

The Sum Multi-Coloring Problem

Ravit Salman

The research was carried out under the supervision of
Doctor Hadas Shachnai in the Department of Computer Science.

I wish to express my deepest gratitude and appreciation
to Dr. Hadas Shachnai
for her excellent guidance which made this work possible.

The generous financial support of
the Technion—Israel Institute of Technology
is greatly acknowledged.

Contents

Abstract	1
list of Symbols	3
1 Introduction	4
1.1 Statement of the Problem	4
1.2 Applications	5
1.3 Main Results	6
1.4 Organization of this Work	7
2 Related Work	8
2.1 The Hardness of the SC Problem	8
2.2 Exact solutions for the SC	9
2.3 Approximate solutions for the SC	9
2.3.1 Classes of Graphs	9
2.3.2 Approximation Results	10
2.4 Recent result for the SMC problem	11
3 Preliminaries	12
3.1 Definitions and Notation	12
3.2 The Execution Models	13
3.2.1 The Preemptive Model	13
3.2.2 The Non-Preemptive Model	13
3.2.3 The Co-Scheduling Model	13
3.3 Approximation Ratios	14
4 Approximate Solutions	15
4.1 Bounded-degree Graphs	15
4.1.1 Approximation Algorithm for the pSMC	15
4.1.2 Approximation Algorithm for the npSMC	16
4.2 Performance Bounds for the MaxIS	17

4.2.1	Extensions for two Variants of SC	17
4.2.2	Application of MaxIS for the coSMC	18
5	Exact Solutions for the SMC Problem	22
5.1	The Tree-Color Algorithm	22
5.1.1	Notation and Data Structures	22
5.1.2	Informal Description of the Algorithm	23
5.1.3	Efficient Implementation of the Tree-color Algorithm	23
5.2	Algorithms for the SMC on Stars	29
5.2.1	An algorithm for npSMC and coSMC on Stars	29
5.2.2	An Algorithm for pSMC on Stars	31
6	Summary	33
6.1	The Contribution of this Work	33
6.2	Open Problems	33
	Publications	35
	Bibliography	36

List of Figures

4.1	The coloring under MMaxIS of the graph \tilde{G}	19
4.2	The coloring of the graph \tilde{G} under A	20
5.1	The <i>Tree-color</i> Algorithm	25
5.2	The <i>Star-color</i> Algorithm	30
5.3	Sketch of Lemma 5.5	32

Abstract

Scheduling dependent jobs on multiple machines is modeled by the graph *multi-coloring* problem. We consider the problem of minimizing the average completion time of all jobs. The scheduler has to satisfy the following two conditions:

- (i) *mutual exclusion*: no two conflicting jobs are executed simultaneously.
- (ii) *no starvation*: the request of any job to run is eventually granted.

The resulting coloring problem can be stated as follows: Given a graph $G = (V, E)$ and the number of colors $x(v) \in \mathbb{Z}^+$ required by each vertex $v \in V$, find a multi-coloring which minimizes the sum of the largest colors assigned to the vertices. A multi-coloring of the vertices in G is a mapping into the power set of the positive integers, $\Psi : V \mapsto 2^{\mathbb{N}}$. Each vertex v is assigned a set S_v of $x(v)$ distinct numbers (colors), and adjacent vertices are assigned disjoint sets of colors.

Earlier work refer to the special case, where all jobs have unit execution time; this is known as the *sum coloring* (SC) problem. Finding an $n^{1-\epsilon}$ -approximation for the SC is NP-hard, for some $\epsilon > 0$, where n is the number of vertices.

In this work we address the SC problem as well as the case, where jobs can have arbitrary execution times. We call this generalized version of the SC the *sum multi-coloring* (SMC) problem.

We study the SMC problem in three models, which appear in many applications. In the Preemption model (pSMC), a vertex can be assigned any set of colors; in the No-Preemption model (npSMC), this set of colors has to be *contiguous*. In the Co-Scheduling model (coSMC), the vertices are colored in rounds: in each round the scheduler completely colors an independent set in the graph. These models correspond to three common approaches in scheduling (dependent) jobs: in the Preemption model jobs may be interrupted during their execution and resumed at later time; in the No-Preemption model scheduled jobs must run to completion. In the Co-Scheduling approach the

scheduler identifies subsets of *cooperating processes* that can benefit from running at the same time interval. Then, each subset is executed simultaneously on several processors, until *all* the processes in the subset are completed.

We present algorithms for the SMC problem in these three models. In particular, we give algorithms for the pSMC problem and the npSMC on bounded-degree graphs, with approximation ratio $O(\Delta)$, where Δ is the maximal degree in G . For trees, we give an algorithm, which solves optimally the npSMC in $O(np)$ steps, where $n = |V|$, and p is the maximal color requirement of any vertex. For the special case where G is a star, we show that the SMC can be solved optimally, in all three models, in $O(\min(p + n, n \lg n))$ steps. We prove that on stars coSMC is equivalent to npSMC, and pSMC behaves similar to these two cases.

Finally we study further the MaxIS algorithm, that was shown in [BBH⁺98] to provide close approximation for the SC problem. We show that MaxIS is efficient also for two variants of the SC problem. In contrast, MaxIS can perform poorly when applied for the SMC problem. Specifically, we show that MaxIS is $\Omega(\min(n, p))$ -approximation for the coSMC.

List of Symbols

$G(V, E)$	a graph G with a set of vertices V and edges E
$x(v)$	the color requirement (or length) of v
$f_{\Psi}(v)$	the maximum color assigned to v by a multi-coloring Ψ
SC	sum coloring
SMC	sum multi-coloring
pSMC	preemptive sum multi-coloring
npSMC	non-preemptive sum multi-coloring
coSMC	co-scheduling sum multi coloring
n	the number of vertices in a graph $G = (V, E)$
p	$\max_{v \in V} x(v)$, i.e., the maximum color requirement in G
Δ	the maximum degree of the graph
O	upper bound
Ω	lower bound
Θ	upper and lower bound
T	a graph $G = (V, E)$ which is a tree
T_v	a tree rooted at v
$kids(v)$	the set of children of v in a rooted tree
$par(v)$	v 's parent in a rooted tree
$N(v)$	the set of neighbours of v
$[x, y]$	the set $\{x, x + 1, \dots, y\}$, where $x, y \in Z$

Chapter 1

Introduction

In this chapter we give a detailed description of the *sum multi-coloring* (SMC) problem, which is studied in this work. We then discuss some of its applications and outline the main results of our study.

1.1 Statement of the Problem

Any multi-processor system has certain resources, which can be made available to one job at a time. A fundamental problem in distributed computing is to efficiently schedule jobs that are competing on such resources. The scheduler has to satisfy the following two conditions:

- (i) *mutual exclusion*: no two conflicting jobs are executed simultaneously.
- (ii) *no starvation*: the request of any job to run is eventually granted.

The problem is well-known in its abstracted form as the *dining/drinking philosophers* problem (see, e.g., [L81, NS93, SP88]).

Scheduling dependent jobs on multiple machines is modeled as a graph *coloring* problem, when all jobs have unit execution times, and as graph *multi-coloring* for arbitrary execution times. The vertices of the graph represent the jobs and an edge in the graph between two vertices represents a dependency between the two corresponding jobs, that forbids scheduling these jobs at the same time. More formally, for a weighted undirected graph $G = (V, E)$ with n vertices, let the *length* of a vertex v be a positive integer denoted by $x(v)$ and called the *color requirement* of v . A multi-coloring of the vertices of G is a mapping into the power set of the positive integers, $\Psi : V \mapsto 2^{\mathbb{N}}$. Each vertex v is assigned a set S_v of $x(v)$ distinct numbers (colors) and adjacent vertices are assigned disjoint sets of colors.

The traditional optimization goal is to minimize the total number of colors assigned to G . In the setting of a job system, this is equivalent to finding a schedule, in which the time until *all* the jobs complete running is minimized. Another important goal in such systems is to minimize the *average* completion time of the jobs. Formally, given a multi-coloring Ψ , define by $f_\Psi(v)$ the maximum color assigned to v by Ψ . Then minimizing the average completion time is equivalent to minimizing the sum of all these numbers. In the SMC problem, we look for a multi-coloring Ψ that minimizes $\sum_{v \in V} f_\Psi(v)$. When all the color requirements equal to 1 the problem is reduced to the *sum coloring* (SC) problem. We study the sum multi-coloring problem in three models:

- In the Preemption model (pSMC), each vertex may get any set of colors.
- In the No-Preemption (npSMC) model, the set of colors assigned to each vertex has to be contiguous.
- In the Co-Scheduling model (coSMC), the vertices are colored in rounds: in each round the scheduler completely colors an independent set in the graph.

The Preemption model corresponds to the scheduling approach commonly used in modern operating systems [SG98]: jobs may be interrupted during their execution and resumed at a later time. The No-Preemption model captures the execution model adopted in real-time systems, where scheduled jobs must run to completion. The Co-Scheduling approach is used in some distributed operating systems [T95]. In such systems, the scheduler identifies subsets of *cooperating processes* that can benefit from running at the same time interval (e.g., since the processes in the set communicate frequently with each other). Then, each subset is executed simultaneously on several processors, until *all* the processes in the subset are complete.

1.2 Applications

The Sum Multi Coloring problem has many applications. We describe below some of the main applications of this problem.

- **Traffic intersection control** [B92, BH94, IL96]: Cars should cross an intersection in conflicting directions. The intersection controller must schedule the vehicles through the intersection so as to avoid any collisions. We can identify a job with a platoon of cars, that has to pass through the intersection as soon as possible.

- **Session scheduling in local-area networks** [CCO93, IL96]: Local area networks with spatial reuse allow the concurrent access and transmission of user data, with no intermediate buffering of packets. If some node s has to send data to some other node t , a session is established between s and t . A session typically lasts for much longer than its data transmission time, and can be active only if it has exclusive use of all the links in its route from s to t . Therefore, sessions whose routes share at least one link are in conflict. Data transmission sessions must be scheduled, so as to avoid these conflicts.
- A natural application, in which the resulting conflict graph is a *tree*, is **packet routing on a tree network topology**: each node can conflict over its neighboring links, either with its parent or children in the tree. Thus, the conflict graph is induced by the network topology. **Conflicts among processes** running on a single-user machine (e.g., PCs) are typically for shared data. In many operating systems, the creation of a new process is done by ‘splitting’ an existing process, via a ‘fork’ system call [SG98]. Thus, the set of processes form a tree where each process is a node. Conflicts over shared data typically occur between a process and its immediate descendants/ancestor in that tree, as these processes will share parts of their codes. Thus, the conflict graph is also a tree.

1.3 Main Results

In this work we present the following results for the SMC problem:

- An $O(np)$ optimal algorithm for npSMC on trees, where $p = \max_{v \in V} x(v)$ and $n = |V|$.
- For the case where G is a star, we give algorithms that solve optimally the npSMC, the coSMC and the pSMC in $O(\min(n+p), n \log n)$ steps.
- For bounded degree graphs we give a $(\Delta + 1)$ -approximation algorithm for the pSMC, and $(2\Delta + 1)$ -approximation algorithm for the npSMC, where Δ is the maximum degree in G .
- The MaxIS algorithm is a $\Omega(\min(n, p))$ -approximation for the coSMC.
- The MaxIS provides $O(1)$ -approximations for two variants of the SC problem, namely the *bounded-IS* and the *constrained-SC* problems.

1.4 Organization of this Work

In Chapter 2 we discuss some related work, dealing with the SC and the SMC problems. In Chapter 3 we introduce the definitions and notation that will be used throughout this study.

In Chapter 4 we present approximation algorithms for the SMC problems. In Chapter 5 we give exact algorithms for the SMC problems on trees and on stars. Finally, chapter 6 contains a summary of our study, its contribution and a discussion of some open problems.

Chapter 2

Related Work

The SC problem was first introduced by Kubicka in [K89]. Earlier literature on graph coloring dealt with the concept of *chromatic number* of a graph, which is the minimum number of colors required for proper coloring. In the SC problem we minimize the *sum* of the colors assigned to the vertices.

In this work we consider the SMC problem, in which every vertex can require an arbitrary number of colors. As far as we know, previous studies do not refer to this problem. In section 2.4 we mention recent papers on the SMC problem, which extend the results in the present work.

2.1 The Hardness of the SC Problem

The SC problem is known to be hard to solve exactly and to approximate. We list below the main hardness results derived for this problem.

- In [KS89] it was shown that the SC problem is *NP-hard*.
- In [KKK89] it was shown that approximating the SC within an additive constant factor is *NP-hard*.
- Feige and Kilian [FK96] have shown that the chromatic number of general graphs is hard to approximate within $n^{1-\epsilon}$ factor, for any $\epsilon > 0$, unless $NP = ZPP$. Bar-Noy et al. in [BKH⁺98] conclude that the SMC is hard to approximate to within factor $n^{1-\epsilon}$.
- Finally, [BK98] showed that the SC is hard to approximate to within some constant factor $c > 1$, on the subclass of bipartite graphs.

These hardness results carry over to the multi-coloring generalizations.

2.2 Exact solutions for the SC

- Kubicka and Schwenk gave in [KS89] a linear time algorithm, which uses dynamic programming, for finding the minimum sum coloring in trees.
- Jansen [J97] extended the dynamic programming strategy to graphs of bounded treewidth. He proved that the SC for graphs $G = (V, E)$ with constant treewidth k , can be solved in time $O((\lg |V|)^{k+1}|V|)$. If the maximum degree $\Delta(G)$ is bounded by a constant, then the problem can be solved in linear time $O(|V|)$. This is done by dynamic-programming.

2.3 Approximate solutions for the SC

Approximation algorithms for the SC problem were given for various classes of graphs. First, we give definitions of these classes; then we list the approximation results for each class.

2.3.1 Classes of Graphs

- Interval graph: given a set of intervals, denoted by η , each interval $i \in \eta$ is represented by its two endpoints l_i and r_i . The interval graph $G = (V, E)$, is the intersection graph of the intervals of η . Thus, the intervals of η are in one to one correspondence with the vertices of G , and an edge connects two vertices, if and only if the corresponding intervals have nonempty intersection.
An interval graph is *proper* if none of the intervals properly contains another one.
A *containment* interval graph is the intersection graph of the family of intervals, such that for any pair of interval that intersect, one of the intervals is properly contained in the other.
- Bipartite graph: suppose that the set of the vertices of a graph G can be divided into two disjoint sets V_1 and V_2 , in such a way that every edge of G joins a vertex of V_1 to a vertex of V_2 ; G is then said to be a bipartite graph.
- Sparse graphs: are graphs in which the number of edges is *linear* in the number of vertices.

- Line graph: the line graph of G , denoted by $L(G)$, is the graph whose vertices are the edges of G , with $(e, f) \in E(L(G))$ when $e = (u, v)$ and $f = (v, w)$ are in G (i.e., when the edges e and f share a vertex in G).
- Perfect graph: a graph G is perfect if the chromatic number of every induced subgraph of G equals the size of the largest clique in this subgraph.
- Planar graph: a plane graph is a graph drawn in the plane in such a way that no two edges intersect geometrically, except at a vertex to which they are both incident; a planar graph is one which is isomorphic to a plane graph.

2.3.2 Approximation Results

- Interval graphs [NSS94]: Nicoloso et al. present an ϵ -approximation algorithm where $\epsilon < 2$ for interval graphs. The running time of the algorithm is $O(m\chi(G) + n \lg n)$, where n, m and $\chi(G)$ are the size of the graph, the number of cliques, and its chromatic number, respectively. This algorithm is shown to solve the SC problem exactly on proper interval graphs and on containment interval graphs, as well as on interval graphs with all integer interval endpoints and interval lengths not exceeding 3.
- Bipartite graphs [BK98]: Bar-Noy and Kortsarz presented a $\frac{10}{9}$ -approximation algorithm for the SC problem on bipartite graphs. This improves the previous $\frac{9}{8}$ approximation algorithm of Bar-Noy et al. [BBH⁺98].
- Sparse graphs [KKK89]: Kubicka et al. showed that for a family F_k of graphs such that $G \in F_k \Rightarrow |E(G)| \leq k|V(G)|$ the performance ratio of a greedy approximation algorithm is bounded by the constant $(k + 1)$.
- Line graphs: Using *compact* coloring, which takes in each stage a maximal independent set, [BBH⁺98] gave 2 approximation for the SC on line graphs.
- Bounded-degree graphs: [BBH⁺98] also showed that compact coloring of a graph G provides a $\frac{\Delta+2}{3}$ approximation to SC, and that it is tight.
- General graphs: A “good” solution for the sum coloring problem tries to color as many vertices as possible, as “early” as possible. This suggests the following natural MaxIS heuristic, proposed in [BBH⁺98] for the SC problem:

MaxIS: Choose a maximum independent set in the graph, color all of its vertices with the next available color, and iterate until all vertices are colored.

This procedure is shown to produce a 4-approximation algorithm. However, the algorithm is polynomial only for those graphs for which a maximum independent set can be found in polynomial time. In addition, this heuristic provides an $O(\rho)$ -approximation polynomial time algorithm, if the maximum independent set can be solved by an $O(\rho)$ -approximation polynomial time algorithm. We note, that coloring the graph with minimum number of colors does not always help in solving the sum coloring problem. For instance, while bipartite graphs can be colored with two colors, there exist bipartite graphs (in fact, trees) for which the optimal sum coloring uses $\Omega(\log n)$ colors [KS89].

- Perfect graphs [BBH⁺98]: In perfect graphs the maximum independent set can be found in polynomial time, so the MaxIS algorithm is an $O(1)$ approximation algorithm.

2.4 Recent result for the SMC problem

Following this work, several recent papers present results for the SMC problems; the paper [BKH⁺98] gives approximation algorithms for general graphs, and for some important special classes of graphs, including bipartite, k -colorable, bounded degree and line graphs. The paper [HKP⁺99] gives an optimal algorithm for the npSMC on trees and a polynomial time approximation scheme (PTAS) for the pSMC on trees. Shortly after, the paper [HK99] gave PTASes for both the npSMC and pSNC on planar graphs.

Chapter 3

Preliminaries

3.1 Definitions and Notation

Let $G = (V, E)$ be an undirected graph, V denotes the vertices, E is the set of edges, and n is the number of vertices.

The color requirement (or *length*) of each vertex $v \in V$ is given by the mapping $x : V \rightarrow \mathbf{N}$. For a given mapping x , we denote by $\mathcal{S}(G) = \sum_v x(v)$ the sum of the color requirements of the vertices in G .

Let p be the maximum color requirement in G , that is $p = \max_{v \in V} x(v)$.

Definition 3.1 *An independent set in G is a subset of vertices $I \subseteq V$, such that any two vertices in I are non-adjacent.*

Definition 3.2 *A multi-coloring of G is an assignment $\Psi : V \rightarrow 2^{\mathbf{N}}$, such that each vertex $v \in V$ is assigned $x(v)$ distinct colors, and the set of vertices colored with i is independent.*

Denote by C_i the independent set of $v \in V$, such that $i \in \Psi(v)$; let $c_1^\Psi(v), \dots, c_{x(v)}^\Psi(v)$ be the collection of $x(v)$ colors assigned to v ; $f_\Psi(v) = c_{x(v)}^\Psi(v)$ is the largest color assigned to v .¹

Definition 3.3 *Given a graph $G = (V, E)$, a mapping $x : V \rightarrow \mathbf{N}$, and a multi coloring of G , $\Psi : V \rightarrow 2^{\mathbf{N}}$, the multi-color sum of G with respect to Ψ is*

$$SMC(G, \Psi) = \sum_{v \in V} f_\Psi(v).$$

¹For a given multi-coloring Ψ we use for short $f(v)$ to denote the highest color of v

3.2 The Execution Models

3.2.1 The Preemptive Model

A multi-coloring Ψ is *preemptive* if each vertex v can get any set of $x(v)$ colors. In the context of scheduling, this means that the jobs may be interrupted during their execution, and resumed at a later time.

Definition 3.4 *The minimum multi-color sum of a graph G , is given by*

$$pSMC(G) = \min_{\Psi} SMC(G, \Psi).$$

3.2.2 The Non-Preemptive Model

Definition 3.5 *A multi-coloring Ψ is non-preemptive (contiguous), if for any $v \in V$, the colors assigned to v satisfy $c_{i+1}^{\Psi}(v) = c_i^{\Psi}(v) + 1$, for $1 \leq i < x(v)$.*

In the context of scheduling, this means that all the jobs are processed without interruption. We denote the minimum contiguous multi-color sum of G by $npSMC(G)$.

3.2.3 The Co-Scheduling Model

Definition 3.6 *A contiguous multi-coloring Ψ solves the co-scheduling problem, if the set of vertices can be partitioned into k disjoint independent sets $V = I_1 \cup \dots \cup I_k$ with the following two properties:*

(i) $c_1^{\Psi}(v) = c_1^{\Psi}(v')$ for any $v, v' \in I_j$, for $1 \leq j \leq k$.

(ii) $c_{x(v)}^{\Psi}(v) < c_1^{\Psi}(v')$ for all $v \in I_j$ and $v' \in I_{j+1}$ for $1 \leq j < k$.

In the context of scheduling, this means scheduling to completion all the jobs corresponding to I_j , and only then starting to process the jobs in I_{j+1} for all $1 \leq j < k$.

The minimum multi-color sum of G for the co-scheduling problem is denoted by $coSMC(G)$.

Lemma 3.1 *For any graph G*

$$\mathcal{S}(G) \leq pSMC(G) \leq npSMC(G) \leq coSMC(G).$$

Proof: Note, that $\mathcal{S}(G)$ is a lower bound to SMC in every graph. This bound is reached when the graph is an independent set. The lemma follows from the fact that $coSMC(G)$ is a special case of $npSMC(G)$ and $npSMC(G)$ is a special case of $pSMC(G)$. \square

3.3 Approximation Ratios

Let P be one of the three sum multi-coloring problems, and \mathcal{C} a class of graphs.

Definition 3.7 *An algorithm A approximates P by a factor β for \mathcal{C} , if for any graph $G \in \mathcal{C}$*

$$\frac{SMC(G, A)}{SMC(G, OPT(G))} \leq \beta ,$$

where $OPT(G)$ is the sum multi-coloring of G produced by the best solution for P on G .

Chapter 4

Approximate Solutions

In this chapter we give approximation algorithms for the SMC problem. Section 4.1 presents approximation algorithms for bounded degree graphs, in which the maximal degree is some constant $\Delta \geq 1$. In section 4.2 we study the performance of the MaxIS, when applied to the SMC problem, and for two variants of the SC problem.

4.1 Bounded-degree Graphs

4.1.1 Approximation Algorithm for the pSMC

The following is the p-Greedy algorithm:

- Order the vertices arbitrarily.
- Assign to each vertex $v \in V$ the set S_v of the smallest $x(v)$ colors, with which non of its preceding neighbours has been colored.

Theorem 4.1 *p-Greedy approximates pSMC within a factor of $\Delta + 1$, where Δ is the maximum degree in G .*

Proof: Let $N(v)$ denotes the set of neighbours of v . Observe, that for any $v \in V$, $f(v)$ is at most $x(v) + \sum_{u \in N(v)} x(u)$. Hence,

$$\begin{aligned} SMC(G, \text{p-Greedy}) &= \sum_{v \in V} f(v) \leq \sum_{v \in V} (x(v) + \sum_{u \in N(v)} x(u)) \\ &= \sum_{v \in V} x(v) + \sum_{v \in V} \sum_{u \in N(v)} x(u). \end{aligned}$$

We note, that each $v \in V$ is added $N(v) \leq \Delta$ times in the right double summation, therefore we get, that

$$SMC(G, \text{p-Greedy}) \leq \sum_{v \in V} x(v) + \Delta \sum_{v \in V} x(v) = (\Delta + 1) \sum_{v \in V} x(v).$$

In any optimal multi-coloring of G , $f(v) \geq x(v)$, $\forall v \in V$, thus

$$pSMC(G) \geq \sum_{v \in V} x(v).$$

This yields the bound. □

4.1.2 Approximation Algorithm for the npSMC

Following is the np-Greedy algorithm:

- Order the vertices arbitrarily.
- Assign to each vertex $v \in V$ a contiguous set S_v of the smallest $x(v)$ colors, with which non of its preceding neighbours have been colored.

Theorem 4.2 *np-Greedy approximates npSMC within a factor of $2\Delta + 1$, where Δ is the maximum degree in G .*

Proof: For any $v \in V$, $f(v) \leq x(v) + \sum_{u \in N(v)} x(u) + \Delta(x(v) - 1)$.

We add the last term, since there can be a ‘gap’ between the colorings of any two neighbours, of at most $x(v) - 1$ colors, which v cannot use; there are at most Δ gaps. Hence,

$$\begin{aligned} SMC(G, \text{np-Greedy}) &= \sum_{v \in V} f(v) \\ &\leq \sum_{v \in V} (x(v) + \sum_{u \in N(v)} x(u) + \Delta(x(v) - 1)) \\ &\leq (\Delta + 1) \sum_{v \in V} x(v) - n\Delta + \Delta \sum_{v \in V} x(v) \\ &= (2\Delta + 1) \sum_{v \in V} x(v) - n\Delta. \end{aligned}$$

Again, $npSMC(G) \geq \mathcal{S}(G)$, this yields the desired bound. □

4.2 Performance Bounds for the MaxIS

The following algorithm, known as the MaxIS [BBH⁺98] is a natural approach for solving the SC problem.

MaxIS: Iteratively, find a maximum independent set IS_i for $i \geq 1$, color IS_i with i , and omit from G the vertices and edges of IS_i , until $G = \emptyset$.

4.2.1 Extensions for two Variants of SC

In this section we show that MaxIS provides efficient approximation for two variants of the SC problem.

The Bounded Independent Sets Problem

The problem: Given $G = (V, E)$, find a coloring for which the color sum is minimized, such that the size of any color class is at most L .

Consider the bMaxIS algorithm which finds in each iteration a maximum independent set I ; if $|I| > L$ it colors an (arbitrary) subset $I' \subseteq I$ of vertices.

Theorem 4.3 *For any graph G , in which a maximum independent set can be found in polynomial time, bMaxIS is a 4-approximation for the bounded-IS problem.*

Proof: Following the result in [BBH⁺98], which states that $\frac{MaxIS(G)}{SC(G)} \leq 4$, we note, that when we look for independent sets of size no more than L , the ratio between MaxIS and SC does not change. The proof of the 4-ratio in [BBH⁺98] proceeds by partitioning the vertices to k subsets, such that $|V_1| \geq |V_2| \geq \dots \geq |V_k|$. We can choose $|V_1|$ to be at most L , and follow the steps of this proof. \square

The Constrained SC Problem

The problem: Let $G = (V, E)$ be a graph and let G_1, G_2, \dots, G_l be a partition of G to subgraphs. Color G , where in each iteration there will be at most l vertices, and in each independent set there can be at most one vertex from every subgraph.

Theorem 4.4 *For any graph G , in which a maximum independent set can be found in polynomial time, MaxIS is a 4-approximation for the constrained SC problem.*

Proof: We prove the theorem by reduction to the SC problem. Given a graph G , construct the graph G' as follows: add in G_i edges, such that the subgraph G_i becomes a clique in G' . Now, we solve the SC problem on G' . Clearly, we can choose in each iteration an independent set I of at most l vertices, such that the number of vertices from each subgraph G_i is at most one. Thus, we can use the 4-bound obtained for the MaxIS in [BBH⁺98] \square

4.2.2 Application of MaxIS for the coSMC

Consider the following algorithm, that we call *Multi MaxIS (MMaxIS)*, for the SMC problem: in each iteration we color every vertex v in the maximum independent set IS_i with $x(v)$ consecutive colors, starting (immediately) after the last color used for vertices in IS_{i-1} . Formally, for any $v \in IS_i$,

$$f(v) = x(v) + \sum_{j=1}^{i-1} \max_{u \in IS_j} x(u).$$

In [BBH⁺98, BKH⁺98] it is shown that the performance ratio of MaxIS for the SC problem is exactly 4, when a maximum independent set (IS) can be found in polynomial time. (The bound generalizes to 4ρ -approximation of SMC for classes of graphs, for which the IS can be approximated within a factor of ρ). In the next result we show, that these bounds do not hold for the MMaxIS, when applied for the SMC problem. We exemplify for the coSMC, in which the optimum is forced to color the vertices contiguously, in rounds, like MMaxIS does.

Theorem 4.5 *MMaxIS is a $\Omega(\min(n, p))$ approximation for the coSMC, where p is $\max_{v \in V} x(v)$.*

Proof: We give an example of a graph \tilde{G} , on which MMaxIS has a performance ratio of $\Omega(\min(n, p))$. The graph $\tilde{G} = (V, E)$ has two sets of vertices S_1 and S_2 : for $v \in S_1$, $x(v) = p$, and for $v \in S_2$, $x(v) = 1$, $|S_1| = \sqrt{n}-1$, $|S_2| = n - \sqrt{n} + 1$. The edges in \tilde{G} are as follows: The vertices in S_1 are non adjacent; the vertices in S_2 are partitioned to $\sqrt{n}-2$ independent sets, $IS_1, \dots, IS_{\sqrt{n}-2}$, such $|IS_1| = \sqrt{n} + 1$ and $|IS_2| = |IS_3| = \dots = |IS_{\sqrt{n}-2}| = \sqrt{n}$. The vertices in IS_i are connected to all vertices in IS_j , for $j \neq i$. Finally, we number the

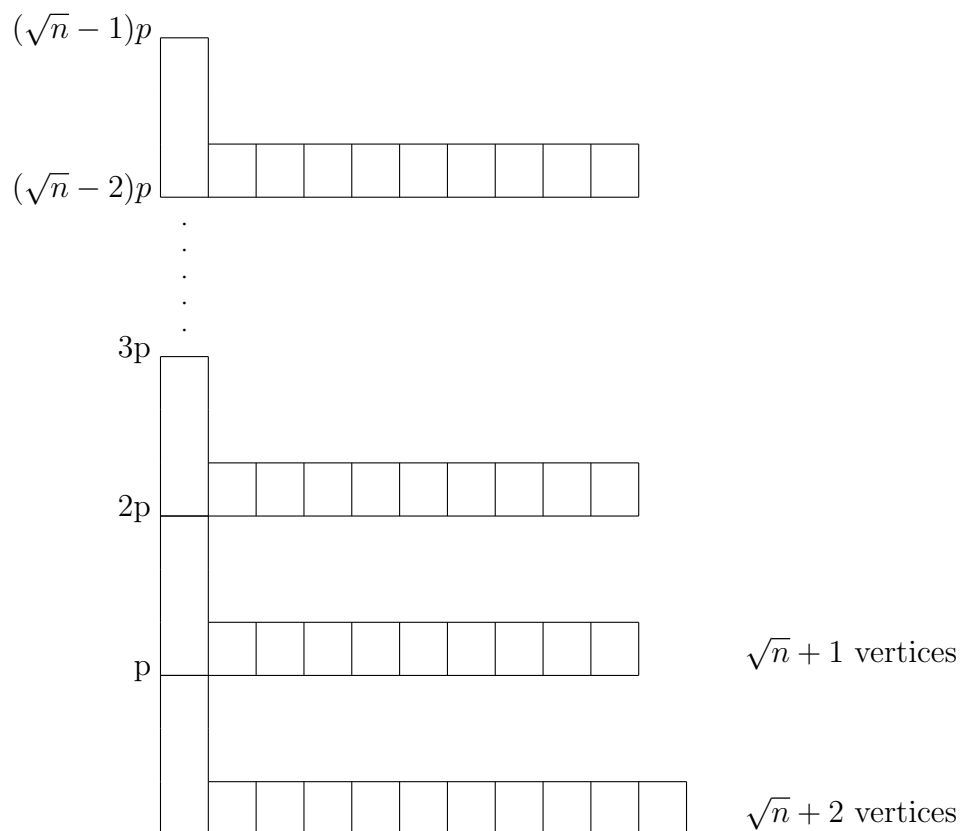


Figure 4.1: The coloring under MMaxIS of the graph \tilde{G} .

vertices in S_1 by $v_1, \dots, v_{\sqrt{n}-1}$, then v_i is connected to all the vertices in S_2 , except for the vertices in IS_i .

MMaxIS will choose initially an independent set of $\sqrt{n}+2$ vertices: $\sqrt{n}+1$ with length 1 and one with length p . Then, MMaxIS will choose iteratively $\sqrt{n}-2$ independent sets, each of size $\sqrt{n}+1$: \sqrt{n} vertices of length 1, and one vertex of length p . Figure 4.1 is a sketch of this coloring.

Consider now the algorithm A , which colors first an independent set of $\sqrt{n} + 1$, of length 1, then $\sqrt{n} - 2$ independent sets, in each \sqrt{n} vertices with length 1, at the end an independent set of size $\sqrt{n} - 1$, such that all vertices have the length p . In other words, A colors the independent sets $IS_1, \dots, IS_{\sqrt{n}-2}$, and finally, all the vertices in S_1 . Figure 4.2 is a sketch of this coloring.

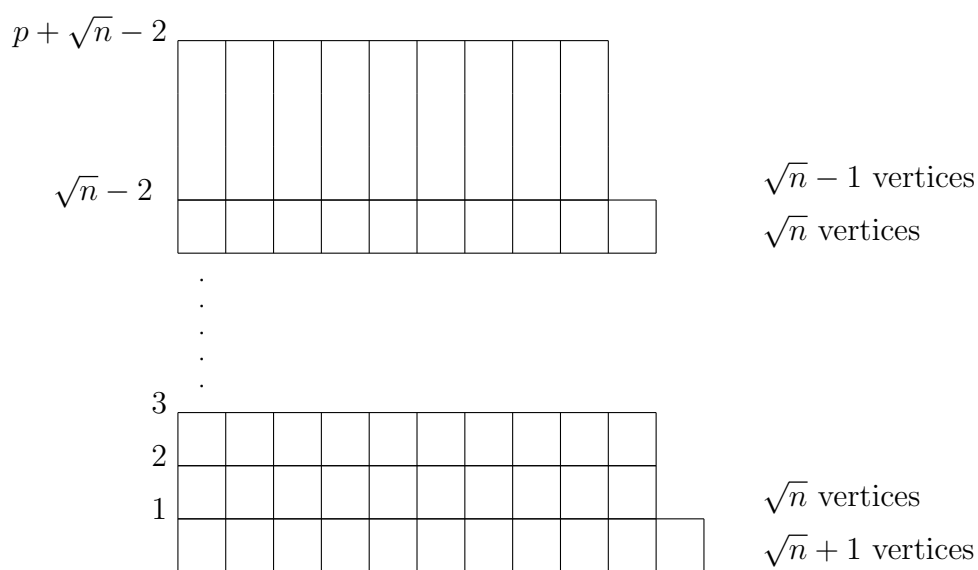


Figure 4.2: The coloring of the graph \tilde{G} under A .

The costs of the two algorithms are given by:

$$\begin{aligned}
A(G) &= \sqrt{n} + 1 + \sum_{i=2}^{\sqrt{n}-1} i\sqrt{n} + (\sqrt{n}-1) \cdot (\sqrt{n}-1+p) \\
&= 2 - p + p\sqrt{n} - 2\sqrt{n} + \frac{n\sqrt{n}}{2} + \frac{n}{2},
\end{aligned}$$

and

$$\begin{aligned}
MMaxIS(G) &= p \sum_{i=1}^{\sqrt{n}-1} i + (\sqrt{n} + 1) + \sqrt{n} \sum_{i=1}^{\sqrt{n}} (p \cdot i + \sqrt{n}) \\
&= pn - \frac{p\sqrt{n}}{2} + \sqrt{n} + 1 + n + \frac{pn\sqrt{n}}{2}.
\end{aligned}$$

Hence, the approximation ratio is:

$$\frac{MMaxIS(G)}{npSMC(G)} \geq \frac{MMaxIS(G)}{A(G)} = \frac{pn - \frac{p\sqrt{n}}{2} + \sqrt{n} + 1 + n + \frac{pn\sqrt{n}}{2}}{2 - p + p\sqrt{n} - 2\sqrt{n} + \frac{n\sqrt{n}}{2} + \frac{n}{2}}$$

If $p = O(n)$, we have

$$\frac{MMaxIS(G)}{A(G)} \rightarrow p.$$

For $p = \Omega(n)$, we have

$$\frac{MMaxIS(G)}{A(G)} \rightarrow n.$$

Thus,

$$\frac{MMaxIS(G)}{npSMC(G)} = \Omega(\min(n, p)).$$

This completes the proof. □

Chapter 5

Exact Solutions for the SMC Problem

In this chapter we give optimal algorithms for the SMC problem on trees and stars. In section 5.1 we introduce an algorithm for the npSMC on trees. In section 5.2 we present algorithms that solve optimally the SMC problem in the three models, for the case where the graph G is a star.

5.1 The Tree-Color Algorithm

5.1.1 Notation and Data Structures

Given a rooted tree T , let T_v denote the subtree rooted at v , $kids(v)$ denote the set of children of v , $par(v)$ its parent, and $N(v)$ its set of neighbours (that is, children plus parent).

Let

$$B(v) = x(v) + \sum_{u \in N(v)} (x(u) + x(v) - 1) .$$

Note, that the finish time of v under any “reasonable” coloring is at most $B(v)$, since each neighbor u of v can delay the completion of v by at most $x(u)$ steps, from its own length, plus $x(v) - 1$, from leaving a “gap” in the set of available colors for v .

We call a finishing time $f(v)$ for vertex v an *optimal finish time*, if there exists an optimal coloring of T_v where v has this finishing time.

We say that $f(v)$ *interferes* with a child u of v , if the respective coloring of v intersects with the coloring which corresponds to u 's optimal finish time in T_u .

We denote by $[x, y]$ the interval of natural numbers $\{x, x + 1, \dots, y\}$.

The data used by the algorithm *Tree-color* will be kept in two vectors, $incr[B(v)]$ and $alt_v[2p]$:

- The vector $incr[B(v)]$ gives the increase in the cost of the optimal coloring of T_v , for each possible finish time for v . The increase is computed relative to initial coloring of T_v , in which the algorithm assumes that $x(v) = 0$.
- The vector $alt_v[2p]$, contains alternative finish times for v , and the respective increase in the cost of the optimal coloring of T_v for each. (Again, assuming initially that $x(v) = 0$). Whenever $par(v)$ chooses a finish time that interferes with v , the coloring of v is changed to the next alternative finish time.

5.1.2 Informal Description of the Algorithm

The algorithm *Tree-color* proceeds by arbitrarily rooting the tree from some vertex r and then coloring it from the leaves up to the root. The coloring of v involves two tasks:

- (a) Finding an optimal finish time of v and the corresponding minimum sum of a multi-coloring of T_v .
- (b) Preparing a set of size at most $x(v) + x(par(v)) - 1 < 2p$, of alternative finish times for v , in the event that $par(v)$ chooses a finish time that interferes with v : $par(v)$ interferes with v if it gets any color in the range $[f(v) - x(v) + 1, f(v) + x(par(v)) - 1]$.

5.1.3 Efficient Implementation of the Tree-color Algorithm

We describe below how the algorithm *Tree-color* can be implemented using the vectors $incr$ and alt_v ; we then show that *Tree-color* solves the npSMC on trees optimally in $O(n \cdot p)$ steps.

Each vertex v computes the vectors $incr$ and alt_v in three phases:

- (i) In the first phase, v initializes the vector $incr$ with values appropriate for the case that no collisions occur with the optimal colors of its children. That is, $incr[i].finish_time = incr[i].cost = i + x(v) - 1$, $1 \leq i \leq B(v) - x(v) + 1$.

- (ii) In the second phase, v updates the vector $incr$: for each child $u \in kids(v)$, v adds in at most $x(u) + x(v) - 1$ entries of the vector $incr$ the increase in the cost of the optimal coloring of T_u (This is done for any finish time of v that is bad for u).
- (iii) Finally, v finds in $incr$ the optimal and alternative finish times. This is done by first defining an interval $[first, last]$ of finish times for $par(v)$, which are bad for v ; the sorted vector $incr$ is then examined, starting from the first entry. For any $s \equiv f(par(v)) \in [first, last]$, we find the smallest $i > 1$, such that $incr[i].finish_time$ does not collide with s , and choose this value to be the alternative finish time of v for s .

Figure 5.1 is a pseudo-code of the algorithm *Tree-color*.

Note, that in order to prepare the vector alt_v in phase (iii), the algorithm *Tree-color* calls the procedure *Counting-sort*: this procedure sorts the entries of the vector $incr$ in non-decreasing order, by the cost field. Thus, in the resulting vector we have $incr[1].cost \leq \dots \leq incr[B(v)].cost$. The procedure implements the standard *Counting-sort* [CLR]: Given an integer array of length t , in which all elements are in the range $[1, k]$, *Counting-sort* can sort the elements in $O(k + t)$ steps. Indeed, when $k = O(t)$ its running time becomes $O(t)$. We show below that this is the case in our algorithm. The basic idea of *Counting-sort* is to determine, for each input element x , the number of elements that are smaller than x . This information can be used to put x directly into its position in the output array.

Algorithm *Tree-color*:

// Pseudocode for vertex v .

for $i \leftarrow 1$ to $B(v) - x(v) + 1$ do

$incr[i].finish_time \leftarrow i + x(v) - 1$

$incr[i].cost \leftarrow i + x(v) - 1$

The first phase

for $u \in kids(v)$ do

$g(u) \leftarrow \max(f(u) - x(u) + 1, x(v))$

 for $l \leftarrow g(u) - x(v) + 1$ to $f(u)$ do

$incr[l].cost \leftarrow incr[l].cost + alt_u[l - g(u) + x(v)].cost$

The second phase

Counting_sort($incr$)

$f(v) \leftarrow incr[1].finish_time$

$first = \max(f(v) - x(v) + 1, x(par))$

$last = f(v) + x(par(v)) - 1$

$i = 1$

while (alt_v is not completely filled in the range $[first, last]$) do

$S_i \leftarrow \{f(par(v)) \in [first, last] \mid f(par(v))$

 does not collide with $incr[i].finish_time\}$

 for $s \in S_i$ do

$alt_v[s].finish_time \leftarrow incr[i].finish_time$

$alt_v[s].cost \leftarrow incr[i].cost - incr[1].cost$

$i = i + 1$

The third phase

Figure 5.1: The *Tree-color* Algorithm

Theorem 5.1 *Tree-color solves npSMC on trees in $O(np)$ steps.*

Proof: We first show the correctness and optimality of the algorithm. We then analyze its complexity.

Correctness: We have to satisfy the following two conditions: (i) *mutual exclusion*, (ii) *no starvation*. The first condition implies that no two conflicting jobs are executed simultaneously, that is a father and a son in the tree cannot get intersecting sets of colors. For any vertex $v \in V$, the second phase in the algorithm modifies the colors of the sons of v to prevent collisions, by that condition (i) is satisfied. The second condition means that the request of any job to run has to be granted. We satisfy this condition by passing over all the vertices from the leaves up to the root assigning to every vertex, in the third phase a finishing time $f(v)$, which means that $f(v)$ is the last among the $x(v)$ colors assigned to v .

Optimality: The optimality of the algorithm follows from the fact that each $v \in V$ finds a finish time, which minimizes the cost of the optimal coloring of T_v . This holds recursively for any subtree of T . Formally, for any finish time $x(v) \leq i \leq B(v)$, the *multi-color sum* of T_v is given by

$$SMC(T_v, i) = i + \sum_{j=1}^{|kids(v)|} SMC(T_j, n_j(i))$$

where $n_j(i)$ is the finish time of child j , which minimizes the cost of the optimal coloring of T_j , when the finish time of v is i . Thus, we have, that

$$SMC(T_v) = \min_{x(v) \leq i \leq B(v)} SMC(T_v, i) .$$

Since we color the vertices in the tree from the leaves upwards, the *multi-color sum* of T is $SMC(T_r)$. Hence the algorithm Treecolor follows a dynamic programming scheme which guarantees the optimality of the solution in each stage, in particular the last one.

Complexity: For the complexity of the algorithm, we analyze separately each phase, when performed by a single vertex v .

- The first phase takes $O(B(v))$ steps.
- In the second phase, for each child u of v , at most $x(u) + x(v) - 1$ entries are updated in the vector *incr*. Recall, that v is bad for its child u , if v gets the values that start from $\max(f(u) - x(u) + 1, x(v))$ and end in $f(u) - x(v) - 1$, i.e., at most $x(u) + x(v) - 1$ values. Summing over $kids(v)$ we get that the complexity of this phase is $O(B(v))$.

- The complexity of the final phase is composed of the sorting of the vector $incr$ and of filling the vector alt_v .

(a) We now analyze the sub-phase of sorting.

Claim 5.1 *The value of each entry in the vector $incr$ at the end of phase (ii) is bounded by*

$$N_v = B(v) + p(\deg(v) + \sum_{u \in \text{kids}(v)} \deg(u)). \quad (5.1)$$

Proof: Note, that in the worst case, each son of v , u , modifies its finish time to $p(1 + |\text{kids}(u)|)$. Hence, the additional cost incurred by conflicts of v with its children is at most $p(\deg(v) + \sum_{u \in \text{kids}(v)} \deg(u))$. Also, initially (i.e., after the first phase) each entry in $incr$ is bounded by $B(v)$. This yields the claim. \square

Hence, the complexity of sorting $incr$ by a single vertex v is $O(B(v) + N_v)$.

- (b) It remains to bound the complexity of filling alt_v , $\forall v \in V$. We now show that the vector alt_v is filled in $O(p)$ steps. The vector alt_v contains the alternative finish times for v , when the finish times of $par(v)$ are in the interval $[first, last]$, where:

$$first = \max(f(v) - x(v) + 1, x(par(v)))$$

and

$$last = f(v) + x(par(v)) - 1$$

The vector alt_v is filled by examining a set of consecutive entries in the sorted vector $incr$, starting from the first entry. For each value $s \in [first, last]$ we fit the first value $incr[i].finish_time$, satisfying:

$$incr[i].finish_time < s - x(par(v) + 1)$$

or

$$incr[i].finish_time - x(v) + 1 > s.$$

This means that $incr[i].finish_time$ is not selected, when $s - x(par(v) + 1) \leq incr[i].finish_time \leq s + x(v) - 1$.

Consequently, at most $x(v) + x(par(v)) - 1$ values are not suitable. It follows, that we need to examine at most $2p$ values in the vector $incr$ till alt_v is filled.

Finally, we need to sum up the complexity of each phase over all the vertices. Note, that the complexity of *Tree-color* is dominated by the subphase of sorting. Hence we need to show the following claim.

Claim 5.2 *The overall complexity of sorting is $O(np)$.*

Proof: It is sufficient to show, that $\sum_{v \in V} (B(v) + N_v) = O(np)$. We first show that $\sum_{v \in V} B(v) = O(np)$:

$$\begin{aligned} \sum_{v \in V} B(v) &= \sum_{v \in V} (x(v) + \sum_{u \in N(v)} (x(u) + x(v) - 1)) \\ &\leq np + \sum_{v \in V} \sum_{u \in N(v)} 2p \\ &= np + 2p \cdot 2|E| = O(np) \end{aligned}$$

The last equality follows from the fact that $|E| = n - 1$. In addition,

$$\begin{aligned} \sum_{v \in V} N_v &= \sum_{v \in V} B(v) + \sum_{v \in V} p(\deg(v) + \sum_{u \in kids(v)} \deg(u)) \\ &= \sum_{v \in V} B(v) + 2|E|p + |E|p = O(np) \end{aligned}$$

and the claim follows. □

This yields the statement of the theorem. □

Corollary 5.1 *If each vertex requires exactly p colors, then *Tree-color* can be implemented in $O(n)$ steps.*

Proof: This case is equivalent to the SC problem, where one color represents a ‘block’ of colors of length p . Clearly, it is sufficient to consider in each pass on the vector *incr* only entries of finish times that are integral multiples of p . Otherwise, we get ‘gaps’ of colors of lengths less than p ; indeed, no vertex can use these colors. We can eliminate the ‘gaps’ iteratively by shifting the vertices that start immediately after a ‘gap’ to start where the ‘gap’ starts, thus reducing the sum. The coloring is still legal, since there cannot be any collision among neighboring vertices. Hence, we obtained an optimal coloring, in which the first color of each vertex is an integral multiple of p . □

5.2 Algorithms for the SMC on Stars

We give below $O(\min(p + n, n \log n))$ algorithms for npSMC, coSMC and pSMC on stars. Let $G = (V, E)$ be a star, and let v_n denote the vertex having $n - 1$ neighbours (We call this vertex the *center*).

5.2.1 An algorithm for npSMC and coSMC on Stars

The algorithm runs in three phases:

- (1) Order the leaves in non-decreasing order by color requirements and mark them by $v_1, v_2, v_3, \dots, v_{n-1}$.
- (2) Compute the multi-color sum in the following cases:
 - The finish time of v_n is $x(v_n)$ and the finish time of v_i is $x(v_n) + x(v_i)$ for $1 \leq i \leq n - 1$.
 - The finish time of v_i in its turn, is $x(v_i)$, the finish time of v_n is $x(v_n) + x(v_i)$, and for any $j \neq n, i$

$$f(v_j) = \begin{cases} x(v_j) & j < i \\ x(v_n) + x(v_i) + x(v_j) & \text{otherwise} \end{cases}$$

Repeat for all $1 \leq i \leq n - 1$.

- (3) Find the minimum among all the sums computed in phase (2).

Figure 5.2 gives the pseudo-code of the algorithm *Star-color*.

Theorem 5.2 *Star-color solves npSMC on stars in $O(\min(p + n, n \log n))$ steps.*

Proof: We show separately the optimality and the complexity of the algorithm.

- The optimality of the *Star-color* follows from the fact that we examine all the reasonable non-preemptive colorings. Note, that in any reasonable (contiguous) coloring, the vertices are partitioned to at most three subset: V_1 , V_2 and V_3 .

V_1 =vertices which complete before v_n

V_2 =vertices which complete after v_n

V_3 = v_n .

There are n such partitions.

Algorithm *Star-color*:

Sort the leaves in non-decreasing order
by color requirements

The first phase

$S_0 \leftarrow nx(v_n)$

The second phase

$SMC \leftarrow S_0$

$r \leftarrow 0$

for $i \leftarrow 1$ to $n - 1$ do

$S_i \leftarrow (n - i)(x(v_n) + x(v_i))$

(*)

$SMC \leftarrow \min(SMC, S_i)$

The third phase

if $SMC = S_i$ then $r \leftarrow i$

$SMC \leftarrow SMC + \sum_{i=1}^{n-1} x(v_i)$

The coloring of the vertices

if $r = 0$ then

$f(v_n) = x(v_n)$

for $i \leftarrow 1$ to $n - 1$ $f(v_i) = x(v_i) + x(v_n)$

else

for $j \leftarrow 1$ to r do $f(v_j) = x(v_j)$

$f(v_n) = x(v_r) + x(v_n)$

for $j \leftarrow r + 1$ to $n - 1$

$f(v_j) = x(v_r) + x(v_n) + x(v_j)$

Figure 5.2: The *Star-color* Algorithm

- To prove the complexity we analyze each phase.
 - In the first phase we sort the $n - 1$ leaves. If $p = O(n)$ we will use **Count-Sort** algorithm with complexity $O(n)$ [CLR], else we can use the **Quick-Sort** algorithm [CLR], which runs in $O(n \lg n)$ steps. Overall, the first phase takes $O(\min(p + n, n \log n))$ steps.
 - In the second phase we compute n sums, for the n possible colorings.
 - In the last phase we find the minimum among n sums, again, in $O(n)$ steps.

Hence, the complexity of **Star-color** is determined by the sorting phase, which is $O(\min(p + n, n \log n))$. \square

Theorem 5.3 *The algorithm Star-color solves optimally the coSMC on stars.*

Proof: Assume by contradiction that the solution for npSMC does not solve coSMC. This means that there exist at least two leaves v_i, v_j , such that v_i, v_j have in common at least one color, denote it by z . For one of these vertices say, v_1 , z is not the first color, assume that v_1 starts with y , such that $y < z$.

The central vertex, v_n cannot get the colors $y+1 \rightarrow z-1$, since it is connected to all the vertices, in particular to v_1 . Hence, we could start coloring both v_1 and v_2 with y and in this way reduce the sum. It follows, that coSMC is equivalent to npSMC on stars. \square

5.2.2 An Algorithm for pSMC on Stars

The next two lemmas show, that the algorithm *Star-color* can be used (with slight modification) for solving the pSMC.

Lemma 5.4 *There exists an optimal coloring which colors v_n contiguously.*

Proof: Assume, that every optimal coloring splits v_n . We can make the coloring of v_n contiguous by not changing the last color and attaching the other colors to the last. We show, that such a change will not increase the *multi-color sum*.

Note, that there are three types of leaves:

- (1) Those that completed before v_n started will retain their finish time.

- (2) Those that completed after v_n , again, remain with the same endpoint.
- (3) Those that completed in the ‘split’ of v_n , will now complete earlier.

Overall, the sum after the change cannot increase. □

Lemma 5.5 *If v_n does not start with 1, it will start immediately after some leaf, which is colored contiguously.*

Proof: Assume by contradiction, that v_n does not start after a contiguously colored leaf. The sum can be reduced by putting v_n before the leaves that it splits. If v_n is still not after a leaf that is colored contiguously, we will continue to put v_n before leaves: we will stop, either after a leaf which is colored contiguously or when v_n starts with 1 (see Figure 5.3). □

$$\begin{array}{|c|c|c|} \hline v_i & v_n & v_i \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|} \hline v_n & v_i \\ \hline \end{array} \quad (1)$$

$$\begin{array}{|c|c|c|c|} \hline v_j & v_i & v_n & v_i \\ \hline \end{array} \quad \rightarrow \quad \begin{array}{|c|c|c|} \hline v_j & v_n & v_i \\ \hline \end{array} \quad (2)$$

Figure 5.3: Sketch of Lemma 5.5

The algorithm is similar to the npSMC algorithm, up to the sum (since it allows to split vertices). Line (*) in Figure 5.2 will change to:
 $S_i = (n - i)x(v_n) + x(v_i)$. We call the resulting algorithm *pStar-color*.

Theorem 5.6 *pStar-color solves pSMC on stars in $O(\min(p + n, n \log n))$ steps.*

Proof: The proof follows the steps of the proof of Theorem 5.2, using Lemmas 5.4 and 5.5. □

Chapter 6

Summary

In this chapter we summarize with the contribution of this work and some directions for future research.

6.1 The Contribution of this Work

In this work we introduced the *sum multi-coloring* problem, and developed algorithms for solving it. The most important achievement of this work is in finding exact algorithms for the SMC problem:

In Chapter 5, we gave an algorithm, which solves the npSMC on trees, and exact solutions in the three models for stars. These are the first exact solutions for the SMC problem. In the past, there were exact algorithms only for the SC problem.

The other results that we obtained for the SMC problem provided approximate solutions for the SMC, and for two variants of the SC problem.

6.2 Open Problems

Until recently, there has been an on-going study only of the SC problem, which is a special case of the SMC. Following this work, there are many directions for further study. We list some of the open problems below.

- Are there additional special graphs (other than trees), for which the SMC problem can be optimally solved, in polynomial time?
Can we use our algorithm for trees on these graphs?
- We solved the npSMC problem on trees. Are the pSMC and the coSMC solvable in polynomial time on trees?

- We have given an algorithm which solve the SMC in the three models on stars. Can similar algorithm be devised for the SMC, on paths?

Publications

A. Bar-Noy, M.M. Halldórsson, G. Kortsarz, R. Salman and H. Shachnai, "Sum Multi-Coloring of Graphs", to appear in *Proceedings of the Seventh Annual European Symposium on Algorithms*, Prague, July 1999.

M.M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai and J. A. Telle, "Multi-Coloring Trees", to appear in *Proceedings of the fifth Annual International Computing and Combinatorics Conference*, Tokyo, July 1999.

Bibliography

- [B92] M. Bell. Future directions in traffic signal control. *Transportation Research Part A*, 26:303–313, 1992.
- [BBH⁺98] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140:183–202, 1998.
- [BH94] D. Bullock and C. Hendrickson. Roadway traffic control software. *IEEE Transactions on Control Systems Technology*, 2:255–264, 1994.
- [BKH⁺98] A. Bar-Noy, M. M. Halldórsson, G. Kortsarz, R. Salman and H. Shachnai. Sum Multi-Coloring of Graphs. To appear in *Proceedings of the Seventh Annual European Symposium on Algorithms*, Prague, July 1999.
- [BK98] A. Bar-Noy and G. Kortsarz. The minimum color-sum of bipartite graphs. *Journal of Algorithms*, 28:339–365, 1998.
- [CCO93] J. Chen, I. Cidon and Y. Ofek. A local fairness algorithm for gigabit LANs/MANs with spatial reuse. *IEEE Journal on Selected Areas in Communications*, 11:1183–1192, 1993.
- [CLR] Corman T., Leiserson C., Rivest R., *Introduction to Algorithms*, MIT Press, 1990.
- [FK96] U. Feige and J. Kilian. Zero Knowledge and the Chromatic number. *Journal of Computer and System Sciences*, 57(2):187-199, October 1998.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [HK99] M. M. Halldórsson and G. Kortsarz. Multi-Coloring Planar Graphs and Partial k -trees. To appear in *Proc. Second International Workshop on Approximation Algorithms Approx-99*.

- [HKP⁺99] M.M. Halldórsson, G. Kortsarz, A. Proskurowski, R. Salman, H. Shachnai and J. A. Telle. Multi-Coloring Trees. To appear in *Proceedings of the fifth Annual International Computing and Combinatorics Conference*, Tokyo, July 1999.
- [IL96] S. Irani and V. Leung. Scheduling with Conflicts, and Applications to Traffic Signal Control. *Proceedings of the Seventh ACM-SIAM Symposium on Discrete Algorithms*, 1996.
- [J97] K. Jansen. The Optimum Cost Chromatic Partition Problem. *Proc. of the Third Italian Conference on Algorithms and Complexity (CIAC '97)*. LNCS 1203, 1997.
- [K89] E. Kubicka. The Chromatic Sum of a Graph. PhD thesis, Western Michigan University, 1989.
- [KKK89] E. Kubicka and G. Kubicki, and D. Kountanis. Approximation Algorithms for the Chromatic Sum. *Proceedings of the First Great Lakes Computer Science Conf.*, Springer LNCS 507, pp. 15–21, 1989.
- [KS89] E. Kubicka and A. J Schwenk. An Introduction to Chromatic Sums. *Proceedings of the ACM Computer Science Conf.*, pp. 39-45, 1989.
- [L81] N. Lynch. Upper Bounds for Static Resource Allocation in a Distributed System. *J. of Computer and System Sciences*, 23:254–278, 1981.
- [NS93] M. Naor and L. Stockmeyer. What Can be Computed Locally? *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, pp. 184–193, 1993.
- [NSS94] S. Nicoloso, M. Sarrafzadeh, and X. Song. On the Sum Coloring Problem on Interval Graphs. Istituto di Analisi dei Sistemi ed Informatica (IASI-CNR), R. 390, 1994.
- [SP88] E. Steyer and G. Peterson. Improved Algorithms for Distributed Resource Allocation. *Proceedings of the Seventh Annual Symposium on Principles of Distributed Computing*, pp. 105–116, 1988.
- [SG98] A. Silberschatz and P. Galvin. Operating System Concepts. Addison-Wesley, 5th Edition, 1998.
- [T95] A. S. Tannenbaum, *Distributed Operating Systems*. Prentice-Hall Int., Editing, 1995.