k - 1}$ or $k \in U_{j_k}$, depending on the value of α. The breaking point between these two events occurs at $\alpha = \log_q \hat{f}_k - (j_k - 1)$. Letting T_α be the quantity $2 \cdot q^{j+\alpha}$ for the group $j \in \{j_k - 1, j_k\}$ that v_k is assigned to for a given α, we have:

$$T_\alpha \leq \begin{cases} 2 \cdot q^{j_k - 1 + \alpha} & k \in U_{j_k - 1} \\ 2 \cdot q^{j_k + \alpha} & k \in U_{j_k} \end{cases}$$

Since we concatenate the schedules for the groups U_j in increasing order of index j and each group U_j is completed by time $2 \cdot q^{j+\alpha}$ (Claim 9), then for any j each job in U_j will be completed by time $\sum_{j'=1}^{j} 2 \cdot q^{j'+\alpha} \leq \frac{2 \cdot q}{q-1} \cdot q^{j+\alpha}$. Therefore we have

$$\begin{aligned}
\mathbf{E}_{\alpha\sim[0,1)}[T_\alpha] &= \int_0^1 T_\alpha \, d\alpha \\
&= \frac{2 \cdot q}{q-1} \cdot \left(\int_0^{\log_q \hat{f}_k - (j_k-1)} q^{j_k+\alpha} \, d\alpha + \int_{\log_q \hat{f}_k - (j_k-1)}^{1} q^{j_k-1+\alpha} \, d\alpha \right) \\
&= \frac{2 \cdot q}{q-1} \cdot \left(\left. \frac{q^{j_k+\alpha}}{\ln q} \right|_0^{\log_q \hat{f}_k - (j_k-1)} + \left. \frac{q^{j_k-1+\alpha}}{\ln q} \right|_{\log_q \hat{f}_k - (j_k-1)}^{1} \right) \\
&= \frac{2 \cdot q}{\ln q} \cdot \tau_{v_k} = \frac{2 \cdot q}{\ln q} \cdot \hat{f}_k
\end{aligned}$$

At this point, we see the optimal choice of q is $e \approx 2.717$, the base of the natural logarithm. Setting q to e yields

$$\mathbf{E}_{\alpha\sim[0,1)}[T_\alpha] = \frac{2 \cdot q}{\ln q} \cdot \hat{f}_k = 2 \cdot e \cdot \hat{f}_k$$

We complete the proof by bounding f_k by $O(\hat{f}_k)$. Recalling $\sum_{t \geq \tau_{v_k}} x_{v_k,t} \geq 1/2$, we see

$$f_k \geq \sum_t t \cdot x_{v_k,t} \geq \sum_{t \geq \tau_{v_k}} t \cdot x_{v_k,t} \geq \tau_v \cdot \sum_{t \geq \tau_{v_k}} x_{v_k,t} \geq \frac{\tau_{v_k}}{2} = \frac{\hat{f}_k}{2}$$

To put this all together, by Claim 9 the completion time of a bundle k is at most T_α. In expectation over the random choice of α, this is at most $2 \cdot e \cdot \hat{f}_k$. Finally, from the bound directly above we see the expected completion time of a bundle is then at most $4 \cdot e \cdot f_k$. Thus, the expected total completion time of all bundles is at most $4 \cdot e \leq 10.874$ times the optimum value of (**LP-MSCB**).

2.3 Extensions

Perfect Graphs. The only change to the LP is that the second collection of constraints is replaced by the following more general constraints:

$$\sum_{v \in C} x_{v,t} \leq 1 \quad \forall\, t, \forall \text{ cliques } C \text{ of } G \tag{1}$$

In general there are exponentially many cliques (and even exponentially-many maximal cliques) in a perfect graph. Still, these constraints can be separated in polynomial time for perfect graphs (Theorem 67.6 in [15]) meaning the LP can still be solved optimally in polynomial time.

The rest of the proof carries through essentially without modification: the size of a maximum clique in U_j is still bounded to be at most $2 \cdot q^{j+\alpha}$. That is, let $C \subseteq U_j$ be a clique. Since each $v \in U_j$ has $\tau_v \leq \lfloor q^{j+\alpha} \rfloor$ then

$$\sum_{v \in C} \sum_{t \leq q^{j+\alpha}} x_{v,t} \geq \sum_{v \in C} \frac{1}{2} = |C|/2.$$

On the other hand, by the more general clique constraints (1) we have

$$\sum_{t \leq q^{j+\alpha}} \sum_{v \in C} x_{v,t} \leq \sum_{t \leq q^{j+\alpha}} 1 \leq q^{j+\alpha}.$$

Since G is perfect, then by definition U_j can be colored using at most $2 \cdot q^{j+\alpha}$ colors and such a coloring can be done in polynomial time (Corollary 67.2c[15]). The rest of the analysis is unchanged, thus the full form of Theorem 2 is proven.

Derandomizing. It is simple to efficiently derandomize our approach. We simply list all break points α of the form $\log_q \hat{f}_k - (j_k - 1)$ over all bundles k and try all α between these break points. Our algorithm is deterministic once α is given and these break points are the only values of α where the behavior of the algorithm changes. Taking the best solution found over all such α is surely no worse than the expected cost of the returned solution when choosing α randomly

Extensions to Other Graph Classes. For Corollary 3, the assumptions mean we can approximately separate the clique constraints $\sum_{v \in C} x_{v,t} \leq 1$ in polynomial time ultimately leading to an efficient algorithm that finds an LP solution with cost at most OPT where all constraints hold except perhaps these new clique constraints. Instead, we would have $\sum_{v \in C} x_{v,t} \leq c$ where c is the approximation factor of computing a maximum-weight clique in G.

The approximate relationship between maximum cliques and the chromatic number of graphs satisfying the assumptions of Corollary 3 allow us to conclude U_j can be colored with at most $c' \cdot q^{i+\alpha}$ colors where c' is also a constant. Carrying this term through the rest of the analysis shows the algorithm is an $O(1)$-approximation.

3 MSCB with Task Concurrencies

Recall in **MSCB-TC**, the bundles form a *partition* $V_1, \ldots, V_p$ of P and for each bundle k we have a bound d_k on the number of jobs in V_k that can be scheduled at any single time. This models settings where clients can only deliver/retrieve a bounded number of their jobs at any single time. Also, recall that we assume G is a chordal graph.

The new algorithm starts with (**LP-MSCB**) except the cliques C_i used to define the constraints are the polynomially-many maximal cliques of G [8] (which can be enumerated in polynomial time) and two additional classes of constraints are added. First, for any bundle $1 \leq k \leq p$ and any time t we add the constraints

$$\sum_{v \in V_k} x_{v,t} \leq d_k.$$

That is, at any given time a maximum of d_k jobs for bundle k can be processed. We call these *concurrency constraints*. Next, For any bundle $1 \leq k \leq p$ we add the constraints

$$f_k \geq \lceil |V_k|/d_k \rceil$$

which enforces the trivial lower bound that $\lceil |V_k|/d_k \rceil$ time steps are required to finish bundle k even if we processed d_k of its jobs per step. Note, without this bound the LP could cheat in the following way: if $d_k = 1$ and $V_k = \{v_1, \ldots, v_m\}$ we could set $x_{v_i,t} = 1/m$ for all $1 \leq i \leq m$ and $1 \leq t \leq m$ which would permit us to set $f_k = (m+1)/2$ whereas an integer solution would clearly require $f_k \geq m$.

For the rest of this section, by “schedule” we mean a proper coloring of G with the additional constraint that for any bundle k and any time t we have at most d_k jobs in V_k being colored with t.

We need to make some minor modifications to the algorithm. First, we now define $\hat{f}_k := \max\{\lceil |V_k|/d_k \rceil, \max_{v \in V_k} \tau_v\}$. Next, we change U_j to be

$$U_j = \{k : \lfloor q^{j-1+\alpha} \rfloor < \hat{f}_k \leq \lfloor q^{j+\alpha} \rfloor\}.$$

Finally, when we color U_j, we will ensure that the new concurrency constraints are satisfied with this coloring. The following structural result enables us to do this while limiting the loss in the final approximation guarantee. Here, we are letting $\chi(G)$ denote the chromatic number of G.

▶ **Lemma 10.** *Let $G = (V, E)$ be a chordal graph whose vertices are partitioned as $V_1, \ldots, V_p$. Further, for each $1 \leq k \leq p$ let $d_k \geq 1$ be an integer. In polynomial time, we can compute a proper coloring of G using at most $\chi(G) + \max_k \left\lceil \frac{|V_k|}{d_k} \right\rceil - 1$ colors such that for each $1 \leq k \leq p$, no color appears more than d_k times among nodes in V_k.*

Proof. Recall that a graph is a chordal graph if and only if it has a perfect elimination ordering, i.e. an ordering $v_1, v_2, \ldots, v_n$ such that for each $1 \leq j \leq n$, the *left-neighborhood* $N_\ell(v_j) = \{i < j : v_i v_j \in E\}$ of each node is a clique and that this ordering can be computed in linear time [8].

To compute the coloring we need, process the nodes v_i in this order. When coloring v_i, we simply avoid using a color already assigned to a node in $N_\ell(v_i)$ or already assigned to d_k nodes in the same part V_k as v_i. This can be done if we allow $\chi(G) + \max_k \left\lceil \frac{|V_k|}{d_k} \right\rceil - 1$ colors. ◀

We briefly remark that Lemma 10 is tight even for interval graphs where $d_k = 1$ for all V_k. Consider the case where V_1 consists of m jobs whose corresponding intervals are $[1, 2], [3, 4], [5, 6], \ldots, [2m-1, 2m]$ and $V_2, \ldots, V_p$ each consists of a single job whose corresponding interval is $[1, 2m]$. The chromatic number is exactly p but no two jobs can receive the same color since the only non-intersecting pairs of intervals have their corresponding jobs in the same bundle V_1. Therefore, $|V_1| + |V_2| + \ldots + |V_p| = p + m - 1$ colors are required.

Towards coloring U_j, we define $\mathcal{V}_j = \{k : V_k \cap U_j\} \neq \emptyset$ to be all bundles having some job in U_j and then we let $S_j = \max_{k \in \mathcal{V}_j} \lceil |V_k \cap U_j|/d_k \rceil$. Since $\lceil |V_k \cap U_j|/d_k \rceil$ is a lower bound on the time required to finish all jobs $V_k \cap U_j$ due to the task concurrency constraint for bundle k, we have that S_j is another lower bound for the time needed to complete all jobs in the set U_j. The new LP constraints help assert this lower bound as well.

▶ **Lemma 11.** *For each group j, $S_j \leq q^{j+\alpha}$.*

Proof. This is demonstrated by leveraging the additional constraint incorporated into our LP. For every $k \in \mathcal{V}_j$, we know that $\frac{|V_k|}{d_k} \leq \hat{f}_k$. Furthermore, based on the new definition of U_j it is clear that $\hat{f}_k \leq q^{j+\alpha}$. Consequently, $\frac{|V_k|}{d_k}$ is less than equal to $q^{j+\alpha}$, implying $|V_k \cap U_j| \leq q^{j+\alpha} \cdot d_k$. Therefore, it follows that $S_j \leq q^{j+\alpha}$. ◀

As with the **MSCB** approximation, the maximum clique size in U_j is at most $2 \cdot q^{j+\alpha}$. Further, we have just shown $|V_k \cap U_j| \leq q^{j+\alpha} \cdot d_k$ for any $k \in \mathcal{V}_j$. So Lemma 10 means there is a proper coloring of U_j using at most $3 \cdot q^{j+\alpha}$ colors such that no bundle in $\mathcal{V}_k$ has more than d_k jobs colored with the same color.

The rest of the analysis is similar to the analysis of the algorithm for **MSCB** except the approximation ratio has changed since we used $3 \cdot q^{\alpha+j}$ colors instead of $2 \cdot q^{\alpha+j}$ colors to color each U_j. Thus, it is a $6 \cdot e \leq 16.31$-approximation.

Corollary 6 essentially follows by how Corollary 3 followed from Theorem 2 but using a more general form of Lemma 10. Namely, if there is an ordering of the nodes $v_1, v_2, \ldots, v_n$ such that the left-neighborhood $N_\ell(v_i) = \{v_j : v_i v_j \in E, j < i\}$ of any node v_i can be covered with $R = O(1)$ cliques then we can find a proper coloring of G using at most $R \cdot \chi(G) + \max_k \lfloor |V_k|/d_k \rfloor$ colors by picking the lowest available color not appearing in the left-neighborhood of v_i that is also not used d_k times in the part V_k with v_i.

4 MSCB-TC in Perfect Graphs – A Barrier

Lemma 10 fails to hold in perfect graphs even within any constant factor. That is, it may require $\Theta(\sqrt{n}) \cdot \max\{\chi(G), \max_k |V_k|/d_k\}$ colors to even if $d_k = 1$ for all k. Consider the following simple example on $n = N^2$ nodes for some integer N. The graph $G_N = (V, E)$ is partitioned into sets $V_1, \ldots, V_N$ and each V_k has exactly N nodes. We have an edge between any pair of nodes in different parts, but each part is an independent set.

It is easy to see such graphs are perfect. More generally, a graph that is partitioned into b nonempty independent sets and has all possible edges between these parts has chromatic number b and maximum clique size b (pick one node from each part). Since any induced subgraph of our graph G_N is of this form, then G_N is also perfect.

But any coloring satisfying task concurrency limits of $d_k = 1$ for all parts must in fact use n colors. Two nodes in different parts cannot receive the same color because they are connected by an edge and two nodes in the same part cannot receive the same color because the task concurrency limit is 1. Yet, $\chi(G) = N = \sqrt{n}$ and the maximum size of a part is also $N = \sqrt{n}$.

Still, this is the worst case. The following variation of Lemma 10 leads to an $O(\sqrt{n})$-approximation for **MSCB-TC** in perfect graphs.

▶ **Lemma 12.** *Let $G = (V, E)$ be a graph whose vertices are partitioned as $V_1, \ldots, V_p$. Further, for each $1 \leq k \leq p$ let $d_k \geq 1$ be an integer. There is a proper coloring of G using at most $\sqrt{n} \cdot \max\left\{\chi(G), \max_k \left\lceil \frac{|V_k|}{d_k} \right\rceil\right\}$ colors such that for each $1 \leq k \leq p$, no color appears more than d_k times among nodes in V_k. Such a coloring can be computed in polynomial time if G can be optimally colored in polynomial time.*

Proof. If $\max_k \lceil |V_k|/d_k \rceil \geq \sqrt{n}$, then the trivial n-coloring (i.e. all nodes get different colors) suffices. Otherwise, consider a proper coloring $\sigma : V \to \{1, 2, \ldots, \chi(G)\}$ of G. Order the nodes $v^k_1, v^k_2, \ldots, v^k_{|V_k|}$ arbitrarily in each part V_k.

Recolor a vertex v^k_i with the pair $(\chi(v^k_i), \lfloor i/d_k \rfloor)$. Clearly, this is a proper coloring since the first components of the new colors of nodes will differ on any edge of G. Further, at most d_k nodes in V_k will have the same second part of the pair describing their color. The number of colors used is $\chi(G) \cdot \max_k \lceil |V_k|/d_k \rceil \leq \chi(G) \cdot \sqrt{n}$, as required.

Finally, this can be done in polynomial time if we can compute a coloring of G with $\chi(G)$ colors in polynomial time. ◀

Using this in place of Lemma 10 yields an $O(\sqrt{n})$-approximation for **MSCB-TC** in perfect graphs. This proves Theorem 7.

5 Conclusion

We have given the first constant-factor approximations for **MSCB** in a large variety of graph classes including perfect graphs and unit-disc graphs. Our techniques extend to give the first constant-factor approximations for **MSCB-TC** in certain graphs like chordal graphs, interval graphs, and unit-disc graphs.

It would be interesting to see what other graph classes admit constant-factor approximations for **MSCB** and, perhaps, also for **MSCB-TC**. Another interesting direction would be to get a purely combinatorial constant-factor approximation for **MSCB** in certain graph classes, i.e. one that avoids solving a linear program. Such algorithms exist for **MSC** in many cases, e.g. [2, 10]. One barrier is that it seems hard to approximate the maximum number of bundles that can be completed in a given number of time steps even in simple graph classes like interval graphs (Lemma 8). Perhaps a bicriteria approximation could be designed to circumvent this hardness, it would immediately lead to an $O(1)$-approximation through standard minimum latency arguments.

References

1 S. Ahmadi, S. Khuller, M. Purohit, and S. Yang. On scheduling coflows – (extended abstract). In *Proceedings of 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 13–24, 2017.

2 A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140(2):183–202, 1998.

3 B. K. Bhattacharya and D. Kaller. An $o(m + n \log n)$ algorithm for the maximum-clique problem in circular-arc graphs. *Journal of Algorithms*, 25(2):336–358, 1997.

4 B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

5 I. DeHaan and Z. Friggstad. Approximate minimum sum colorings and maximum k-colorable subgraphs of chordal graphs. In *Algorithms and Data Structures Symposium (WADS)*, pages 326–339, 2023.

6 T. Fukunaga. Integrality gap of time-indexed linear programming relaxation for coflow scheduling. In *In Proceedings of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques(APPROX)*, volume 245, pages 36:1–36:13, 2022.

7 R. Gandhi, M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Improved bounds for sum multicoloring and scheduling dependent jobs with minsum criteria. In *Approximation and Online Algorithms*, pages 68–82, 2005.

8 M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

9 M. M. Halldórsson and G. Kortsarz. Algorithms for chromatic sums, multicoloring, and scheduling dependent jobs. In *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methologies and Traditional Applications*, pages 671–684. Chapman and Hall/CRC, 2018.

10 M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37:187–209, 2003.

11 P. E. Haxell. A condition for matchability in hypergraphs. *Graphs and Combinatorics*, 11:245–248, 1995.

12 S. Im, B. Moseley, K. Pruhs, and M. Purohit. Matroid Coflow Scheduling. In *46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 145:1–145:13, 2019.

13 M. J. P. Peeters. On coloring j-unit sphere graphs. Research Memorandum FEW 512, Tilburg University, School of Economics and Management, 1991.

14 Z. Qiu, C. Stein, and Y. Zhong. Minimizing the total weighted completion time of coflows in datacenter networks. In *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 294–303, 2015.

15 A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency.* Springer, 2003.

16 M. Shafiee and J. Ghaderi. An improved bound for minimizing the total weighted completion time of coflows in datacenters. *IEEE/ACM Transactions on Networking*, 26(4):1674–1687, 2018.

17 D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. of ACM Symposium on Theory of Computing (STOC 2006)*, pages 681–690, 2006.

A Greedy Coloring in Trees

We point out the greedy algorithm that iteratively picks a maximum independent set to color a graph may be as bad as an $\Omega(\log n)$-approximation. We point this out to show that the greedy 4-approximation for unweighted **MSC** does not extend to our setting which includes, as a special case, computing the chromatic number of a graph. While this seems to be well known in the community, we are unaware of a particular reference for such an example so we provide a simple construction here for completeness.

Let T_0 be the trivial tree with a single node. Inductively for $i \geq 1$ let T_i be constructed by attaching 2 leaf nodes to each node of T_{i-1}. So the number of nodes in T_i is 3^i.

The only maximum independent set in T_i is the set of all leaves in T_i (which clearly forms an independent set). To see this, let I be an independent set that includes an internal vertex of T_i (i.e. a node of T_{i-1}). Neither leaf that was attached to v to form T_i is in I because I is an independent set. But then we get a larger independent set by removing v from I and adding in the two associated leaves.

The greedy algorithm to compute a maximum independent set in T_i will first pick all of its leaves. Removing them leaves tree T_{i-1}. So by induction, with the base case $i = 0$ clearly requiring a single iteration to color, the number of iterations will be $i + 1$. Since $i + 1$ is logarithmic in the size of T_i (i.e. $n := 3^i$) and since the chromatic number of $T_i, i \geq 1$ is 2 (as is true for any tree with at least one edge), this is an $\Omega(\log n)$-approximation for the chromatic number of a tree.