A polynomial time list colouring algorithm for series-parallel graphs

Jeannette Janssen
Philippe Maheux

Technical Report CS-2002-02

April, 2002

Faculty of Computer Science
6050 University Ave., Halifax, Nova Scotia, B3H 1W5, Canada

A polynomial time list colouring algorithm for series-parallel graphs

Jeannette Janssen Philippe Maheux

May 1, 2002

Abstract

Given a graph and an assignment of lists to its nodes, a list colouring is an assignment of colours to the nodes of the graph, so that adjacent nodes receive different colours and each colour receives a colour from its list. The question is: given a particular graph and list assignment, does a list colouring exist? An algorithm is presented here that answers this question for series-parallel graphs. The algorithm has complexity $\mathcal{O}(md^2)$, where m is the number of edges, and d is the maximum degree of the graph. A necessary and sufficient condition for the existence of a list colouring is given, which is based on a recursively defined function. The analysis of the necessity of this condition leads to a set of valid inequalities for the list colouring polytope.

1 Introduction

Graph colouring is a theoretical problem which is at the heart of a great number of diverse applications. Examples are: frequency assignment in cellular networks [11], timetabling, [2], and routing in optical networks [9]. A graph colouring problem arises whenever a limited number of resources (frequency channels, time slots) must be assigned to a number of entities, subject to constraints on the simultaneous assignment of a resource to certain pairs of entities. A graph model can be used to model these constraints.

In a situation as just described, it often happens that the list of resources available to a particular entity is restricted. For example, in a cellular network certain frequencies may be forbidden from use at certain transmitters, due to interference with, say, a military base. Also, in any situation where an existing assignment is being expanded, the resources available to an entity will be limited by possible conflict with the existing assignment. This leads to a list colouring problem.

A list colouring problem is defined by a graph and an assignment of sets of allowed colours to the nodes of the graph. The sets of allowed colours are called the lists. The goal is to find an assignment of colours to the nodes of the graph, so that each node receives a colour from its list, and adjacent nodes receive different colours. Most work on list colourings has focused on determining the list-chromatic number of a graph, which is the minimal list size so that a list colouring is guaranteed to exist for any assignment of lists of that size. An overview of results in this direction can be found in [13].

However, the existence of a list colouring does not only depend on size of the lists, but also on their content. Even if a particular list assignment does not adhere to the minimal list size prescribed by the list-chromatic number of the graph, then a list colouring for that particular assignment may still exist. Moreover, in practical situations the size of the lists may be beyond a planner's control. Therefore, this paper addresses the following decision problem:

Given a graph and a particular list assignment for that graph, does a list colouring exist?

It has been shown in [8] that this problem is NP-complete even under quite restrictive conditions on the lists. In [10] a simple condition for the existence of a list *edge* colouring is given, and it is shown that this condition is sufficient only for trees and for a limited class of related graphs called multiforests. In [3], an elegant algorithm to find a list colouring for trees is given. The equivalent condition for *node* list colourings is discussed in [1], where it is referred to as the Hall condition.

This paper gives a necessary and sufficient condition for the existence of a list colouring of a series-parallel graph. The condition is based on a recursively defined function that can be computed in polynomial time. The function essentially encodes the effect of a colour used at a node on colours available to other nodes. If the condition holds, then the same function can be used to find a list colouring, again in polynomial time. The necessity of the condition depends on the analysis of the list colouring polytope: the convex hull of the 0/1 vectors that represent list colourings of the graph or an induced subgraph.

Series-parallel graphs are well-studied in terms of colourings. In [12] results on edge colouring and a simple algorithm for node colouring of series-paralllel graphs are given, while [14] contains an edge colouring algorithm. Juvan *et al.* in [5] establish the minimum list size required for list colourings of the *edges* of a series-parallel graph, and give a linear time algorithm to find such colourings for lists of the prescribed size.

2 Series-Parallel Graphs

A *series-parallel graph* is a graph that can be obtained by a sequence of edge operations, series and parallel, from a graph consisting of only one edge. A *parallel operation* on an edge e replaces e by two (parallel) edges with the same endpoints as e. A *series operation* on an edge e replaces the edge by a path of length two. The two endpoints of the original edge are called the *terminals* of the series-parallel graph. Note that the edge operations do not affect existing nodes, and that all nodes except the terminals are created through series operations.

The construction of a series-parallel graph G can be represented by a set of *ur-edges* U and a *parse tree* T. The ur-edges are the edges used in the construction of G. The set U can be partioned into three sets, P, S and E. The edges in P are the edges that were replaced by a parallel operation, the edges in S are the ones that were replaced by a series operation, and E contains the final edges, the edges that are still present in G. The parse tree T is a binary rooted tree with node set U. The root of T corresponds to the *root edge* e_0, the original edge between the terminals s and t from which G was constructed. The edges in E correspond to the leaves of T. For every ur-edge $e \in P$ or $e \in S$, its children represent the two edges by which it was replaced after a parallel or series operation, respectively.

A parse tree of a given series-parallel graph, implying a possible construction sequence, can be obtained in linear time. An algorithm to do so is described in [6], and can be derived with minor modifications from the algorithm given in [4]. Several parallel algorithms have also been described; see [7] for a recent example.

Our list colouring algorithm uses an orientation of the series-parallel graph that is consistent with an orientation of the root edge from s to t. An edge e directed from u to v will have v as its *head*, denoted by $v = h(e)$, and u as its *tail*, denoted by $t(e)$.

Given a series-parallel graph G with parse tree T, and an ur-edge e of G, the graph G_e is the series-parallel graph represented by the subtree of the parse tree of G with root e. Obviously, G_e is a subgraph of G. The expressions $V(G)$ and $E(G)$ will be used to denote the node set and the edge set of of a graph G, respectively.

3 List-Colouring

A *list assignment* for a graph G is a collection of sets $\mathcal{L} = \{L_v \,|\, v \in V(G)\}$. The list L_v can be interpreted as the set of colours that may be used to colour node v. The *colour set* of a list assignment $\mathcal{L}$ is the set $\bigcup_{L_v \in \mathcal{L}} L_v$. A *list colouring* of a graph G with list assignment $\mathcal{L}$ is an assignment of colours to the nodes of G so that each node v receives a colour from L_v, and adjacent nodes receive different colours.

In order to estimate the complexity of the algorithm presented in this paper, we need an upper bound on the size of the lists of a list assignment. Suppose that a node v is assigned a list L_v of size larger than its degree $d(v)$. Then, for any list colouring, L_v always contains at least one colour that is not used to colour any of its neighbours. Hence, if L_v is replaced by a list containing just one colour which is not present in any of the other lists, then any list colouring obtained with this new list for v could be transformed into a list colouring for the original lists by replacing v's colour by a colour from L_v not used at v's neighbours. This is stated formally in the following claim.

Claim 3.1 *Let G be a series-parallel graph, and $\mathcal{L}$ a list assignment for G. Let $\{\infty_v | v \in V(G)\}$ be a set of colours disjoint from the colour set of $\mathcal{L}$, and let $\mathcal{L}'$ be the list assignment obtained from $\mathcal{L}$ by replacing L_v by $L'_v = \{\infty_v\}$ for every node $v \in V(G)$ for which $|L_v| > d(v)$. Then $(G, \mathcal{L})$ has a list colouring if and only if $(G, \mathcal{L}')$ has a list colouring.*

The claim implies that we may assume without loss of generality that $|L_v| \leq d(v)$ for all nodes $v \in V(G)$ in any list colouring problem $(G, \mathcal{L})$.

A list colouring can be represented by a 0-1 vector indexed by the nodes and colours. Specifically, given a list assignment $\mathcal{L}$ for a graph G, let $V_{\mathcal{L}} = \{(v, \gamma) \,|\, v \in V(G),\ \gamma \in L_v\}$. Then any list colouring of $(G, \mathcal{L})$ can be represented by a vector $\mathbf{x} \in \{0,1\}^{V_{\mathcal{L}}}$ in the following manner: $x_{v,\gamma} = 1$ iff node v receives colour γ.

From the definition it follows that a vector $\mathbf{x} \in \mathbb{Z}^{V_{\mathcal{L}}}$ representing a list colouring is a feasible solution to the following integer linear Inequality System (IS):

IS for List Colouring

$$\sum_{\gamma \in L_v} x_{v,\gamma} \leq 1 \qquad \text{for all } v \in V(G) \tag{1}$$

$$x_{u,\gamma} + x_{v,\gamma} \leq 1 \quad \text{for all adjacent pairs } u, v \in V(G), \text{ all } \gamma \in L_u \cap L_v \tag{2}$$

$$x_{v,\gamma} \in \{0,1\} \qquad \text{for all } v \in V(G),\ \gamma \in L_v \tag{3}$$

In fact, any feasible solution to the IS represents a list colouring if and only if equality is achieved in (1) for all $v \in V(G)$. Any feasible solution to the IS which does not necessarily achieve inequality in (1) represents a partial list colouring. A *partial list colouring* of a graph G with list assignment $\mathcal{L}$ is a list colouring of $(H, \mathcal{L}_H)$, where H is an induced subgraph of G, and $\mathcal{L}_H$ is the list assignment $\{L_v \,|\, v \in V(H)\}$ induced by $\mathcal{L}$ on H.

The vector representations derived here will be used in Section 5 to derive a condition for non-existence of a list colouring.

4 The List Colouring Algorithm

In this section we present our list colouring algorithm for series-parallel graphs. The idea behind the algorithm is to compute which colours can be used on terminal t if a particular colour is used on terminal s. The computation is done in an iterative fashion starting from the leaves of the parse tree. A set-valued function A is created, where $A(e, \gamma)$ represents the set of colours allowed at $t(e)$ if colour γ is used at $h(e)$. We give a recursive definition of A below, and will prove later that this formal definition coincides with the intuitive one.

Definition 4.1 *Let G be a series-parallel graph represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G with colour set Γ. The function $A_G : \{(e, \gamma) : e \in U,\ \gamma \in L_{t(e)}\} \to 2^\Gamma$ is defined as follows:*

- *For any $e \in E$, and any $\gamma \in L_{t(e)}$, $A_G(e, \gamma) = L_{h(e)} - \{\gamma\}$,*
- *For any $e \in P$ with children e_1 and e_2, and any $\gamma \in L_{t(e)}$,*

$$A_G(e, \gamma) = A_G(e_1, \gamma) \cap A_G(e_2, \gamma),$$

- *For any $e \in S$ with children e_1 and e_2, and any $\gamma \in L_{t(e)}$,*

$$A_G(e, \gamma) = \bigcup_{\beta \in A_G(e_1, \gamma)} A_G(e_2, \beta).$$

For brevity, $A(e, \gamma)$ is used to denote $A_G(e, \gamma)$ whenever it is clear from the context which graph G the function A_G refers to.

The computation of A_G is at the heart of list colouring algorithm. Moreover, as the following theorem shows, by computing A_G it can be determined whether or not a list-colouring exists.

Theorem 4.2 *Given are a series-parallel graph G with terminals s and t and root edge e_0, represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and a list assignment $\mathcal{L}$ for G. Let the function A_G be as defined above. Then a list colouring of $(G, \mathcal{L})$ exists precisely when*

$$\bigcup_{\gamma \in L_s} A_G(e_0, \gamma) \neq \emptyset.$$

The proof of this theorem will follow from the results in the remainder of this paper, and will be based on the following algorithm for *List Colouring Series-Parallel graphs* (LCSP).

LCSP Algorithm

INPUT: A series-parallel graph G with terminals s and t and root-edge e_0, represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and a list assignment $\mathcal{L}$ for G.
OUTPUT: A valid list-colouring of G, if it exists.

```
1. Recursively compute $A(e,\gamma)$ for all $e \in U$, $\gamma \in L_{t(e)}$.
2. If $\bigcup_{\gamma \in L_s} A(e_0,\gamma) = \emptyset$ then report that the graph is not list
   colourable and exit.
3. Choose $\gamma \in L_s$ and $\beta \in A(e_0,\gamma)$. Assign colour $\gamma$ to node $s$, and
   colour $\beta$ to node $t$.
4. Consider all ur-edges in $S$, in an order consistent with their distance from the root in $T$. For every $e \in S$:
    4a. Let $\gamma$ and $\beta$ be the colours of $t(e)$ and $h(e)$, respectively. Let
        $e_1$ and $e_2$ be the children of $e$, and let $w = h(e_1) = t(e_2)$
        be the node created by the subdivision of $e$. Find a colour
        $\alpha \in A(e_1,\gamma)$ so that $\beta \in A(e_2,\alpha)$. Assign colour $\alpha$ to node
        $w$.
```

Because of the order in which the ur-edges in S are considered in Step 4 of the LCSP algoritm, each edge when considered will have both its endpoints already coloured. Moreover, every node except s and t results from a series operation on an ur-edge in S, so when all edges in S have been considered then all nodes will be coloured. It remains to be shown that a colour α as required in Step 4a will can always be found; this will follow from Claims 4.3 and 4.4 to follow. Claim 4.5 will show that the resulting colouring is indeed a valid list colouring.

Claim 4.3 *If both endpoints of the ur-edge $e \in U$ are coloured by the LCSP Algorithm, and γ and β are the colours assigned to $h(e)$ and $t(e)$, respectively, then $\gamma \in L_{t(e)}$, and $\beta \in A(e,\gamma)$.*

Proof. We prove the claim by induction on the distance of e from the root in the parse tree. If $e = e_0$, the root edge, then the claim follows directly from the first step of the LCSP Algorithm.

Suppose then that the claim holds for ur-edge e. We will show that the claim also holds for its children e_1 and e_2. Let $u = t(e)$, $v = h(e)$, and let γ, β be the colours assigned to u and v, respectively. Suppose first that $e \in P$. Then $t(e_1) = t(e_2) = u$ and $h(e_1) = h(e_2) = v$, and, by induction, $\gamma \in L_u$. By definition, $A(e, \gamma) \subseteq A(e_1, \gamma)$, and by induction, $\beta \in A(e, \gamma)$, so $\beta \in A(e_1, \gamma)$. The same holds for e_2.

Suppose that $e \in S$. Let w be the node created by subdivision of e, where $w = h(e_1) = t(e_2)$, and let α be the colour assigned to w. It follows directly from Step 4 that $\alpha \in A(e_1, \gamma)$ and $\beta \in A(e_2, \alpha)$, as required. From Definition 4.1 it follows easily that $A(e, \gamma) \subseteq L_{h(e)}$ for all $e \in U$, $\gamma \in L_{t(e)}$, so $\alpha \in L_{t(e_1)}$ and $\beta \in L_{t(e_2}$. □

Claim 4.4 *If $\bigcup_{\gamma \in L_s} A(e_0, \gamma) \neq \emptyset$, then upon termination of the LCSP algorithm all nodes have been coloured.*

Proof. We show that, in Step 4a of the algorithm, a colour α satisfying the requirements can always be found. Let $e \in S$ be an ur-edge with $t(e) = u$, $h(e) = v$, where u and v are coloured by the LCSP algorithm with colours γ and β, respectively. By claim 4.3, $\beta \in A(e, \gamma)$. Let e_1, e_2 be the children of e, and $w = h(e_1) = t(e_2)$. By definition, $A(e, \gamma) = \bigcup_{\alpha \in A(e_1, \gamma)} A(e_2, \alpha)$. So there exists a colour $\alpha \in A(e_1, \gamma)$ so that $\beta \in A(e_2, \alpha)$.□

Claim 4.5 *If $\bigcup_{\gamma} A(e_0, \gamma) \neq \emptyset$, then the colours assigned to the nodes by the LCSP algorithm form a valid list colouring.*

Proof. By Claim 4.3, each node v receives a colour from its list L_v. To show that the colouring is valid, consider any edge $e \in E$ with $u = t(e)$, $v = h(e)$, assigned colours γ and β, respectively. By Claim 4.3, $\beta \in A(e, \gamma)$, and by Definition 4.1, $A(e, \gamma) = L_v - \{\gamma\}$. Hence $\beta \neq \gamma$, and thus the colouring is valid. □

Lemma 4.6 *Let G be a series-parallel graph with terminals s and t and root edge e_0, represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G. If $\bigcup_{\gamma \in L_s} A_G(e_0, \gamma) \neq \emptyset$, then the LCSP algorithm finds a valid list colouring of $(G, \mathcal{L})$ in $\mathcal{O}(md^2)$ steps, where m is the number of edges, and d is the maximum degree of G.*

Proof. Let $(G, \mathcal{L})$ be as in the statement of the lemma, and let U be the set of ur-edges of G. Let $m = |E(G)|$ and $n = |V(G)|$. Given that the parse tree is a binary tree with node set U, and that the edges in E correspond to its leaves, we have that $|U| < 2|E| = 2m$.

Also, since every node except s and t is the result of a series operation, we have that $|S| = |V(G)| - 2 = n - 2$. By Claim 3.1, we may assume without loss of generality that $|L_v| \leq d$ for all $v \in V$. To compute A_G, every ur-edge $e \in U$, and every colour $\gamma \in L_{t(e)}$ is considered once. The number of operations required for the computation of $A(e, \gamma)$ is greatest when $e \in S$. In this case, the union of at most d sets, each of size at most d, must be computed. This takes $\mathcal{O}(d^2)$ operations. Since $A(\gamma, e)$ must be computed for each ur-edge e and each colour $\gamma \in L_{t(e)}$, $\mathcal{O}(nd^3)$ operations are required in total for the computation that involves ur-edges from S. Since $2m \leq dn$, $\mathcal{O}(nd^3) = \mathcal{O}(md^2)$. For all other ur-edges, at most $\mathcal{O}(d)$ operations are required to compute $A_G(e, \gamma)$, resulting in an additional $\mathcal{O}(d^2 m)$ operations.

To execute Step 4 of the LCSP algorithm, at most d sets, each of size at most d, must be searched for the presence of colour β. This takes at most d^2 operations. Step 4 is executed n times, so in total this step takes $\mathcal{O}(nd^2)$ operations. □

Note also that at most md sets $A(e, \gamma)$, each of size at most d, must be stored. So the algorithm utilizes $\mathcal{O}(md^2)$ units of storage space.

5 The List Colouring Polytope

The previous section gave the first part of the proof of Theorem 4.2. We will now proceed to show that if the LCSP algorithm halts without returning a colouring, then no valid list colouring exists. This will be done by the establishment of a set of inequalities that are valid for the list colouring polytope.

Definition 5.1 *Given a graph $G = (V, E)$ and a list assignment $\mathcal{L}$ for G, the* list colouring polytope *$\mathcal{P}(G, \mathcal{L})$ is the convex hull of all $0/1$ vectors representing partial list colourings of $(G, \mathcal{L})$.*

An inequality $\mathbf{ax} \leq b$ is *valid* for $\mathcal{P}(G, \mathcal{L})$ if it holds for every vector $\mathbf{x} \in \mathcal{P}(G, \mathcal{L})$. For example, the defining inequalities $x_{u,\gamma} + x_{v,\gamma} \leq 1$, where $u, v \in V$, $u \sim v$, and $\gamma \in L_u \cap L_v$, are valid for $\mathcal{P}(G, \mathcal{L})$.

The following linear functions will form the left-hand sides of our inequalities.

Definition 5.2 *Let G be a series-parallel graph represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G. Let the function A_G be as defined in Definition 4.1. For every ur-edge $e \in U$, the linear function $\ell_{e,\gamma} : \mathbb{R}^{V_{\mathcal{L}}} \to \mathbb{R}$ is defined as follows:*

$$\ell_{e,\gamma}(\mathbf{x}) = x_{t(e),\gamma} + \sum_{w \in V_e^*} \sum_{\alpha \in L_w} x_{w,\alpha} + \sum_{\alpha \in F(e,\gamma)} x_{h(e),\alpha},$$

where $V_e^* = V(G_e) - \{h(e), t(e)\}$, and $F(e, \gamma) = L_{h(e)} - A_G(e, \gamma)$.

The expression $F(e, \gamma)$, as introduced in the preceeding definition will be used in the rest of this section for convenience. So for every $e \in U$ and $\gamma \in L_{t(e)}$, $F(e, \gamma) = L_{h(e)} - A(e, \gamma)$. Since $A(e, \gamma)$ is the set of colours *allowed* at node $h(e)$ when γ is used at $t(e)$, so $F(e, \gamma)$ is the set of colours *forbidden* at $h(e)$ when γ is used at $t(e)$.

Lemma 5.3 *Let G be a series-parallel graph represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G. Then for every $e \in U$ and for every $\gamma \in L_{t(e)}$, the inequality*

$$\ell_{e,\gamma}(\mathbf{x}) \leq |V_e| - 1$$

is valid for $\mathcal{P}(G, \mathcal{L})$.

The proof of this lemma can be found in the appendix.

The definition of $\ell_{e,\gamma}$ can be extended from single colours to subsets of colours in the following way:

For every ur-edge $e \in U$, and every set $S \subseteq L_{t(e)}$, the linear function $\ell_{e,S} : \mathbb{R}^{V_{\mathcal{L}}} \to \mathbb{R}$ is defined as follows:

$$\ell_{e,S}(\mathbf{x}) = \sum_{\gamma \in S} x_{t(e),\gamma} + \sum_{w \in V_e^*} \sum_{\alpha \in L_w} x_{w,\alpha} + \sum_{\alpha \in F(e,S)} x_{h(e),\alpha},$$

where $V_e^* = V(G_e) - \{h(e), t(e)\}$, and $F(e, S) = \bigcap_{\alpha \in S} F(e, \alpha)$.

Lemma 5.4 *Let G be a series-parallel graph represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G. Then for every $e \in U$ and for every $S \subseteq L_{t(e)}$, the inequality*

$$\ell_{e,S}(\mathbf{x}) \leq |V_e| - 1$$

is valid for $\mathcal{P}(G, \mathcal{L})$.

Proof. Let $(G, \mathcal{L})$, ur-edge e and set S be as in the statement of the lemma. Now

$$\ell_{e,S}(\mathbf{x}) = \frac{1}{|S|} \sum_{\gamma \in S} \ell_{e,\gamma}(\mathbf{x}) + \frac{|S| - 1}{|S|} \sum_{\gamma \in S} x_{t(e),\gamma} - \epsilon(\mathbf{x}),$$

where $\epsilon(\mathbf{x})$ is a linear function with positive coefficients.

By Lemma 5.3, $\ell_{e,\gamma}(\mathbf{x}) \le |V_e| - 1$ is a valid inequality for $\mathcal{P}(G, \mathcal{L})$, for each $\gamma \in S$. Also, $\sum_{\gamma \in L_s} x_{t(e),\gamma} \le 1$ is valid for $\mathcal{P}(G, \mathcal{L})$. Using this as well as the fact that all vectors in $\mathcal{P}(G, \mathcal{L})$ are non-negative, we obtain that the inequality

$$\ell_{e,S}(\mathbf{x}) \le |V_e| - 1 + \frac{|S|}{|S| - 1}$$

is valid for $\mathcal{P}(G, \mathcal{L})$

Since all vertices of $\mathcal{P}(G, \mathcal{L})$, as well as the coefficients of $\ell_{e,S}$, are integers, we may round down the right-hand side of this inequality and it will remain valid. The result follows. □

The remaining part of the proof of Theorem 4.2 now follows easily from the preceeding lemmas.

Corollary 5.5 *Let G be a series-parallel graph with terminals s and t and root edge e_0, represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and let $\mathcal{L}$ be a list assignment for G. If $\bigcup_{\gamma \in L_s} A_G(e_0, \gamma) = \emptyset$, then no list colouring of $(G, \mathcal{L})$ exists.*

Proof. Let $(G, \mathcal{L})$ be as in the statement of the theorem. Note that $G_{e_0} = G$ and $\bigcup\limits_{\gamma \in L_s} F(e_0, \gamma) = L_t$. By Lemma 5.4, taking $e = e_0$ and $S = L_s$, it follows that

$$\sum_{v \in V(G)} \sum_{\gamma \in L_v} x_{v,\gamma} \le |V(G)| - 1$$

is a valid inequality for $\mathcal{P}(G, \mathcal{L})$.

However, it is clear that for any vector $\mathbf{x}$ representing a list colouring,

$$\sum_{v \in V} \sum_{\gamma \in L_v} x_{v,\gamma} = |V|.$$

Therefore, a list colouring does not exist. □

6 Future work

The results presented in Section 5 point to some interesting directions for further study. Firstly, the question arises whether the inequalities introduced in this section suffice to describe the polytope $\mathcal{P}(G, \mathcal{L})$ completely. If this turns out to be so, then the question immediately arises whether there are any other graphs for which the list colouring polytope is completely described by these inequalities. Secondly, it would be worthwhile to derive an algorithm which, if no complete list colouring can be found, will find a partial list colouring which optimizes the number of nodes coloured.

References

[1] M.M. Cropper, J.L. Goldwasser, A.J.W Hilton, D.G. Hoffman, and Johnson P.D. Extending the disjoint-representatives theorems of hall, holmos, and vaughan to list-multicolorings of graphs. *J. Graph Th.*, 33(4):199–219, 2000.

[2] D. de Werra. the combinatorics of timetabling. *Eur. J. Oper. Res.*, 96:504–513, 1997.

[3] D. de Werra, A.J. Hoffman, N.V.R. Mahadev, and U.N. Peled. Restrictions and preassignments in preemptive open shop scheduling. *Discr. Appl. Math.*, 68:169–188, 1996.

[4] J.E. Hopcroft and R.F. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2:135–158, 1973.

[5] Martin Juvan, Bojan Mohar, and Robin Thomas. List edge-colorings of series-parallel graphs. *Electr. J. Comb.*, 6, 1999.

[6] Tohru Kikuno, Noriyoshi Yoshida, and Yoshiaki Kakuda. A linear algorithm for the domination number of a series-parallel graph. *Discr. Appl. Math.*, 5:299–311, 1983.

[7] M. Eduardo Kortright, Arobinda Gupta, and Richard B. Borie. Past parallel algorithms for the balanced decomposition of series-parallel graphs. *Congr. Num.*, 134:193–218, 1998.

[8] Jan Kratochvíl and Zsolt Tuza. Algorithmic complexity of list colorings. *Discr. Appl. Math.*, 50:297–302, 1994.

[9] S.R. Kuman, A. Russell, and R. Sundaram. Approximating latin square extensions. *Algorithmica*, 24:128–138, 1999.

[10] O. Marcotte and P.D. Seymour. Extending an edge-coloring. *J. Graph. Th.*, 14(5):565–573, 1990.

[11] L. Narayanan. Channel assignment and graph multicoloring. In Ivan Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing.* Wiley, 2001.

[12] P.D. Seymour. Colouring series-parallel graphs. *Combinatorica*, 10:379–392, 1990.

[13] Zsolt Tuza. Graph coloring with local constraints — a survey. *Discuss. Math. Graph Th.*, 17:161–228, 1997.

[14] X. Zhou, H. Suzuki, and T. Nishizeki. A linear algorithm for edge-coloring series-parallel graphs. *J. Algorithms*, 20:174–201, 1996.

Appendix

In this appendix we give the proof of Lemma 5.3. We first restate the lemma.

Lemma 5.3. *Given are a series-parallel graph G represented by a set of ur-edges $U = (P, S, E)$ and a parse tree T, and a list assignment $\mathcal{L}$ for G. Then for every $e \in U$ and for every $\gamma \in L_{t(e)}$, the inequality*

$$\ell_{e,\gamma}(\mathbf{x}) \leq |V_e| - 1$$

is valid for $\mathcal{P}(G, \mathcal{L})$.

Proof. Let $(G, \mathcal{L})$ be as in the statement of the lemma. The proof will be by induction from the leaves to the root of the parse tree T. Let $e \in U$ be an ur-edge with $t(e) = u$ and $h(e) = v$, and let γ be a colour from L_u. If $e \in E$, then by Definition 4.1, $A(e, \gamma) = L_v - \{\gamma\}$, so $F(e, \gamma) = \{\gamma\}$. Since $e \in E$, G_e only has two nodes, so $|V_e| = 2$ and

$$\ell_{e,\gamma}(\mathbf{x}) = x_{u,\gamma} + x_{v,\gamma}.$$

The inequality $x_{u,\gamma} + x_{v,\gamma} \leq 1$ is obviously valid for $\mathcal{P}(G, \mathcal{L})$.

Suppose then that $e \in P$, with children e_1 and e_2, and assume that the statement of the lemma holds for e_1 and e_2. By Definition 4.1, $F(e, \gamma) = L_v - A(e, \gamma) = L_v - (A(e_1, \gamma) \cap A(e_2, \gamma)) = F(e_1, \gamma) \cup F(e_2, \gamma)$. Now

$$\ell_{e,\gamma}(\mathbf{x}) = \frac{1}{2}\ell_{e_1,\gamma}(\mathbf{x}) + \frac{1}{2}\ell_{e_2,\gamma}(\mathbf{x}) + \frac{1}{2}\sum_{w \in V_e^*}\sum_{\alpha \in L_w} x_{w,\alpha} + \frac{1}{2}\sum_{\alpha \in F(e_1,\gamma) \triangle F(e_2,\gamma)} x_{v,\alpha},$$

where $V_e^* = V(G_e) - \{u, v\}$.

Using the fact that $\ell_{e_1,\gamma}(\mathbf{x}) \leq |V_{e_1}| - 1$, $\ell_{e_2,\gamma}(\mathbf{x}) \leq |V_{e_2}| - 1$ and $\sum_{\alpha \in L_w} x_{w,\alpha} \leq 1$ are valid inequalities (the latter one holding for all w), we obtain that

$$\ell_{e,\gamma}(\mathbf{x}) \leq \frac{1}{2}(|V_{e_1}| - 1) + \frac{1}{2}(|V_{e_2}| - 1) + \frac{1}{2}|V_e^*| + \frac{1}{2}$$

is valid for $\mathcal{P}(G, \mathcal{L})$. Since all vertices of $\mathcal{P}(G, \mathcal{L})$, as well as the coefficients of $\ell_{e,\gamma}(\mathbf{x})$, are integers, we may round down the right-hand side of this inequality, and it will remain valid. Therefore the inequality

$$\ell_{e,\gamma}(\mathbf{x}) \leq \frac{1}{2}|V_{e_1}| + \frac{1}{2}|V_{e_2}| + \frac{1}{2}|V_e^*| - 1$$

is valid for $\mathcal{P}(G, \mathcal{L})$. Since G_{e_1} and G_{e_2} overlap only in u and v, we have that

$$|V_e| = |V_{e_1}| + |V_{e_2}| - 2.$$

Also, $|V_e^*| = |V_e| - 2$. This proves that the inequality $\ell_{e,\gamma}(\mathbf{x}) \le |V_e| - 1$ is valid when $e \in P$.

Suppose next that $e \in S$, with children e_1 and e_2, and assume the statement of the lemma holds for e_1 and e_2. Let $w = h(e_1) = t(e_2)$. It follows from Definition 4.1 that $F(e,\gamma) = L_v - A(e,\gamma) = L_v - \bigcup_{\alpha\in A(e_1,\gamma)} A(e_2,\alpha) = \bigcap_{\alpha\in A(e_1,\gamma)} F(e_2,\gamma)$. Hence

$$\ell_{e,\gamma}(\mathbf{x}) = \ell_{e_1,\gamma}(\mathbf{x}) + \frac{1}{n_A}\sum_{\alpha\in A(e_1,\gamma)} \ell_{e_2,\alpha}(\mathbf{x}) + \frac{n_A - 1}{n_A}\sum_{\alpha\in A(e_1,\gamma)} x_{w,\alpha} - \epsilon(\mathbf{x}),$$

where $\epsilon(\mathbf{x})$ is a linear function with positive coefficients, and $n_A = |A(e_1,\gamma)|$.

Using the fact that $\ell_{e_1,\gamma}(\mathbf{x}) \le |V_{e_1}| - 1$ and $\ell_{e_2,\alpha}(\mathbf{x}) \le |V_{e_2}| - 1$ are valid inequalities (the second one holding for all $\alpha \in L_w$), and that $\displaystyle\sum_{\alpha\in L_w} x_{w,\alpha} \le 1$, as well as the fact that all vectors in $\mathcal{P}(G,\mathcal{L})$ are non-negative, we obtain that

$$\ell_{e,\gamma}(\mathbf{x}) \le (|V_{e_1}| - 1) + (|V_{e_2}| - 1) + \frac{n_A - 1}{n_A}$$

is valid for $\mathcal{P}(G,\mathcal{L})$. As in the previous case, we may round down the right hand side. Hence

$$\ell_{e,\gamma}(\mathbf{x}) \le |V_{e_1}| + |V_{e_2}| - 2$$

is a valid inequality for $\mathcal{P}(G,\mathcal{L})$. Since G_{e_1} and G_{e_2} overlap only in w,

$$|V_e| = |V_{e_1}| + |V_{e_2}| - 1.$$

Hence the result for $e \in S$ follows. □