Extended Skew Partition Problem*

Simone Dantas[†] Celina M. H. de Figueiredo[‡]
Sylvain Gravier[§] Sulamita Klein[‡]

Abstract

A skew partition as defined by Chvátal is a partition of the vertex set of a graph into four nonempty parts A_1, A_2, B_1, B_2 such that there are all possible edges between A_1 and A_2, and no edges between B_1 and B_2. We introduce the concept of (n_1, n_2)-extended skew partition which includes all partitioning problems into $n_1 + n_2$ nonempty parts $A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}$ such that there are all possible edges between the A_i parts, no edges between the B_j parts, $i \in \{1, \ldots, n_1\}, j \in \{1, \ldots, n_2\}$, which generalizes the skew partition. We present a polynomial-time algorithm for testing whether a graph admits an (n_1, n_2)-extended skew partition. As a tool to complete this task we also develop a generalized 2-SAT algorithm, which by itself may have application to other partition problems.

Keywords : Algorithms and data structures, Computational and structural complexity, Skew partition, 2-SAT

1 Introduction

A *skew partition* is a partition of the vertex set of a graph into four nonempty parts A_1,A_2,B_1,B_2 such that there are all possible edges between A_1 and A_2, and no edges between B_1 and B_2. A skew partition was defined by

*This research was partially supported by CNPq, MCT/FINEP PRONEX Project 107/97, CAPES (Brazil)/COFECUB (France), project number 213/97, FAPERJ.

[†]COPPE, Universidade Federal do Rio de Janeiro, Brazil.

[‡]Instituto de Matemática and COPPE, Universidade Federal do Rio de Janeiro, Brazil.

[§]CNRS, Laboratoire Leibniz, France.

Chvátal [3] in the context of perfect graphs and it has a key role in the recent celebrated proof of the Strong Perfect Graph Conjecture by Seymour et al. [13]. De Figueiredo et al. [5] presented a polynomial-time algorithm for testing whether a graph admits a skew partition. In this paper we introduce the concept of extended skew partition, which generalizes the skew partition.

An (n_1, n_2)-*extended skew partition* is a partition of the vertex set of a graph into $n_1 + n_2$ nonempty parts $A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}$ such that there are all possible edges between the A_i parts, no edges between the B_j parts, $i \in \{1, \ldots, n_1\}, j \in \{1, \ldots, n_2\}$.

An extended skew partition can be viewed also as a special M-partition problem. The M-partition problem was defined by Feder et al. [8] as a partition of the vertex set of a graph into k parts $X_1, X_2, \ldots, X_k$ with a fixed "pattern" of requirements as to which X_i are independent or complete and which pairs X_i, X_j are completely nonadjacent or completely adjacent. These requirements may be conveniently encoded by a symmetric $k \times k$ matrix M in which the diagonal entry $M_{i,i}$ is 0 if X_i is required to be independent, 2 if X_i is required to be a clique, and 1 otherwise (no restriction). Similarly, the off-diagonal entry $M_{i,j}$ is 0, 1, or 2, if X_i and X_j are required to be completely nonadjacent, have arbitrary connections, or are required to be completely adjacent, respectively.

In our case, an (n_1, n_2)-*extended skew partition* is an M-partition with the additional constraint that all parts must be nonempty, and M is the following $(n_1 + n_2) \times (n_1 + n_2)$ matrix: $M_{i,j} = 2$, if $1 \leq i \neq j \leq n_1$; $M_{i,j} = 0$, if $i \neq j > n_1$; and $M_{i,j} = 1$ otherwise.

The most convenient way to express these additional constraints is to allow specifying (as part of the input) for each vertex a "list" of parts in which the vertex is allowed to be. Specifically, the *list-M-partition problem* asks for an M-partition of the input graph in which each vertex is placed in a part which is in its list. Both the basic M-partition problem ("Does the input graph admit an M-partition?") and the problem of existence of an M-partition with all parts nonempty admit polynomial-time reductions to the list-M-partition problem, as do all of the above problems with the "additional" constraints. List partitions generalize list-colorings, which have proved very fruitful in the study of graph colorings [1, 9]. They also generalize list-homomorphisms, which were studied earlier [6, 7]. Feder et al. [8] were the first to introduce and investigate the list version of these problems. List partition problems have attracted much attention lately [8, 10, 11, 12, 13].

Our algorithm follows closely the algorithm for finding skew partition given in [5]. In order to describe a more general algorithm for finding an extended skew partition we generalize the procedures described in [5]. A key element of our algorithm is a simple but non obvious way of developping of what we call *generalized 2-SAT* procedure. We believe that this procedure may be of broader use to other partition problems.

2 Overview

The goal of this paper is to present a polynomial-time algorithm for the following decision problem:

(n_1, n_2)-*Extended Skew Partition Problem*
Input: a graph $G = (V, E)$.
Question: Does G admit as extended skew partition $A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}$?

For each vertex v, we associate a subset L_v of $\{A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}\}$ which we call *list*. We actually consider extended list skew partition (ELSP) problems, stated as decision problems as follows:

(n_1, n_2)-*Extended List Skew Partition Problem*
Input: a graph $G = (V, E)$ and, for each vertex $v \in V$, a list $L_v \subseteq \{A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}\}$.
Question: Is there an extended skew partition $A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}$ of G such that each v is contained in some element of the corresponding L_v?

Throughout the algorithm, we have a partition of V into at most $2^{n_1+n_2} - 1$ sets S_L, indexed by the nonempty subsets L of $\{A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}\}$, such that Property 1 below is always satisfied.

Property 1 *If the algorithm returns an extended skew partition, then if v is in S_L, then the returned extended skew partition set containing v is in L.*

The relevant inputs for ELSP have S_{A_i} and S_{B_j} nonempty, $i \in \{1, \ldots, n_1\}$, $j \in \{1, \ldots, n_2\}$. We refer to the unitary lists as *trivial* lists. Initially, we set $S_L = \{v : L_v = L\}$, for each $L \subseteq \{A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}\}$. We denote the list $\mathcal{A} = \{A_1, A_2, \ldots, A_{n_1}\}$, the list $\mathcal{B} = \{B_1, B_2, \ldots, B_{n_2}\}$, and the list $\mathcal{AB} = \{A_1, A_2, \ldots, A_{n_1}, B_1, B_2, \ldots, B_{n_2}\}$. Thus initially the vertex set is partitioned into $n_1 + n_2$ sets corresponding to the trivial lists, plus a set corresponding to list $\mathcal{AB}$.

We also restrict our attention to ELSP instances that satisfy the following property:

Property 2 *If $v \in S_L$, for some L with $A_i \in L$, then v is adjacent to every vertex of S_{A_k}, for all $A_k \in \mathcal{A} \setminus A_i$. If $v \in S_L$, for some L with $B_j \in L$, then v is nonadjacent to every vertex of S_{B_l}, for all $B_l \in \mathcal{B} \setminus B_j$.*

Both Properties 1 and 2 hold throughout the algorithm. The algorithm proceeds by reducing the size of nontrivial lists. An extended skew partition returned by the algorithm is a set of n trivial lists. The following remark characterizes the set of *possible lists* throughout the algorithm.

Remark 1 *By Property 1, every list L_v satisfies*

- *If $L_v \cap \mathcal{A} \neq \emptyset$, then $L_v \cap \mathcal{A} = \{A_k\}$ or $L_v \cap \mathcal{A} = \mathcal{A}$, and*
- *If $L_v \cap \mathcal{B} \neq \emptyset$, then $L_v \cap \mathcal{B} = \{B_k\}$ or $L_v \cap \mathcal{B} = \mathcal{B}$.*

For, if $A_i \notin L_v$, then there exists $A_k \in \mathcal{A} \setminus A_i$ such that v is non-adjacent to $w \in A_k$, which implies that $A_j \notin L_v$, for all $j \neq k$, i.e., if $L_v \cap \mathcal{A} \neq \emptyset$, then $L_v \cap \mathcal{A} = \{A_k\}$.

So, the set of possible lists is the following: $n_1 + n_2$ trivial lists A_1, A_2, …, A_{n_1}, B_1, B_2, …, B_{n_2}; $n_1 n_2$ lists of type $A_i B_j$; the list $\mathcal{A}$; the list $\mathcal{B}$; n_1 lists of type $A_i\mathcal{B}$; n_2 lists of type $B_j\mathcal{A}$; the list $\mathcal{AB}$.

Remark 2 *Since S_{A_l} must be contained in A_l, we know that if v is to be in A_j for some solution to the problem, then v must be adjacent to all vertices of S_{A_l}. Thus if some $v \in S_{A_j}$ is not adjacent to a vertex of S_{A_l}, then there is no solution to the problem and we need not continue. If there is some L with A_j properly contained in L and a vertex v in S_L which is not adjacent to a vertex of S_{A_l}, then we know that in any solution to the problem v must be contained in some element of $L \setminus A_j$. So we can reduce to a new problem where we replace S_L by $S_L \setminus v$, we replace $S_{L\setminus A_j}$ by $S_{L\setminus A_j} + v$ and all other S_L are as before. Such a reduction reduces $\sum_L |S_L||L|$ by 1. Since this sum is at most $(n_1 + n_2)n$, where n denotes the number of vertices in the input graph G, after $O(n)$ similar reductions we must obtain an ELSP problem satisfying Property 2 (or halt because the original problem has no solution).*

Along the algorithm, we often create new ELSP instances and whenever we do so, we always perform the procedure described in Remark 2 to reduce to an ELSP problem satisfying Property 2. For an instance I of ELSP we have $\{S_L(I) : L \subseteq \{A_1, \ldots, A_{n_1}, B_1, \ldots, B_{n_2}\}$, but we drop the (I) when it is not needed for clarity.

We consider a number of restricted versions of the ELSP problems:

- $\mathcal{AB}$-TRIV-ELSP: an ELSP problem satisfying Property 2 such that $S_{\mathcal{AB}} = \emptyset$;
- GEN-MAX-2-ELSP: an ELSP problem satisfying Property 2 such that if $|L| > 2$ and $L \neq \mathcal{A}$ and $L \neq \mathcal{B}$, then $S_L = \emptyset$;
- A_qB_p-TRSV-ELSP: an ELSP problem satisfying Property 2 such that A_qB_p is a *list transversal* for indices $q \in \{1, \ldots, n_1\}$, $p \in \{1, \ldots, n_2\}$, i.e., a list L_T that intersects all lists of size at least 2, and such that L_T is trivial or nontrivial having no restrictions between the parts contained in L_T.

Remark 3 *Recall that the relevant inputs for ELSP have $S_{A_1}, \ldots, S_{A_{n_1}}$, $S_{B_1}, \ldots, S_{B_{n_2}}$ nonempty. It is easy to obtain a solution to an instance of A_qB_p-TRSV-ELSP as follows:*

$$A_i = S_{A_i},\ i \neq q; A_q \ = \bigcup_{A_q \in L, B_p \notin L} S_L; B_p \ = \bigcup_{B_p \in L} S_L; B_j = S_{B_j},\ j \neq p.$$

By Property 2 this is indeed an extended skew partition.

Our algorithm for solving ELSP requires four subalgorithms which replace an instance of ELSP by a polynomial number of instances of more restricted versions of ELSP. Algorithms 1, 2, 3 and 4 replace an instance of ELSP by a polynomial number of instances of GEN-MAX-2-ELSP or $A_qB_q - TRSV - ELSP$ which are solved by Algorithm 5 and 6, respectively.

Algorithm 1 *Takes an instance of ELSP and returns in polynomial time a list $\mathcal{L}$ of a polynomial number of instances of $\mathcal{AB}$-TRIV-ELSP such that*

(i) *a solution to any problem in $\mathcal{L}$ is a solution of the original problem, and*

(ii) *if none of the problems in $\mathcal{L}$ have a solution, then the original problem has no solution.*

Algorithm 2 *Takes an instance of $\mathcal{AB}$-TRIV-ELSP and returns in polynomial time a list $\mathcal{L}$ of a polynomial number of instances of $\mathcal{AB}$-TRIV-ELSP such that*

(i) and (ii) *of Algorithm 1 hold, and*

(iii) *for each problem in $\mathcal{L}$, there exists $p \in \{1, \ldots, n_2\}$, such that $S_{B_j\mathcal{A}} = \emptyset$, for all $j \neq p$.*

Algorithm 3 *Takes an instance of $\mathcal{AB}$-TRIV–ELSP and returns in polynomial time a list $\mathcal{L}$ of a polynomial number of instances of $\mathcal{AB}$-TRIV-ELSP such that*

(i) and (ii) *of Algorithm 1 hold, and*

(iii) *for each problem in $\mathcal{L}$, there exists $q \in \{1, \ldots, n_1\}$, such that $S_{A_i\mathcal{B}} = \emptyset$, for all $i \neq q$.*

Algorithm 4 *Takes an instance of $\mathcal{AB}$-TRIV-ELSP such that*

(a) *there exists $p \in \{1, \ldots, n_2\}$, such that $S_{B_j\mathcal{A}} = \emptyset$, for all $j \neq p$, and*

(b) *there exists $q \in \{1, \ldots, n_1\}$, such that $S_{A_i\mathcal{B}} = \emptyset$, for all $i \neq q$.*

and returns in polynomial time a list $\mathcal{L}$ of a polynomial number of problems each of which is an instance of one of GEN-MAX-2-ELSP or A_qB_p-TRSV-ELSP such that (i) and (ii) of Algorithm 1 hold.

Algorithm 5 *(***generalized 2-SAT***) Takes an instance of GEN-MAX-2-ELSP and returns either*

(i) *a solution to this instance of GEN-MAX-2-ELSP, or*

(ii) *the information that this problem instance has no solution.*

Algorithm 6 *Takes an instance of A_qB_p-TRSV-ELSP returns a solution using the partition discussed in Remark 3.*

To solve an instance of ELSP, we first apply Algorithm 1 to obtain a list $\mathcal{L}_1$ of instances of $\mathcal{AB}$-TRIV-ELSP. For each problem instance I on $\mathcal{L}_1$, we apply Algorithm 2 and let $\mathcal{L}_I$ be the output list of problem I. We let $\mathcal{L}_2$ be the concatenation of the lists $\{\mathcal{L}_I : I \in \mathcal{L}_1\}$. For each I in $\mathcal{L}_2$, we apply Algorithm 3. Let $\mathcal{L}_3$ be the concatenation of the lists $\{\mathcal{L}_I : I \in \mathcal{L}_2\}$. For each problem instance I on $\mathcal{L}_3$, we apply Algorithm 4. Let $\mathcal{L}_4$ be the concatenation of the lists $\{\mathcal{L}_I : I \in \mathcal{L}_3\}$. Each element of $\mathcal{L}_4$ can be solved in polynomial time using either Algorithm 5 or Algorithm 6. If any of these problems has a solution S, then by the specifications of the algorithms, S is a solution to the original problem. Otherwise, by the specifications of the algorithms, there is no solution to the original problem. Clearly, the whole algorithm runs in polynomial time.

3 Some Recursive Procedures

Algorithm 1 recursively applies Procedure 1, which runs in polynomial time.

Procedure 1 Input*: An instance I of ELSP.*
Output*: $n_1 + n_2$ instances $I_1, \ldots, I_{n_1+n_2}$ of ELSP such that, for $1 \le t \le n_1 + n_2$, we have $|S_{\mathcal{AB}}(I_t)| \le \frac{9}{10}|S_{\mathcal{AB}}(I)|$.*

It is easy to prove inductively that applying Procedure 1 recursively yields a polynomial-time implementation of Algorithm 1 which when applied to an input graph with n vertices creates as output a list $\mathcal{L}$ of instances of ELSP such that $|\mathcal{L}| \le (n_1 + n_2)^{\log_{\frac{10}{9}} n} = n^{\log_{\frac{10}{9}}(n_1+n_2)}$.

Algorithm 2 recursively applies Procedure 2, which runs in polynomial time.

Procedure 2 Input*: An instance I of $\mathcal{AB}$-TRIV-ELSP.*
Output*: $n_1 + n_2$ instances $I_1, \ldots, I_{n_1+n_2}$ of $\mathcal{AB}$-TRIV-ELSP such that, for all $1 \le t \le n_1 + n_2$, we have $|S_{B_1\mathcal{A}}(I_t)||S_{B_2\mathcal{A}}(I_t)| \le \frac{9}{10}|S_{B_1\mathcal{A}}(I)||S_{B_2\mathcal{A}}(I)|$.*

Algorithm 2 recursively applies $O(n_2^2)$ procedures whose definitions are similar to Procedure 2 and consider all possible values of pairs $B_j\mathcal{A}$, $B_l\mathcal{A}$, with $j \ne l$, $j, l \in \{1, \ldots, n_2\}$. It is easy to see that recursively applying Procedure 2 or one of its variants, as appropriate, yields a polynomial- time implementation of Algorithm 2 which when applied to an input graph with n vertices creates an output list $\mathcal{L}$ with $O(n^{2\log_{\frac{10}{9}}(n_1+n_2)})$.

Algorithm 3 recursively applies Procedure 3, which runs in polynomial time.

Procedure 3 Input*: An instance I of $\mathcal{AB}$-TRIV-ELSP.*
Output*: $n_1 + n_2$ instances $I_1, \ldots, I_{n_1+n_2}$ of $\mathcal{AB}$-TRIV-ELSP such that, for all $1 \le t \le n_1 + n_2$, we have $|S_{A_1\mathcal{B}}(I_t)||S_{A_2\mathcal{B}}(I_t)| \le \frac{9}{10}|S_{A_1\mathcal{B}}(I)||S_{A_2\mathcal{B}}(I)|$.*

Algorithm 3 recursively applies $O(n_2^2)$ procedures whose definitions are similar to Procedure 3 and consider all possible values of pairs $A_j\mathcal{B}$, $A_l\mathcal{B}$, with $j \ne l$, $j, l \in \{1, \ldots, n_1\}$. It is easy to see that recursively applying Procedure 3 or one of its variants, as appropriate, yields a polynomial- time implementation of Algorithm 3 which when applied to an input graph with n vertices creates an output list $\mathcal{L}$ with $O(n^{2\log_{\frac{10}{9}}(n_1+n_2)})$.

Algorithm 4 recursively applies Procedure 4, which runs in polynomial time.

Procedure 4 Input*: An instance I of $\mathcal{AB}$-TRIV-ELSP such that*

- *there exists $p \in \{1, \dots, n_2\}$, such that $S_{B_j\mathcal{A}} = \emptyset$, for all $j \neq p$, and*
- *there exists $q \in \{1, \dots, n_1\}$, such that $S_{A_i\mathcal{B}} = \emptyset$, for all $i \neq q$.*

Output*: n_1+n_2 instances $I_1, \dots, I_{n_1+n_2}$ of $\mathcal{AB}$-TRIV-ELSP such that, there exists $j' \neq p$, for all $1 \leq t \leq n_1 + n_2$, satisfying $|S_{B_p\mathcal{A}}(I_t)||S_{A_iB_{j'}}(I_t)| \leq \frac{9}{10}|S_{B_p\mathcal{A}}(I)||S_{A_iB_{j'}}(I)|$.*

Procedure 4 has $n_1 \times (n_2 - 1)$ variants corresponding to the lists $B_p\mathcal{A}$ and A_iB_j, with $1 \leq i \leq n_1$, and $j \neq p$, $1 \leq j \leq n_2$, and $(n_1-1)\times n_2$ variants corresponding to the lists $A_q\mathcal{B}$ and A_iB_j, with $1 \leq j \leq n_2$, and $i \neq q$, $1 \leq i \leq n_1$. Algorithm 4 recursively applies these $O(n_1 \times (n_2-1) + (n_1-1)\times n_2)$ procedures.

It is easy to see that recursively applying Procedure 4 or one of its variants, as appropriate, yields a polynomial- time implementation of Algorithm 4 which when applied to an input graph with n vertices creates an output list $\mathcal{L}$ with $O(n^{2\log_{\frac{10}{9}}(n_1+n_2)})$.

The four procedures above are based on those methods applied by de Figueiredo, Klein, Kohayakawa and Reed [5] with appropriate modifications.

4 The Details of the recursive procedures

Procedure 1

Let $n = |S_{\mathcal{AB}}(I)|$. For an extended skew partition $\{A_1, \dots, A_{n_1}, B_1, \dots, B_{n_2}\}$, let $A'_i = A_i \cap S_{\mathcal{AB}}(I)$ and $B'_j = B_j \cap S_{\mathcal{AB}}(I)$ for all $i = 1, \dots, n_1$ and $j = 1, \dots, n_2$.

Case 1: There exists a vertex v in $S_{\mathcal{AB}}$ such that $\frac{n}{10} \leq |S_{\mathcal{AB}} \cap N(v)| \leq \frac{9n}{10}$.

Branch according to whether $v \in A_1, \dots, v \in A_{n_1}, v \in B_1, \dots,$ or $v \in B_{n_2}$ with instances $I_{A_1}, \dots, I_{A_{n_1}}, I_{B_1}, \dots, I_{B_{n_2}}$, respectively. For all $i = 1, \dots, n_1$, define I_{A_i} by initially setting $S_{A_i}(I_{A_i}) = v + S_{A_i}(I)$ and reducing so that Property 2 holds. Define $I_{B_1}, \dots, I_{B_{n_2}}$ similarly. Note that by Property 2, if $v \in B_i$, then $B_j \cap N(v) = \emptyset$ for all $j \neq i$. So, $S_{\mathcal{AB}}(I_{B_i}) \subset S_{\mathcal{AB}}(I) \setminus N(v)$. Because there are at least $\frac{n}{10}$ vertices in $S_{\mathcal{AB}} \cap N(v)$, this means $|S_{\mathcal{AB}}(I_{B_i})| \leq \frac{9n}{10}$ for all $i = 1, \dots, n_2$.

Similarly, by Property 2, $S_{\mathcal{AB}}(I_{A_i}) \subset S_{\mathcal{AB}}(I) \cap N(v)$, so $|S_{\mathcal{AB}}(I_{A_i})| \leq \frac{9n}{10}$ for all $i = 1, \ldots, n_1$. $\blacksquare$

Let $W = \{v \in S_{\mathcal{AB}} : |S_{\mathcal{AB}} \cap N(v)| > \frac{9n}{10}\}$ and $X = \{v \in S_{\mathcal{AB}} : |S_{\mathcal{AB}} \cap N(v)| < \frac{n}{10}\}$.

Case 2: $|W| \geq \frac{n}{10}$ and $|X| \geq \frac{n}{10}$.

Branch according to :

(i) $I_1 : |A'_1| \geq \frac{n}{10}$, or

(ii) $I_2 : |\cup_{i \neq 1} A'_i| \geq \frac{n}{10}$, or

(iii) $I_3 : |B'_1| \geq \frac{n}{10}$, or

(iv) $I_4 : |\cup_{i \neq 1} B'_i| \geq \frac{n}{10}$.

Each of these choices forces either all the vertices in W or all the vertices in X to have smaller label sets, as follows.
If $|A'_1| \geq \frac{n}{10}$, then every vertex in A'_j $\forall j \neq 1$ has $\frac{n}{10}$ neighbours in $S_{\mathcal{AB}}(I)$, so $A'_j \cap X = \emptyset$ for all $j \neq 1$.
If $|\cup_{i \neq 1} A'_i| \geq \frac{n}{10}$ then every vertex in A'_1 has $\frac{n}{10}$ neighbours in $S_{\mathcal{AB}}(I)$, so $A'_1 \cap X = \emptyset$.
Thus, for $j = 1, 2$ we have $S_{\mathcal{AB}}(I_j) = S_{\mathcal{AB}}(I) \setminus X$, and $|S_{\mathcal{AB}}(I_j)| \leq \frac{9n}{10}$.
If $|B'_1| \geq \frac{n}{10}$ then every vertex in B_j $\forall j \neq 1$ has at least $\frac{n}{10}$ non-neighbours in $S_{\mathcal{AB}}(I)$. Hence $W \cap B_j = \emptyset$ for all $j \neq 1$.
If $|\cup_{i \neq 1} B'_i| \geq \frac{n}{10}$ then every vertex in B_1 has at least $\frac{n}{10}$ non-neighbours in $S_{\mathcal{AB}}(I)$. Hence $W \cap B_1 = \emptyset$.
Hence, for $j = 3, 4$ we have $S_{\mathcal{AB}}(I_j) = S_{\mathcal{AB}}(I) \setminus W$, and so $|S_{\mathcal{AB}}(I_j)| \leq \frac{9n}{10}$. $\blacksquare$

Case 3: $|W| > \frac{9n}{10}$.

In [5], the authors proved that there exists 3 subsets O, T and NT of W satisfying :

- There are all edges between O and T;
- For every w in NT, there exists v in O such that w is not adjacent to v;
- The complement of O is connected.

And such that :

(i) $|O| + |NT| \geq \frac{n}{10}$ and $|T| \geq \frac{n}{10}$; or

(ii) $NT = \emptyset$.

In case (i), we consider the following intances :

(b_1) I_1 : $B_1 \cap O \neq \emptyset$, or ...

(b_{n_2}) I_{n_2} : $B_{n_2} \cap O \neq \emptyset$, $(\cup_{i \neq n_2} B_i) \cap O = \emptyset$, or

(a_1) I_{n_2+1} : $O \subseteq A_1$, or ...

(a_{n_1}) $I_{n_2+n_1}$: $O \subseteq A_{n_1}$.

Recall that the complement of O is connected, which implies that if $O \cap \mathcal{B} = \emptyset$, then $O \subseteq A_i$ for some $i \in \{1, \ldots, n_1\}$.
If $O \subseteq A_i$ for some i, then $NT \cap A_j = \emptyset$ for all $j \neq i$ since for every $w \in NT$ there is a vertex $v \in O$ such that $vw \notin E$.
Thus, $(O \cup NT) \cap S_{\mathcal{AB}}(I_{n_2+i}) = \emptyset$. Hence $|S_{\mathcal{AB}}(I_{n_2+i})| \leq \frac{9n}{10}$ for all $i = 1, \ldots, n_1$.
If $B_i \cap O \neq \emptyset$ for some i, then $(\cup_{j \neq i} B_j) \cap T = \emptyset$.
Thus, $T \cap S_{\mathcal{AB}}(I_i) = \emptyset$, which implies $|S_{\mathcal{AB}}(I_i)| \leq \frac{9n}{10}$ for all $i = 1, \ldots, n_2$.
Hence if (i) holds then we have found $n_1 + n_2$ desired output instances of ELSP. Otherwise O, T and NT satisfy (ii). In this case, the authors in [5] proved that there exist two subsets Y and Z of W such that :

- There are all edges between Y and Z;
- $|Y| \geq \frac{n}{10}$;
- $|Z| \geq \frac{n}{10}$.

Now, we consider the instances :

(b_1) I_1: $B_1 \cap Z \neq \emptyset$, ...

(b_{n_1}) I_{n_1}: $B_{n_1} \cap Z \neq \emptyset$,

(b_{n_1+1}) I_{n_1+1}: $(\cup B_i) \cap Z = \emptyset$.

Since there are all edges between Y and Z, for every $j \leq n_1$, $S_{\mathcal{AB}}(I_j) \subseteq S_{\mathcal{AB}}(I) \backslash Y$ and $S_{\mathcal{AB}}(I_{n_1+1}) \subseteq S_{\mathcal{AB}}(I) \backslash Z$. Thus for all j, we have $|S_{\mathcal{AB}}(I_j)| \leq \frac{9n}{10}$. ∎

Note that the case $|X| > \frac{9n}{10}$ is symmetric to Case 3 (consider $\overline{G}$) and is omitted. ∎

Procedure 2

Let $S_1 = S_{B_j\mathcal{A}}(I)$ and and $S_2 = S_{B_l\mathcal{A}}(I)$. Let $s_1 = |S_1|$, $s_2 = |S_2|$. Given $v \in S_1 \cup S_2$, let $d_i(v) = |N(v) \cap S_i|$, $i = 1, 2$.

Case 1: There exists a vertex $v \in S_1$ with $s_2/10 \leq d_2(v) \leq 9s_2/10$.

This case is analogous to Case 1 of Procedure 1: we create $n_1 + 1$ new instances $I_{B_j}, I_{A_1}, \ldots, A_{n_1}$ according to whether $v \in B_j$, or $v \in A_1$ or ... or $v \in A_{n_1}$.
Thus $S_{\mathcal{A}B_l}(I_{A_k}) \subseteq S_2 \cap N(v)$ and hence $|S_{\mathcal{A}B_l}(I_{A_k})| \leq \frac{9n_2}{10}$ for all k. And $S_{\mathcal{A}B_l}(I_{B_j}) \subseteq S_{B_l} \setminus (S_2 \cap N(v))$ and hence $|S_{\mathcal{A}B_l}(I_{B_j})| \leq \frac{9n_2}{10}$. ■

The case "there exists a vertex $v \in S_2$ with $\frac{s_1}{10} \leq d_1(v) \leq \frac{9s_1}{10}$", symmetric to Case 1 is ommited.

Case 2: Every vertex v in S_1 satisfies $d_2(v) < \frac{s_2}{10}$ or $d_2(v) > \frac{9s_2}{10}$. Every vertex v in S_2 satisfies $d_1(v) < \frac{s_1}{10}$ or $d_1(v) > \frac{9s_1}{10}$.

Define four auxiliary sets, as follows:

$$\begin{aligned}
X_1 &= \{v \in S_1 : d_2(v) < \frac{s_2}{10}\}, \\
X_2 &= \{v \in S_2 : d_1(v) < \frac{s_1}{10}\}, \\
W_1 &= \{v \in S_1 : d_2(v) > \frac{9s_2}{10}\}, \\
W_2 &= \{v \in S_2 : d_1(v) > \frac{9s_1}{10}\}.
\end{aligned}$$

Note that Case 2 means that $S_1 = X_1 \cup W_1$ and $S_2 = X_2 \cup W_2$. We handle Case 2 according to the following possibilities.

Case 2.1: $|X_1|, |W_1| \geq \frac{s_1}{10}$.

This case is analogous to Case 2 of Procedure 1. We create $n_1 + 1$ new instances of ELSP according to the size of skew partition sets, as follows:

(a_1) I_1: $|A_1 \cap S_2| \geq \frac{s_2}{10}$, or ...

(a_{n_1}) I_{n_1}: $|A_{n_1} \cap S_2| \geq \frac{s_2}{10}$, or

(b_1) I_{n_1+1}: $|B_l \cap S_2| \geq \frac{s_2}{10}$.

If $|A_i \cap S_2| \geq \frac{s_2}{10}$ for some i, then as every vertex in A_k ($k \neq i$) is adjacent to every vertex of A_i, $A_k \cap X_1 = \emptyset$ for all $k \neq i$. Thus $S_{\mathcal{A}B_j}(I_i) \leq \frac{9s_1}{10}$. Similarly, $S_{\mathcal{A}B_j}(I_{n_1+1}) \cap W_1 = \emptyset$. So, for all n_1+1 output instances, we have $|S_{\mathcal{A}B_j}| \leq \frac{9s_1}{10}$, as required. $\blacksquare$

The case "otherwise $|X_2|, |W_2| \geq \frac{s_2}{10}$", symmetric to Case 2.1, is omitted.

Case 2.2: $|X_1| > \frac{9s_1}{10}$.

In [5], the authors proved that there exists three sets O, M, and NM such that:

- $O \subseteq X_1$, $S_2 = M \cup NM$;
- There are no edges between O and M;
- For every $u \in NM$, there is a $w \in O$ with $wu \in E$;

And for which either :

(i) $|M| \leq \frac{s_2}{2}$; or

(ii) $|O| \geq \frac{3s_1}{10}$.

If condition (ii) holds, i.e., $|O| \geq \frac{3s_1}{10}$ and $|M| > \frac{s_2}{2}$, then define two new instances of ELSP as follows:

(a) $I_1 : O \cap A_1 = \emptyset$,

(b) $I_2 : O \cap A_1 \neq \emptyset$.

Clearly, $S_{\mathcal{A}B_j}(I_1) \subseteq S_1 \setminus O$, so $|S_{\mathcal{A}B_j}(I_1)| \leq \frac{9s_1}{10}$. Further, if $O \cap A_1 \neq \emptyset$ then $M \cap B_l = \emptyset$, so $|S_{\mathcal{A}B_l}(I_2)| \leq \frac{9s_2}{10}$.
If condition (i) holds, i.e., $|M| \leq \frac{s_2}{2}$, then the authors in [5] proved that we may assume that $\frac{4s_2}{10} < |M|$ and $|NM| \geq \frac{s_2}{2}$. We define $n_1 + 1$ new ELSP instances:

(a_1) $I_1 : O \cap A_1 \neq \emptyset$, or ...

(a_{n_1}) $I_{n_1} : O \cap A_{n_1} \neq \emptyset$, or

(a_{n_1+1}) $I_{n_1+1} : O \subseteq B_j$.

If $O \cap A_i \neq \emptyset$ for some i, then $A_k \subseteq (S_2 \setminus M)$ for all $k \neq i$, so $|S_{\mathcal{A}B_l}(I_i)| \leq \frac{9s_2}{10}$. Finally, if $O \subseteq B_j$ then $NM \cap B_l = \emptyset$ so $|S_{\mathcal{A}B_l}(I_{n_1+1})| \leq \frac{s_2}{2}$. $\blacksquare$

Note that the case "otherwise $|X_2| > 9s_2/10$", symmetric to Case 2.2, is omitted.

Case 2.3: $|W_1| > \frac{9s_1}{10}$, and $|W_2| > \frac{9s_2}{10}$.

Let $W = W_1 \cup W_2$. In [5], the authors proved that there is a partition of W into three sets O, T and NT such that:

- The complement of O is connected;
- There are all edges between O and T;
- For every $w \in NT$, there exists $u \in O$ such that $uw \notin E$.

and with the property that :

(i) $|O \cap S_1| + |NT \cap S_1| \geq \frac{s_1}{10}$, or $|O \cap S_2| + |NT \cap S_2| \geq \frac{s_2}{10}$; or

(ii) $NT = \emptyset$.

If condition (i) holds, say w.l.o.g. that $|O \cap S_1| + |NT \cap S_1| \geq \frac{s_1}{10}$. In [5], the authors shown that $|O \cap S_1| + |NT \cap S_1| < \frac{s_1}{5}$, $|O \cap S_2| + |NT \cap S_2| < \frac{s_2}{5}$, and on the other hand, $|T \cap S_1| \geq \frac{s_1}{10}$, and $|T \cap S_2| \geq \frac{s_2}{10}$.
Recall that the complement of O is connected, which implies that if $O \cap (B_j \cup B_l) = \emptyset$, then there exists an $i \in \{1, \ldots, n_1\}$ such that $O \subseteq A_i$. So we consider the following $n_1 + 2$ ELSP instances:

(a_1) I_1: $B_j \cap O \neq \emptyset$,

(a_2) I_2: $B_l \cap O \neq \emptyset$,

(a_3) I_3: $O \subseteq A_1$, ...

(a_{n_1+2}) I_{n_1+2}: $O \subseteq A_{n_1}$.

If $B_j \cap O \neq \emptyset$, then $(T \cap S_2) \cap B_l = \emptyset$, so $|S_{\mathcal{A}B_l}(I_1)| \leq \frac{9s_2}{10}$. If $B_l \cap O \neq \emptyset$, then $(T \cap S_1) \cap B_j = \emptyset$, and analogously $|S_{\mathcal{A}B_j}(I_2)| \leq \frac{9s_1}{10}$.
If $O \subseteq A_i$ for some i, then $(NT \cap S_1) \cap A_k = \emptyset$ for every $k \neq i$. Thus, $|S_{\mathcal{A}B_j}(I_{2+i})| \leq \frac{9s_1}{10}$.

If condition (ii) holds, i.e., $NT = \emptyset$ and both $|O \cap S_1| + |NT \cap S_1| < \frac{s_1}{10}$, and $|O \cap S_2| + |NT \cap S_2| < \frac{s_2}{10}$. In this case, the authors in [5] proved that we can find two subsets Y and Z of O with all edges between them and such that $|Z| \geq \frac{s_1}{10}$ and $|Y| \geq \frac{s_2}{10}$. Observe that since there are all edges between Y and Z then either $B_j \cap Z = \emptyset$ or $B_l \cap Y = \emptyset$.
We now define three new instances of ELSP according to the intersection of skew partition sets C or D with Z, as follows:

(a) I_1: $B_j \cap Z \neq \emptyset$,

(b) I_2: $B_l \cap Z \neq \emptyset$,

(c) I_3: $Z \subseteq \mathcal{A}$.

If $B_j \cap Z \neq \emptyset$, then $B_l \cap Y = \emptyset$. Thus, $(Y \cap S_2) \subseteq \mathcal{A}$, which implies $|S_{\mathcal{A}B_l}(I_1)| \leq \frac{9s_2}{10}$.
If $B_l \cap Z \neq \emptyset$, then an argument symmetric to $B_j \cap Z \neq \emptyset$ shows that $|S_{\mathcal{A}B_j}(I_2)| \leq \frac{9s_1}{10}$.
Otherwise, $Z \subseteq \mathcal{A}$, which implies $|S_{\mathcal{A}B_j}(I_3)| \leq \frac{9s_1}{10}$. ■

This ends the description of Procedure 2. ■

Procedure 3 is a mirror image of Procedure 2 and is omitted.

Procedure 4

We will give the details of the procedure constructing the 2 instances I_1, I_2 such that $|S_{B_p\mathcal{A}}(I_t)||S_{A_aB_b}(I_t)| \leq \frac{9}{10}|S_{B_p\mathcal{A}}(I)||S_{A_aB_b}(I)|$ with $b \neq p$. The other 2 instances I_3, I_4 are obtained similarly.
Let $S_{\mathcal{A}B_p} = S_1$ with $|S_{\mathcal{A}B_p}| = s_1$, and $S_{A_aB_b} = S_2$ with $|S_{A_aB_b}| = s_2$ and $b \neq p$. Given $v \in S_1 \cup S_2$, let $d_i(v) = |N(v) \cap S_i|$, $i = 1, 2$.

Case 1: There exists a vertex $v \in S_2$ with $\frac{s_1}{10} \leq d_1(v) \leq \frac{9s_1}{10}$.

This case is analogous to Case 1 of Procedure 1. Define two new instances of ELSP, as follows:

(a) I_1: $v \in A_a$,

(b) I_2: $v \in B_b$.

If $v \in A_a$, then every vertex of S_1 that is nonadjacent to v cannot be placed in B_b, so $|S_{\mathcal{A}B_p}(I_1)| \leq \frac{9s_1}{10}$.
If $v \in B_b$, then every vertex of S_1 that is adjacent to v cannot be placed in B_p. So $|S_{\mathcal{A}B_p}(I_2)| \leq \frac{9s_1}{10}$, as required. ■

Case 2: Every vertex $v \in S_2$ has either $d_1(v) < \frac{s_1}{10}$ or $d_1(v) > \frac{9s_1}{10}$.

Let $W = \{v \in S_2 : |N(v) \cap S_1| > \frac{9s_1}{10}\}$. We handle Case 2 according to the following two possibilities.

Case 2.1: $|W| > \frac{s_2}{2}$.

Let $v_1 \in W$. In [5], the authors proved that there are three sets O, T, and NT such that:

- O is contained in W;
- $S_1 = T \cup NT$;
- There are all edges between O and T;
- For every w in NT, there exists v in O such that v is not adjacent to w.

satisfying :

(i) $|T| \leq \frac{9s_1}{10}$; or

(ii) $|O| \geq \frac{s_2}{10}$.

If condition (i) holds, i.e., $|T| \leq \frac{9s_1}{10}$, then $|NT| \geq \frac{s_1}{10}$. In addition, they prove that we may assume that $|T| > \frac{8s_1}{10}$.

Consider two new instances of ELSP, as follows:

(a) I_1: $B_b \cap O \neq \emptyset$,

(b) I_2: $O \subseteq A_a$.

If $B_b \cap O \neq \emptyset$, then $T \cap B_p = \emptyset$ which implies $|S_{\mathcal{A}B_p}(I_1)| \leq \frac{9s_1}{10}$. Otherwise, $O \subseteq A_a$, and $NT \cap B = \emptyset$, since for every $w \in NT$ there is a vertex $v \in O$ such that $vw \notin E$. Thus $|S_{\mathcal{A}B_p}(I_2)| \leq \frac{9s_1}{10}$.

Now suppose that condition (ii) holds, i.e., $|O| \geq \frac{s_2}{10}$ and $|T| > \frac{9s_1}{10}$. Consider two new instances of ELSP, as follows.

(a) I_1: $B_p \cap T \neq \emptyset$,

(b) I_2: $T \subset \mathcal{A}$.

If $B_p \cap T \neq \emptyset$ then $O \cap B_b = \emptyset$, so $|S_{AB_b}(I_1)| \leq \frac{9s_1}{10}$.
Clearly, $T \cap S_{\mathcal{A}B_p}(I_2) = \emptyset$, so $|S_{\mathcal{A}B_p}(I_2)| \leq \frac{9s_1}{10}$. ■

Case 2.2: $|S_2 \setminus W| > \frac{s_2}{2}$.

Let $X = S_2 \setminus W$, and $v \in X$. In [5], they show that there exist two sets O, and M such that:

- $O \subseteq X$, $M \subseteq S_1$;

- There are no edges between M and O.

with :

(i) $|M| \leq \frac{9s_1}{10}$; or

(ii) $|O| \geq \frac{s_2}{10}$.

If condition (i) holds, i.e., $|M| \leq \frac{9s_1}{10}$ then the authors in [5] show that we may assume that $|M| > \frac{8s_1}{10}$. Then, in either case (i) or (ii), we have $|M| > \frac{8n_1}{10}$.

Define two new ELSP instances as follows;

(a) I_1: $A_a \cap O \neq \emptyset$,

(b) I_2: $O \subseteq B_b$.

If $A_a \cap O \neq \emptyset$, then $M \cap A_i = \emptyset$ for all $i \neq a$. Hence because $|M| > \frac{8s_1}{10}$, we have $|S_{\mathcal{A}B_p}(I_1)| < \frac{2s_1}{10} \leq 9s_1/10$.
Otherwise, $O \subseteq B_b$. In case (i), $|M| \leq 9s_1/10$, which implies $|S_1 \setminus M| \geq s_1/10$, so we have $|S_{\mathcal{A}B_p}(I_2) \leq \frac{9s_1}{10}$.
In case (ii), $|O| \geq \frac{s_2}{10}$, which implies $|S_{A_aB_b}(I_2)| \leq \frac{9s_2}{10}$. So for either output instance I_i $|S_{\mathcal{A}B_p}(I_i)||S_{A_aB_b}(I_i)| \leq \frac{9s_1s_2}{10}$ as required. ■

This ends the description of Procedure 4. ■

5 Details of Algorithm 5

In this section, we give the details of Algorithm 5 which we call *generalized 2-SAT* algorithm. This algorithm takes as input an instance of GEN-MAX-2-ELSP. The lists L present in such an instance are either equal to $\mathcal{A}$ or $\mathcal{B}$, or have size $|L| \leq 2$. We prove below that such restrictions are enough for a solution through an algorithm similar to that of Aspvall et al. [2] for 2-SAT.

Suppose an instance of GEN-MAX-2-ELSP is given, i.e., a graph $G = (V(G), E(G))$ and a set of lists L_v of the following types, trivial lists: A_1, A_2, …, A_{n_1}, B_1, B_2, …, B_{n_2}; lists of size 2: A_iB_j; lists of size greater than 2: the list $\mathcal{A}$ and the list $\mathcal{B}$.

Let $A_i, A_k \in \mathcal{A}$ and $B_j, B_l \in \mathcal{B}$ with $i \neq k$, $j \neq l$, $i, k \in \{1, \ldots, n_1\}$ and $j, l \in \{1, \ldots, n_2\}$.

We define next a digraph $\overrightarrow{G}_f = (V_f, E_f)$. Each directed edge of E_f corresponds to a "forcing" defined by the adjacency relation in the original graph G.

The vertex set of $\overrightarrow{G}_f$ is the following set $V_f = \{(u, I) : u \in V \textit{ and } I \in L_u\}$.

The edge set of $\overrightarrow{G}_f$ is the following set E_f:

If $u \in A_iB_j$ and $v \in A_kB_j$: $uv \notin E(G)$: $((u, A_i), (v, B_j))$ and $((v, A_k), (u, B_j))$.

If $u \in A_iB_j$ and $v \in A_iB_l$: $uv \in E(G)$: $((u, B_j), (v, A_i))$ and $((v, B_l), (u, A_i))$.

If $u \in A_iB_j$ and $v \in A_kB_l$: $uv \in E(G)$: $((u, B_j), (v, A_k))$ and $((v, B_l), (u, A_i))$; $uv \notin E(G)$: $((u, A_i), (v, B_l))$ and $((v, A_k), (u, B_j))$.

If $u \in A_iB_j$ and $v \in \mathcal{A}$: $uv \notin E(G)$: $((u, A_i), (v, A_i))$ and $((v, I), (u, B_j))$, $\forall I \in \mathcal{A} \setminus A_i$.

If $u \in A_iB_j$ and $v \in \mathcal{B}$: $uv \in E(G)$: $((u, B_j), (v, B_j))$ and $((v, J), (u, A_i))$, $\forall J \in \mathcal{B} \setminus B_j$.

If $u \in \mathcal{A}$ and $v \in \mathcal{A}$: $uv \notin E(G)$: $((u, A_i), (v, A_i))$ and $((v, A_i), (u, A_i))$, $\forall A_i \in \mathcal{A}$.

If $u \in \mathcal{B}$ and $v \in \mathcal{B}$: $uv \in E(G)$: $((u, B_j), (v, B_j))$ and $((v, B_j), (u, B_j))$, $\forall B_j \in \mathcal{B}$.

We define a *forcing class* $C(v, I)$ as the set of "forcings" induced by the choice of part I for vertex v, i.e., the set of vertices of $\overrightarrow{G}_f$ that we can reach starting from (v, I).

Proposition 3 *Let u and v be two vertices of G. If $(v, J) \in C(u, I)$ then for all $J' \in L_v \setminus \{J\}$, there exists $I' \in L_u \setminus \{I\}$ such that $(u, I') \in C(v, J')$.*

Proof. The proof is by induction on the number of edges in a path p from (u, I) to (v, J). If $|p| = 1$ then the path consists of a single forcing edge $((u, I), (v, J)) \in E_f$, with $u, v \in V(G)$, $I \in L_u$, $J \in L_v$. We consider the possibilities for L_u, L_v that correspond to forcing edges. In every case, the desired property holds.
Let p be a path, $|p| > 1$, from (u, I) to (v, J). Let $((y, M), (v, J))$ be the last edge in the path p. If we have the edge $((y, M), (v, J))$ then, for all $J' \in L_v \setminus J$, there exists an $M' \in L_y \setminus M$ such that $((v, J'), (y, M')) \in E_f$.
Since (y, M) is in the path p before (v, J), by induction, for all $M'' \in L_y \setminus M$, there exists a $I'' \in L_u \setminus I$ such that $(u, I'') \in C(y, M'')$. In particular for M', there exists $I' \in L_u \setminus I$ such that $(u, I') \in C(y, M')$.
Now, $(y, M') \in C(v, J')$ and $(u, I') \in C(y, M')$ imply the existence of $I' \in I \setminus L_u$ such that $(u, I') \in C(v, J')$, for all $J' \in L_v \setminus J$, as required. ∎

We say that the graph $\overrightarrow{G}_f$ admits an *Obstruction* if there exists a vertex $u \in V(G)$ such that for all $I \in L_u$, there is a $I' \in L_u \setminus I$ such that $(u, I') \in C(u, I)$. We can decide in polynomial time whether $\overrightarrow{G}_f$ admits an obstrution by computing the strong connected components of the digraph $\overrightarrow{G}_f$.

Proposition 4 *The digraph $\overrightarrow{G}_f$ admits an obstruction, if and only if the corresponding instance of GEN-MAX-2-ELSP has no solution.*

Proof. The definition of obstruction immediately implies that the corresponding instance of GEN-MAX-2-ELSP has no solution.
Suppose the digraph $\overrightarrow{G}_f$ admits no obstruction. So, by hypothesis, every vertex $u \in V(G)$, has a *safe* part $F_u \in L_u$ such that $(u, I) \notin C(u, F_u)$, for all $I \in L_u \setminus F_u$.
Define a solution for the corresponding instance of GEN-MAX-2-ELSP as follows. Choose an arbitrary vertex $u \in V(G)$ and place u in its safe part F_u. Note that, if $x \in V(G)$ is such that $(x, K) \in C(u, F_u)$, then $(x, K') \notin C(u, F_u)$, for all $K' \in L_x \setminus K$, as otherwise by Proposition 3 we have a contradiction to our hypothesis. Thus, we may place accordingly x in part K, for all $(x, K) \in C(u, F_u)$.
While there exists $w \in V(G)$ not placed, repeat the above rule by placing w in a safe part F_w and by placing accordingly all vertices y such that $(y, T) \in C(w, F_w)$.
Suppose there exists $x \in V(G)$ such that $(x, K) \in C(u, F_u)$ and $(x, K') \in C(w, F_w)$. Then Proposition 3 implies the existence of $K'' \in L_w \setminus F_w$ such that $(w, K'') \in C(x, K)$ and hence $(w, K'') \in C(u, F_u)$, which contradicts that w was not placed by placement of vertex u. ■

6 Conclusion

It is evident to the authors that the techniques we have developed will apply to large classes of list-M-partition problems. For instance, we have studied the concept of H-partition which includes all vertex partitioning problems into nonempty parts with only external restrictions according to the structure of a model graph H. In the present paper, we presented an algorithm for the case where H contains $n_1 + n_2$ vertices such that n_1 vertices induce a clique and n_2 vertices induce a stable set. All cases when H has four vertices are studied in [4].

We would like also to make some observations about the status of n_1 and n_2. Our algorithm depends on the values of n_1 and n_2. If a graph admits an (n_1, n_2)-extended skew partition, then it admits an (n'_1, n'_2)-extended skew partition, for any pair n'_1, n'_2 satisfying $n'_1 \leq n_1$, $n'_2 \leq n_2$. This monotonicity property suggests the following combinatorial optimization problem : Find the largest values of n_1 and n_2 such that a graph G admits an (n_1, n_2)-extended skew partition, which stated as a decision problem gives:

Maximum Skew Partition Problem
Input: a graph $G = (V, E)$, and integers n_1, n_2.
Question: Is there a (k, l)-extended skew partition with $k \geq n_1$, $l \geq n_2$?

We conjecture that this problem is NP-complete and we propose the study of its complexity status as an open problem.

We believe that studying the (n_1, n_2)-extended skew partition problem contributes to a better understanding of the techniques that were used to solve both problems: skew partition and (n_1, n_2)-extended skew partition, and that soon it will be possible to reduce the high complexity of the polynomial-time algorithms known to solve both problems.

References

[1] N. Alon and M. Tarsi. Colorings and orientations of graphs. *Combinatorica* **12** (1992) 125–134.

[2] B. Aspvall, F. Plass and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Infor. Process. Lett.* **8** (1979) 121–123.

[3] V. Chvátal. Star-cutsets and perfect graphs. *Journal of Combinatorial Theory Series B* **39** (1985) 189–199.

[4] S. Dantas, C. M. H. de Figueiredo, S. Gravier and S. Klein. *On H-partition problems.* Relatório Técnico COPPE/Engenharia de Sistemas e Computação, ES-579/02, Maio - 2002.

[5] C. M. H. de Figueiredo, S. Klein, Y. Kohayakawa, and B. Reed. Finding skew partitions efficiently. *Journal of Algorithms*, 37 (2000) 505–521.

[6] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory Series B* **72** (1998) 236–250.

[7] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica* **19** (1999) 487–505.

[8] T. Feder, P. Hell, S. Klein, and R. Motwani. Complexity of graph partition problems. *In Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing* (1999) 464–472.

[9] H. Fleischner and M. Stiebitz. A solution of a coloring problem of P. Erdős. *Discrete Mathematics* **101** (1992) 39–48.

[10] P. Hell, S. Klein, L. T. Nogueira, and F. Protti. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, to appear.

[11] P. Hell, S. Klein, L. T. Nogueira, and F. Protti. Independent K_r's in chordal graphs. *In Proceedings of CLAIO'2002 - XI Latin-Iberian American Congress of Operations Research*, to appear.

[12] C. Hoàng, K. Cameron, E. Eschen, and R. Sritharan List partitions. *In Perfect Graph Conjecture workshop*, P. Seymour and R. Thomas, organizers, http://www.aimath.org/ARCC/workshops/perfectgraph.html, American Institute of Mathematics, 2002.

[13] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas. Strong Perfect Graph Theorem. *In Perfect Graph Conjecture workshop*, P. Seymour and R. Thomas, organizers, http://www.aimath.org/ARCC/workshops/perfectgraph.html, American Institute of Mathematics, 2002.