Complexity of Graph Partition Problems

Tomas Feder* Pavol Hell† Sulamita Klein‡ Rajeev Motwani§

December 2, 1998

Abstract

We introduce a parametrized family of graph problems that includes several well-known graph partition problems as special cases. We develop tools which allow us to classify the complexity of many problems in this family, and in particular lead us to a complete classification for small values of the parameters. Along the way, we obtain a variety of specific results including the following: a generalization of a communication bound on the number of clique-versus-independent-set separators; polynomial-time algorithms to recognize generalized split graphs; and, a quasi-polynomial algorithm for the Skew Cutset Problem that essentially resolves an open problem posed by Chvátal. The last two problems have interesting connections to the Strong Perfect Graph Conjecture of Berge.

*E-mail: `tomas@theory.stanford.edu`.

†Simon Fraser University, Burnaby, B.C., Canada, V5A1S6. E-mail: `pavol@cs.sfu.ca`.

‡Departamento da Ciência da Computação-I.M., COPPE/Sistemas, Universidade Federal do Rio de Janeiro, RJ, 21945-970, Brasil. E-mail: `sula@cos.ufrj.br`. Supported by CNPq and PRONEX 107/97.

§Department of Computer Science, Stanford University, CA 94305-9045. E-mail: `rajeev@cs.stanford.edu`.
Supported by an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04–96–1–0007, and NSF Young Investigator Award CCR–9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

1 Introduction

Many combinatorial problems can be described as finding a partition of the vertices of a given graph into subsets satisfying certain properties *internally* (some parts may be required to be independent, or sparse in some other sense, others may conversely be required to be complete or dense), and *externally* (some pairs of parts may be required to be completely nonadjacent, others completely adjacent). We define a parametrized family of graph problems of this type and develop tools to classify the complexity of many problems in this family. Of particular interest is the consequent resolution of a couple of open problems with interesting connections to the Strong Perfect Graph Conjecture of Berge: polynomial-time algorithms to recognize generalized split graphs; and, a quasi-polynomial algorithm for the Skew Cutset Problem that (almost) resolves an open problem posed by Chvátal.

The basic family of problems we consider is as follows: partition a graph into k parts $A_1, A_2, \ldots, A_k$ with a fixed "pattern" of requirements as to which A_i are independent or complete and which pairs A_i, A_j are completely nonadjacent or completely adjacent. (In some cases, we can also deal with a generalization where we replace "independent" and "complete" with a more general notion of "sparse" and "dense" graphs.) These requirements may be conveniently encoded by a symmetric k-by-k matrix M in which the diagonal entry $M_{i,i}$ is 0 if A_i is required to be independent, 2 if A_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 A_i and A_j are required to be completely nonadjacent, have arbitrary connections, or are required to be completely adjacent, respectively. We call such a partition an *M-partition*[1].

Many combinatorial problems just ask for an M-partition. For instance a k-coloring is an M-partition where M is the adjacency matrix of the complete k-graph, and, more generally, H-coloring (homomorphism to a fixed graph H [14]) is an M-partition where M is the adjacency matrix of H. It is known that H-coloring is polynomial-time solvable when H is bipartite and NP-complete otherwise [14]. When M is the adjacency matrix of H plus twice the identity matrix (all diagonal elements are 2), then M-partitions reduce to the so-called (H, K)-partitions which were studied by MacGillivray and Yu [18]. When H is triangle-free then (H, K)-partition is polynomial-time solvable, otherwise it is NP-complete.

Other well-known problems ask for M-partitions in which all parts are restricted to be nonempty (e.g., skew partitions, clique cutsets, stable cutsets). In yet other problems there are additional constraints, such as those in the definition of a homogeneous set (requiring A to have at least two and at most $n-1$ vertices). For instance, Winkler asked for the complexity of deciding the existence of an M-partition, where M has the rows $1101, 1110, 0111$, and 1011, such that all parts are nonempty and there is at least one edge between parts A and B, B and C, C and D, and D and A. This has recently been shown NP-complete by Vikas [22].

The most convenient way to express these additional constraints turns out to be to allow specifying for each vertex (as part of the input) a "list" of parts in which it 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, 11]. They also generalize list-homomorphisms which were studied earlier [8, 9, 10].

[1] In the sequel, when k is small, we usually refer to parts A, B, C, ... instead of A_1, A_2, A_3, ..., and write, for example, $A = 0$ to mean $M_{A,A} = 0$, and $AB = 2$ instead of $M_{A,B} = 2$.

1.1 Main Results and Related Work

In Section 2 we define a generalized problem called a *sparse-dense partitions* and give algorithms for various versions of this problem. We also resolve the complexity of the generalized split graph problem defined as follows. A graph is a *split graph* [13] if its vertices can be partitioned into an independent set and a clique. Split graphs can be recognized in polynomial time, and admit polynomial-time optimization algorithms [13]. Recently there has been new interest in generalizations of split graphs. Indeed, Brandstadt [2] introduced the concepts of $(2,1)$-graphs and $(2,2)$-graphs, as graphs that can be partitioned into two independent sets and a clique, or two independent sets and two cliques, respectively. Hoang [15] proved that these generalized split graphs satisfy the Strong Perfect Graph Conjecture of Berge [13]. Brandstadt gave polynomial-time algorithms for the recognition of these generalized split graphs [2]; however, these algorithms are incorrect[2]. We describe a general context in which these and other similar problems can be solved in polynomial time.

In Section 3, we define the notion of a *sparse-dense separator* which separates a graph into dense and sparse subgraphs. We generalize a communication bound on the clique-versus-independent-set separators to more general dense and sparse subgraphs. We apply this result to devising a tool for solving the list-M-partition problem.

In Section 4, we characterize the complexity of the list-M-partition problems in the cases where M is a 3-by-3 or 4-by-4 matrix. As a consequence we obtain a (near-)resolution of the Skew Cutset Conjecture of Chvátal that is described next.

A *stable cutset* in a graph G corresponds to a partition of $V(G)$ into three nonempty sets A, B, C such that B is independent and A, C are completely nonadjacent (in other words, no vertex of A is adjacent to any vertex of C). Tucker [21] proved that a minimum counterexample to the Strong Perfect Graph Conjecture does not admit a stable cutset; de Figueiredo and Klein [12] showed that recognizing graphs with a stable cutset is NP-complete. A *clique cutset* similarly corresponds to a partition of the vertex set into three nonempty sets A, B, and C, such that B is complete and A and C are completely nonadjacent. Clique cutsets can be found in polynomial time, and lead to efficient decomposition algorithms which are useful for solving optimization problems [20, 23, 24]. A *homogeneous set* corresponds to a partition of the vertex set into sets A, B, and C, such that A, B are completely adjacent, A and C are completely nonadjacent, and A has more than one but fewer than all vertices. Homogeneous sets lead to modular decompositions which yield the most efficient recognition algorithms for certain classes of graphs [6, 19].

One of the main open problems in this area is the *Skew Cutset Conjecture* of Chvátal. A *skew partition* of a graph is a partition of the vertex set into nonempty sets A, B, C, and D, such that A, C are completely adjacent, and B, D completely nonadjacent. Chvátal [5] conjectured that a minimal imperfect graph does not have a skew partition, and he proved this for the special case when the set A has exactly one vertex. In this special case he has also given a polynomial-time algorithm for recognizing graphs which admit such a special skew partition. He asked for the complexity of the recognition problem for graphs which admit a general skew partition; this has remained an open problem. We will give a quasi-polynomial algorithm for this problem, strongly suggesting that it is not NP-complete.

[2] According to the corrigendum [3] a more involved polynomial-time algorithm was given later [4].

2 Sparse-Dense Partitions

Consider first the "generalized split graphs" discussed above. In these cases the main diagonal of M has a occurrences of 0's and $b = n - k$ occurrences of 2's (but no 1's), while all the off-diagonal entries are 1. An M-partition can then be viewed as a partition into two parts, one of which is the union of a independent sets (an a-colorable graph), and the other a union of b cliques (the complement of a b-colorable graph). Split graphs have $a = b = 1$, $(2,1)$-graphs have $a = 2$ and $b = 1$, while $(2,2)$-graphs have $a = b = 2$.

To consider a wider class of such problems we introduce the following model: Let $\mathcal{S}$ and $\mathcal{D}$ be two classes of graphs, called *sparse* and *dense* respectively, such that any $S \in \mathcal{S}$ and $D \in \mathcal{D}$ have at most c vertices in common, for some constant c. A *sparse-dense partition* of a graph G (with respect to $\mathcal{S}$ and $\mathcal{D}$) is a partition of $V(G)$ into two parts, one of which is sparse and the other dense.

Theorem 1 *A graph on n vertices has at most n^{2c} different sparse-dense partitions. Furthermore, all these partitions can be found in time $n^{2c+2}T(n)$, where $T(n)$ is the time for recognizing sparse and dense graphs.*

PROOF SKETCH. We perform a $(2c+1)$-local search to maximize the number of vertices in a sparse subgraph. Suppose there is a partition (S, D) into sparse and dense subgraphs. Let S' be a sparse subgraph that has been found at some stage of the search, and suppose S' has fewer vertices than S. Since S' intersects D in at most c vertices, we can remove $c' \leq c$ vertices from S' and obtain a subgraph of S. We can then add $c' + 1$ vertices from S and obtain a sparse subgraph S'' with one more vertex than S'. We have thus made progress by changing S' in at most $2c + 1$ vertices, as desired.

When the search terminates, since it can not find a larger sparse subgraph, the subgraph S' obtained has at least as many vertices as S for every pair (S, D) in a partition into sparse and dense subgraphs. Now perform a $2c$-local search step, but this time test the subgraph S'' obtained and the subgraph D'' induced by the remaining vertices to check whether (S'', D'') is a pair of sparse and dense subgraphs. This finds all the pairs (S, D), because as before we can remove $c' \leq c$ vertices from S' to obtain a subgraph of S and then add at most c' vertices to obtain S, by the cardinality constraint.

The last step examines n^{2c} candidate pairs (S'', D''), proving the claimed n^{2c} bound on the number of partitions (S, D) into sparse and dense subgraphs. Furthermore, each step of the local search examines n^{2c+1} candidate sets S'', and the search lasts for at most n steps, proving the time bound. □

It may in general be hard to test sparse and dense subgraphs. For instance, if we take as sparse graphs the c-colorable graphs, then the test is NP-complete for $c \geq 3$. We note that this test is needed to determine whether a partition exists.

Proposition 1 *Suppose that the disjoint union of sparse graphs is also sparse. If testing sparse graphs is NP-complete, then the partition problem into sparse and dense graphs is also NP-complete.*

PROOF SKETCH. Suppose we wish to test whether G is sparse. We can construct G' by taking the disjoint union of $c + 1$ copies of G. Then G' has a sparse-dense partition if and only if G is sparse. Indeed, if G is sparse then G' is sparse by the assumption; on the other hand, if G' admits a partition, then our algorithm will find a pair (S, D) where D has at most c vertices. Therefore one of the copies of G is contained in S, so G is sparse. □

We have defined sparse and dense subgraphs with respect to each other, since the definition depends on the existence of a constant c bounding the intersection of the two. The next result shows that we can define sparse and dense graphs independently, under the assumption that independent sets are always sparse and cliques are always dense.

Proposition 2 *Suppose we have a class of sparse graphs with respect to the cliques, and a class of dense graphs with respect to the independent sets, with some associated constant c. Then the two classes are sparse and dense with respect to each other, with some associated constant c'.*

PROOF SKETCH. The sparse graphs cannot contain a $(c+1)$-clique by the intersection constraint. Similarly, the dense graphs cannot contain a $(c+1)$-independent set by the intersection constraint. The intersection of sparse and dense subgraphs is a graph that contain neither a $(c+1)$-clique nor a $(c+1)$-independent set. Such a graph has size bounded by some $c' = f(c)$ by Ramsey Theorem. □

This result makes it easy to find sparse and dense classes. Examples of sparse classes are independent sets, bipartite graphs, $(c+1)$-clique-free graphs, planar graphs, and c-colorable graphs. The last one is NP-complete for $c \geq 3$, the remaining are polynomial-time solvable. As examples of dense classes, we can take the complements, viz., cliques, cobipartite graphs, graphs without $(c+1)$-independent sets, complements of planar graphs, and complements of c-colorable graphs. We now return to the case of generalized split graphs. The proof of the following corollary is omitted.

Corollary 1 *Let M be a matrix with all off-diagonal entries equal to 1 and with $a \leq 2$ diagonal entries equal to 0 and $b = k - a \leq 2$ diagonal entries equal to 2. Then the list-M-partition problem is polynomial-time solvable.*

Note that the matrix M has $k \leq 4$. In the next section we classify the complexity of all list-M-partition problems for matrices M with $k \leq 4$ with no 1's on the main diagonal, i.e., also in the cases when not all the off-diagonal entries are 1.

The above corollary yields a correct algorithm for both the problems considered by Brandstadt [2]. Note that if a or b is greater than two then the problem is NP-complete by Proposition 1.

Another situation in which we can directly apply our theorem is one where the matrix M contains no 1's at all:

Corollary 2 *If every entry of M is 0 or 2, then the list-M-partition problem is polynomial-time solvable.*

PROOF SKETCH. Once we know from Theorem 1 which vertices map to parts of type 0 and which to parts of type 2, we can find the equivalence classes where two vertices that both map to parts 0 are equivalent if they have the same open neighborhood, and two vertices that both map to parts 2 are equivalent if they have the same closed neighborhood. Having these equivalence classes in hand, we can easily check if a partition exists. (Recall that the size k of the matrix M is fixed.) □

Note that if every entry of M is 0 or 1, the problem has been studied in [8, 9, 10]. We give a complete classification when all diagonal entries of M are the same (all 0 or all 1). For instance, when all the diagonal entries are 1 we prove that the list-M-partition problem is polynomial-time solvable if M is the adjacency matrix of an interval graph and is NP-complete otherwise [8]. By complementation, we also obtain corresponding results for matrices M in which every entry 1 or 2.

3 Sparse-Dense Separators

In general, a sparse-dense partition may not exist. It is then interesting to find sets that separate the sparse subgraphs from the dense subgraphs. We say that a set of vertices *separates* a pair of disjoint sparse and dense subgraphs if it is disjoint from the sparse subgraph and contains the dense subgraph. We wish to know whether there exists a small collection of sets that separate every pair of disjoint sparse and dense subgraphs. For the case where the sparse and dense graphs are the independent sets and the cliques, it is known that there is a collection of at most $2^{(\log^2 n)/2}$ separators [17]. It is not known whether this bound can be improved to a polynomial. The bound is obtained by setting up a communication game between two parties. For a given graph, the first party is given an independent set S and the second party is given a clique D. They wish to prove that the two are disjoint, with minimal communication. The communication bound of $\log^2 n$ bits is obtained as follows. If the independent set has a vertex v of degree at least $\frac{n-1}{2}$, then the first party sends the name of v. The two parties then put v in $\overline{E}$ and the neighbors of v in E, since they cannot belong to S. These vertices are removed from the graph. Similarly, if the clique has a vertex w of degree at most $\frac{n-1}{2}$, then the second party sends w, and then w is put in E and the non-neighbors of w in $\overline{E}$. Since the graph is halved at each step, at most $\log n$ rounds of communication take place, each requiring $\log n$ bits. If no appropriate v or w can be found, then the vertices of degree at most $\frac{n-1}{2}$ can be put in $\overline{E}$ and the vertices of degree greater than $\frac{n-1}{2}$ can be put in E. Notice that E depends only on the vertices communicated, proving the $2^{\log^2 n}$ bound. A more careful analysis [17] shows that the bound is actually $2^{\binom{1+\log(n+1)}{2}}$.

We have obtained two generalizations of this result for other definitions of sparse and dense graphs:

Theorem 2 *Let the sparse graphs be the a-colorable graphs, and the dense graphs the complements of b-colorable graphs. Then there is a collection of at most $2^{(ab\log^2 n)/2}$ separators.*

Proof Sketch. Sparse graphs decompose into a independent sets and dense graphs decompose into b cliques. Thus a sparse graph can meet a dense graph in at most $c = ab$ vertices.

We know that $2^{\binom{1+\log(n+1)}{2}}$ separators E are sufficient to separate each independent set from each clique. If we separate each of the a independent sets from a sparse subgraph and each of the b cliques from a dense subgraph, we obtain $c = ab$ such sets E. We can then construct a separator E' for the sparse and dense subgraphs by taking, for each of the b cliques, the intersection of the a separators E corresponding to the a independent sets, and then letting E' be the union of the b intersections corresponding to the b cliques. Since there are at most $2^{\binom{1+\log(n+1)}{2}}$ separators E, and the separator E' is constructed from $c = ab$ such separators, the $2^{c\binom{1+\log(n+1)}{2}}$ bound follows. □

In general, sparse graphs are $(a + 1)$-clique-free for some constant a, and dense graphs are $(b + 1)$-independent-set-free for some constant b. We have also proved the following result, whose proof is omitted.

Theorem 3 *The $(a + 1)$-clique-free graphs and the $(b + 1)$-independent-set-free graphs can be separated with $2^{C(a+b,n)}$ separators, where $C(t, n) \leq \binom{t+\log(n+1)}{t} - 1 - \log(n + 1)$ is the solution of the recurrence $C(t, 0) = C(1, n) = 0$, $C(t, n) = \log(n + 1) + \max_{\frac{n-1}{2} \leq l < n} C(t - 1, l) + C(t, n - l - 1)$.*

We now explain how to apply the separator theorem (Theorem 2) to list-M-partitions. Suppose M has a diagonal entries equal to 0 and b equal to 2 (and possibly some others equal to 1). Define the *restricted-list-M-partition* problem, where none of the lists of the input graph contain parts of both types 0 and 2. In some situations the restricted-list-M-partition problem may be easy to solve — for instance by 2-SAT.

Theorem 4 *The general list-M-partition problem reduces to* $2^{ab\binom{1+\log(n+1)}{2}}$ *instances of the restricted-list-M-partition problem.*

Proof Sketch. This is the bound in Theorem 2 on the number of separators E to contain the vertices that map to type 2 and disjoint from the vertices that map to type 0. If E is such a separator, then the vertices in E cannot map to type 0 and the vertices not in E cannot map to type 2, as desired. □

Thus in situations where the restricted-list-M-partition problem is polynomial-time solvable, we obtain a quasi-polynomial algorithm for list-M-partitions. Examples of this will be given in the next section.

4 M-partitions for Small Matrices M

Recall that most of our motivating examples of M-partitions dealt with small values of k. (Split graphs have $k = 2$, stable set partition, clique partition, and homogeneous set have $k = 3$, skew partition and the problem of Winkler have $k = 4$, and so on.) This suggests a systematic investigation of M-partition problems with small k; we focus on $k \leq 4$.

When M is of size $k = 2$, we can solve the list-M-partition problem by introducing a variable for each vertex of the input graph G (encoding whether or not the vertex belongs to one fixed part of the partition). It is then easy to see that all the constraints, and lists, can be reflected by polynomially many clauses with at most two literals each, and hence solved by 2-SAT. The same technique applies any time we have an instance G in which every list has size at most two.

To deal with matrices of larger size, we introduce the following concepts:

- We say that X *dominates* Y if for each Z (possibly equal to X, Y) we have $M_{X,Z} = M_{Y,Z}$ or $M_{X,Z} = 1$. If X dominates Y, we can eliminate Y from any list containing $\{X, Y\}$, since any vertex that maps to Y can be mapped to X instead.

- We say that M is *disconnected* if it can be written as two-by-two block matrix with diagonal blocks M_1, M_2 and off-diagonal blocks having all entries 0. If M is disconnected and if both the list-M_i-partition problems can be solved in polynomial time, then also the list-M-partition problem is polynomial-time solvable, since each component of the input graph must map to M_1 or M_2 entirely.

- We say that a matrix M *contains* a matrix M', if M' is a principal submatrix of M. In other words, the parts of the M'-partition problem are a subset of the parts of the M-partition problem, with the same constraints and connections. If M contains M' and list-M'-partition is NP-complete then so is list-M-partition, as we can use the lists to restrict the input vertices to map to M'.

- The *complement* M^- of a matrix M is obtained from M by replacing each entry e by $2 - e$. Clearly (list) M-partitions of G correspond to (list) M^--partitions of the complement of G.

Suppose a row X of M has both a 0 and a 2, say $XY = 0$ and $XZ = 2$ (either of Y, Z could be X). In this case it suffices to make at most $n + 1$ checks: first check whether or not the input graph G has a partition in which no vertex lies in part X, and then check for each vertex v of G

which has X in its list whether or not G has a partition with v in X. The former can clearly be accomplished by removing X from all lists, the latter can be tested by placing v in X and updating the lists of all other vertices of G as follows: For neighbors of v we remove Y from the lists, and for nonneighbors of v we remove Z from the lists. Thus we obtain one instance with no list containing X, and at most n instances with no list containing both Y and Z. This is particularly useful when $k = 3$ as in this case all lists become of size at most two.

Theorem 5 *Suppose the size of M is $k = 3$. Then, all list-M-partition problems are polynomial-time solvable with the following exceptions:*

- *the 3-coloring problem (M has rows* $011, 101, 110$*),*
- *the stable-cutset problem (M has rows* $110, 011, 101$*),*
- *and, their complements (*$211, 121, 112$ *and* $112, 211, 121$*).*

PROOF SKETCH. The exceptions are standard results [16, 12].

Consider now a matrix M with rows A, B, C and connections AB, AC, BC, which is different from the exceptions' matrices listed in the theorem. We may assume that M is connected, thus at most one of the connections is 0. We may also assume that no row has both a 0 and a 2, cf. above, and, in particular, we cannot have both a connection 0 and a connection 2. Thus without loss of generality assume that $AC = BC = 1$; we may also assume that $C \neq 1$, otherwise C dominates all other rows.

If each part A, B, C is of type either 0 or 2 and not all are the same type, then the problem is polynomial-time solvable by our Theorem 1, since after we have decided which vertices map to parts of type 0 and which to parts of type 2, we are left with each list of size one or two.

If $AB = 1$ as well, then we also have $A, B \neq 1$ and so we may assume that all three values A, B, C are the same, 0 or 2. This is impossible, as M is different from the matrix M for 3-coloring and its complement.

Up to complementation we may assume that $AB = 0$. We may also assume that $A, B \neq 2$, else we have a row with both a 0 and a 2. If one of A, B dominates the other, we reduce all lists to size at most two. The only other possibility is that $A = B = 1$. Since M is not the matrix of the stable-cutset problem we conclude that $C = 2$, i.e., it is the matrix of the clique-cutset problem. A polynomial time algorithm for this problem, when all lists are $\{A, B, C\}$ was given by Tarjan [20] and Whitesides [23]. We now explain how to solve the general case of "clique cutset problem with lists."

Let $A = B = 1$, $C = 2$, $AB = 0$, and $AC = BC = 1$. The following is a sequence of problems where each one reduces to the next problem in the sequence.

1. Vertices have lists among $\{A\}, \{B\}, \{C\}, \{A, B\}, \{A, C\}, \{B, C\}, \{A, B, C\}$. This is the list problem we are trying to solve.

2. Vertices have lists among $\{A\}, \{B\}, \{C\}, \{A, B, C\}$. We must get rid of the two-element lists. Replace the two-element lists with the list $\{A, B, C\}$, but add a single vertex to the graph, as follows:

   - If v was labeled $\{A, B\}$, add a vertex w labeled $\{C\}$ adjacent to all vertices but v; clearly, whatever clique C the original problem had can be extended to include w, but v cannot be in the clique since it is not adjacent to w.

- If v was labeled $\{A, C\}$, add a vertex w labeled $\{A\}$ adjacent to v; clearly, v cannot be B because of its adjacency to w.
- If v was labeled $\{B, C\}$, add a vertex w labeled $\{B\}$ adjacent to v; again, v cannot be A because of its adjacency to w.

3. Single vertex for each of the lists $\{A\}, \{B\}, \{C\}$; all others are $\{A, B, C\}$. We must merge lists:

   - Merge all vertices v labeled $\{C\}$, checking that these already form a clique, into a single vertex w whose neighbors are the common neighbors of all merged v.
   - Merge all vertices v labeled $\{A\}$ into a single vertex w whose neighborhood is the union of the neighborhoods of all merged v.
   - Merge all vertices v labeled $\{B\}$ into a single vertex w whose neighborhood is the union of the neighborhoods of all merged v.

4. Single vertex for each of the lists $\{A\}, \{B\}$; all others are $\{A, B, C\}$. We must replace the single $\{C\}$ label with $\{A, B, C\}$.

   - Add edges joining pairs of vertices with lists $\{A\}$ and $\{C\}$, and with lists $\{B\}$ and $\{C\}$, and replace $\{C\}$ label with $\{A, B, C\}$. Clearly, to separate $\{A\}$ and $\{B\}$ we map $\{C\}$ in the clique, so the label $\{C\}$ is not needed.

5. All lists are $\{A, B, C\}$, and we want nonempty parts. This is the problem solved by Tarjan [20] and Whitesides [23].

Now, solve the last problem. If the clique found separates the $\{A\}$ and $\{B\}$ vertices, we are done. Otherwise the clique found must create at least one component containing neither the $\{A\}$ nor the $\{B\}$ vertices; remove this component from the graph. We have thus created a smaller but equivalent instance of (4). Repeat as many times as necessary, i.e., solve (5) on the smaller instance, and so on. □

We now proceed to say something about matrices of size $k = 4$. Giving a complete classification of the list version of these problems would completely resolve the skew partition problem. While we cannot achieve this goal fully, we do give a partial classification that will lead to a quasi-polynomial algorithm for skew partitions. This is sufficiently strong evidence that the problem is not NP-complete.

Theorem 6 *Suppose M is of size $k = 4$ and assume it does not contain any 1's on the main diagonal. Then*

- *if M contains the matrix corresponding to 3-coloring, then the list M-partition problem is NP-complete*
- *otherwise the list-M-partition problem is polynomial-time solvable.*

PROOF SKETCH. The first statement is obvious. Thus assume M does not contain 3-coloring.

If there are two parts of type 0 and two of type 2, we proceed as the case of $k = 3$, solving, for each sparse-dense partition, the remaining problem by 2-SAT. If there are three parts of type 0 and one part of type 2 (or three 2's and one 0), then we proceed the same way, since the three 0 parts do not yield 3-coloring and all other three-part problems without a diagonal 1 are polynomial time solvable. Once the vertices that go to the part of type 2 are known, we can remove them, modifying the remaining lists, and solve the three-part subproblem.

By complementation, we may now assume that all four parts are of type 0. If some row X has two 2's, then we can reduce the problem (by trying to place a vertex into X) to one instance with lists without X (polynomial-time solvable since M does not contain 3-coloring), and at most n instances with lists of size at most two.

Suppose there are two disjoint connections of type 2, say $AB = CD = 2$. Then we may try to place two vertices — v into A and w into C: for vertices adjacent to both v and w we can remove A, C from the lists, for those nonadjacent to both we can remove B, D, for those adjacent to v but not to w we can remove A, D, and for those adjacent to w but not v we can remove B, C. Thus we can reduce the problem to the following polynomial-time solvable instances: One instance with lists without A, one instance with lists without C, and at most n instances with lists of size at most two.

Similar arguments apply when (say) $AB = 2$ and both A and B have a connection of type 0, or when (say) $AB = BC = AC = 2$ — we try placing a vertex into A and a vertex into B.

The remaining cases can all be solved using domination. We only illustrate the technique with the following case: Suppose $AC = 0, BD = 2$ (all other connections being 1, and $A = B = C = D = 0$). Since A dominates C we can assume no list contains both A and C. Now try placing a vertex v into B: vertices adjacent to v have lists contained in $\{A, C, D\}$ but not containing both A and C, and vertices nonadjacent to v have lists contained in $\{A, B, C\}$ but again not containing both A and C. Thus we have reduced the problem to one instance with lists not containing B (polynomial-time solvable), and at most n instances with lists of size at most two. (We note here that the complement of this particular problem has been studied by Everett, Klein, and Reed [7], where a linear-time algorithm is given.) □

Corollary 3 *If the size of M is $k = 4$, then the M-partition problem (without lists) is*

- *NP-complete when M contains 3-coloring*
- *polynomial-time solvable otherwise*

There are a number of interesting cases when some parts are of type 1 which we now discuss.

First we illustrate the technique of Theorem 2. Consider the case where $A = B = 0, C = 1, D = 2, AC = BC = 2$, and $AB = AD = BD = CD = 1$. Here, for the restricted list problem, the lists are contained either in $\{A, B, C\}$ or in $\{C, D\}$. The lists contained in $\{C, D\}$ allow only two possible values. As for the lists contained in $\{A, B, C\}$, we can consider a vertex v that maps to A, its neighbors must map to $\{B, C\}$ and its nonneighbors must map to $\{A, B\}$, so again there are only two possible values. The restricted list problem can therefore be solved in polynomial time by 2-SAT, and so we obtain a quasi-polynomial list-M-partition algorithm.

The most interesting unsolved case of size $k = 4$ is that of the Skew Cutset Problem, i.e., $AC = 0$, $BD = 2$, and all others 1. The complexity of this problem is a well-known open problem in the theory of perfect graphs. We have the following partial resolution strongly suggesting that the problem is unlikely to be NP-complete:

Theorem 7 *There is a quasi-polynomial algorithm for the Skew Cutset Problem.*

Proof Sketch. The problem can be solved in $2^{\log^2 n} poly(n)$ time, as follows. During the algorithm, if any vertex has a single-element list, we eliminate it and restrict its neighbors and nonneighbors accordingly. Let S be the set of vertices whose lists contain $\{A, B\}$, and let T be the set of vertices whose lists contain $\{C, D\}$. We describe $\log |S|$ vertices that go to either A or B to make either S or T empty, as follows. Say that a vertex in T has high degree if it is adjacent to at least half the vertices in S, and that it is low degree if it is adjacent to less than half the vertices in S. If a high degree vertex in T goes to part C, then its neighbors cannot go to part A, reducing S to half the size. Similarly, if a low degree vertex in T goes to part D, then its non-neighbors cannot go to part B, reducing S to half the size. After $\log |S|$ such steps, either S is empty, or no appropriate high or low degree vertices in T can be found, in which case high degree vertices in T do not go to C, and low degree vertices in T do not go to D, making T empty. Say T is empty. Since all lists contain at least two elements, they must all contain A or B, so we can put all vertices in parts A or B, since there are no restrictions involving A or B alone. $\square$

There is a similar problem we can solve in quasi-polynomial time: The *two-clique separator problem*, that is, $AC = 0, B = D = 2$, all others 1, can be solved in $2^{2\log^2 n} poly(n)$ time, as follows. Let S' be the set of vertices whose lists contain $\{A, B\}$, and let T' be the set of vertices whose lists contain $\{B, C\}$. As before, we define high and low degree vertices in T', with respect to S', and after specifying for at most $\log |S'|$ vertices in T' that they go to B or to C, either S' is empty, or T' can be made empty. Similarly, letting S'' be the set of vertices whose lists contain $\{A, D\}$, and T'' be the set of vertices whose lists contain $\{D, C\}$, after at most $\log |S''|$ steps, either S'' or T'' is empty. Now, if all lists contain only two elements, solve the 2SAT problem. Otherwise, a three-element list cannot be $\{A, B, C\}$, which would be in both S' and T', and cannot be $\{A, D, C\}$, which would be in both S'' and T''; if it is, say, $\{B, C, D\}$, then both T' and T'' are nonempty, so both S' and S'' are empty, and then the only list not containing C is $\{B, D\}$. In this last case, check that the elements with list $\{B, D\}$ form the complement of a bipartite graph, and put all other elements in C.

We also have the following technique which generalizes two polynomial cases, sparse-dense and clique separators, but here the bound is quasi-polynomial:

If $AC = 0$ and $BD = 2$, then we get $2^{\log^2 n}$ instances with either no list containing $\{A, B\}$, or no list containing $\{C, D\}$. In the special case $A = 0$ and $B = 2$, the number of instances can be reduced to $2^{\binom{1+\log(n+1)}{2}}$ ("the restricted lists problem").

References

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

[2] A. Brandstadt. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics* 152 (1996): 47–54.

[3] A. Brandstadt. Corrigendum. *Discrete Mathematics* 186 (1998): 295.

[4] A. Brandstadt. Partitions of graphs into one or two independent sets and cliques. Informatik-Berichte Nr. 105, 1/1991, FernUniversitat Hagen 1991 Technical Report.

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

[6] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. Sophie Tison, ed., *CAAP'94: International Colloquium, Lectures Notes in Computer Science*, Edinburgh, UK (1994), pp. 68–82.

[7] H. Everett, S. Klein, and B. Reed. An algorithm for finding clique-cross partitions. Universite de Quebec a Montreal Technical Report 208, 1993.

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

[9] T. Feder, P. Hell, and J. Huang. List homomorphisms to bipartite graphs. Manuscript.

[10] T. Feder, P. Hell, and J. Huang. List homomorphisms to general graphs. Manuscript.

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

[12] C.M.H. de Figueiredo and S. Klein. The NP-completeness of multipartite cutset testing, *Congressus Numerantium* 119 (1996): 217–222.

[13] M.C. Golumbic. **Algorithmic Graph Theory and Perfect Graphs**. Academic Press, New York, 1980.

[14] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B* 48 (1990): 92–110.

[15] C. Hoang. Personal communication.

[16] M.R. Garey and D.S. Johnson. **Computers and Intractability**. W.H. Freeman and Company, San Francisco, 1979.

[17] L. Lovász. Communication complexity: a survey. In: **Paths, flows, and VLSI-Layout**, (B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds.), Springer, pp. 235–265.

[18] G. MacGillivray and M.L. Yu. Generalized partitions of graphs, Manuscript, 1997.

[19] R.M. McConnel and J.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 536–545.

[20] R.E. Tarjan. Decomposition by clique separators. *Discrete Mathematics* 55 (1985): 221–232.

[21] A. Tucker. Coloring graphs with stable sets. *Journal of Combinatorial Theory, Series B* 34 (1983): 258–267.

[22] N. Vikas. Computational complexity of graph compaction. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1999 (to appear).

[23] S. Whitesides. An algorithm for finding clique cutsets. *Information Processing Letters* 12 (1981): 31–32.

[24] S. Whitesides. A method for solving certain graph recognition and optimization problems, with applications to perfect graphs. *Annals of Discrete Mathematics* 21 (1984): 281–297.