2007

The Complexity of the List Partition Problem for Graphs

Kathie Cameron
Wilfrid Laurier University, kcameron@wlu.ca

Elaine M. Eschen
West Virginia University

Chính T. Hoàng
Wilfrid Laurier University, choang@wlu.ca

R. Sritharan
University of Dayton

SIAM J. DISCRETE MATH.
Vol. 21, No. 4, pp. 900–929

THE COMPLEXITY OF THE LIST PARTITION PROBLEM FOR GRAPHS*

KATHIE CAMERON†, ELAINE M. ESCHEN‡, CHÍNH T. HOÀNG§, AND R. SRITHARAN¶

Abstract. The k-partition problem is as follows: Given a graph G and a positive integer k, partition the vertices of G into at most k parts $A_1, A_2, \ldots, A_k$, where it may be specified that A_i induces a stable set, a clique, or an arbitrary subgraph, and pairs A_i, A_j $(i \neq j)$ be completely nonadjacent, completely adjacent, or arbitrarily adjacent. The list k-partition problem generalizes the k-partition problem by specifying for each vertex x, a list $L(x)$ of parts in which it is allowed to be placed. Many well-known graph problems can be formulated as list k-partition problems: e.g., 3-colorability, clique cutset, stable cutset, homogeneous set, skew partition, and 2-clique cutset. We classify, with the exception of two polynomially equivalent problems, each list 4-partition problem as either solvable in polynomial time or NP-complete. In doing so, we provide polynomial-time algorithms for many problems whose polynomial-time solvability was open, including the list 2-clique cutset problem. This also allows us to classify each list generalized 2-clique cutset problem and list generalized skew partition problem as solvable in polynomial time or NP-complete.

Key words. graph partition, list partition, complexity, algorithm

AMS subject classifications. 05C85, 05C69, 68R10

DOI. 10.1137/060666238

1. Introduction.

The problem of partitioning the vertex-set of a graph subject to a given set of constraints on adjacencies between vertices in two distinct parts, or among vertices within a part, is fundamental and ubiquitous in algorithmic graph theory. For example, the problem of testing whether graph G is bipartite is equivalent to testing whether the vertex-set of G can be partitioned into parts A_1 and A_2 such that each A_i is a stable set; here we have no constraint on the adjacencies between vertices in A_1 and vertices in A_2. A graph is a *split graph* [28] if its vertex-set can be partitioned into a clique and a stable set. As the definition itself suggests, testing whether graph G is a split graph is another partition problem where we do not restrict the adjacencies between vertices placed in different parts of the partition. On the other hand, testing whether graph G is a complete tripartite graph is equivalent to testing whether the vertex-set of G can be partitioned into parts A_1, A_2, and A_3 such that each A_i induces a stable set, and between vertices in parts A_i, A_j, $i \neq j$, we have all possible edges; hence, the relationship between vertices placed in distinct parts is relevant here.

*Received by the editors July 27, 2006; accepted for publication (in revised form) January 5, 2007; published electronically December 7, 2007. A preliminary version of this paper appeared in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.
http://www.siam.org/journals/sidma/21-4/66623.html

†Corresponding author. Department of Mathematics, Wilfrid Laurier University, Waterloo, Canada (kcameron@wlu.ca). This author acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

‡Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506 (Elaine.Eschen@wvu.edu). This author acknowledges support from NSERC of Canada.

§Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, Canada (choang@wlu.ca). This author acknowledges support from NSERC of Canada.

¶Computer Science Department, The University of Dayton, Dayton, OH 45469 (srithara@notes.udayton.edu). This author acknowledges support from the Research Council of The University of Dayton and NSERC of Canada.

1.1. The problem. In general, we can ask whether the vertex-set of a graph can be partitioned into at most k parts, $A_1, A_2, \ldots, A_k$, subject to constraints that require "no edges," "all edges," or "no restriction" between vertices placed in parts A_i and A_j; when $i = j$, the resulting constraint is on the subgraph induced by A_i. We can specify the required constraints on the partition via a symmetric $k \times k$ matrix M over $\{0, 1, *\}$. The natural interpretation is as follows: for $i \neq j$, if $M_{i,j} = 0$ (resp., 1, *), then we require "no edges" (resp., "all edges", "no restriction") between vertices placed in part A_i and vertices placed in part A_j; if $M_{i,i} = 0$ (resp., 1, *), then we require A_i to be a stable set (resp., clique, arbitrary subgraph). An *M-partition* of graph G then is a partition of the vertex-set of G into at most k parts so that all the constraints specified by M are respected. The *M-partition problem* asks the following: "Given G and a symmetric $k \times k$ matrix M over $\{0, 1, *\}$, does G admit an M-partition?". Many well-known graph theoretic problems are specific instances of the M-partition problem. For example, the 3-colorability problem is an M-partition problem where M is a 3×3 matrix with zeros on the main diagonal and asterisks everywhere else. Testing whether a graph is a split graph is asking whether the graph has an M-partition where M is a 2×2 matrix with a zero and one on the diagonal and asterisks everywhere else.

Feder et al. [22] introduced the M-partition problem and also generalized it to the *list M-partition problem*. In the list M-partition problem, in addition to being given graph G and a symmetric $k \times k$ matrix M over $\{0, 1, *\}$, for each vertex v of G, we are given a list $\mathcal{L}(v)$ that is a nonempty subset of $\{A_1, A_2, \ldots, A_k\}$. The problem asks the following: "Does G admit an M-partition in which each vertex v of G is assigned to a part in $\mathcal{L}(v)$?".

Many well-known graph problems can be formulated as list M-partition problems: e.g., list k-coloring, clique cutset, stable cutset, homogeneous set, skew partition, and 2-clique cutset. We study the list M-partition problems when M has dimension 4 with the goal of classifying them according to their complexity. Figure 1.1 illustrates the matrices corresponding to some of the problems we discuss.

1.2. Main results. In the following discussion, we use A, B, C, D to denote the parts of the M-partition problem. Let the *stubborn problem* be the list M-partition problem where $M_{A,A} = 0$, $M_{B,B} = 0$, $M_{D,D} = 1$, $M_{A,C} = M_{C,A} = 0$, and all other entries are asterisks (see Figure 1.1). The complement problem is obtained by interchanging the zeros and ones in the matrix. When M has dimension 4, we classify, with the sole exception of the stubborn problem and its complement, each list M-partition problem as either solvable in polynomial time or NP-complete. In doing so, we provide polynomial-time algorithms for many problems whose polynomial-time solvability was open. For example, we settle the open problem posed by Feder et al. [22] as to the existence of a polynomial-time algorithm to find a 2-clique cutset in a graph by providing a polynomial-time algorithm for the list 2-clique cutset problem. A 2-*clique cutset* is a cutset that induces the union of two cliques (or, equivalently, induces a bipartite graph in the complement).

Suppose $\mathcal{P}$ is an M-partition problem. A *generalized $\mathcal{P}$ problem* is an M'-partition problem where M' is obtained from M by changing some asterisks to either 0 or 1. Among other results, we prove that *any* list generalized 2-clique cutset problem (i.e., $M'_{A,A} = 1$, $M'_{B,B} = 1$, $M'_{C,D} = M'_{D,C} = 0$, and the other entries are 0, 1, or $*$) is solvable in polynomial time, unless it contains the complement of the 3-colorability problem, in which case it is NP-complete. This implies that the list strict 2-clique cutset problem is polynomial-time solvable, and via this we provide

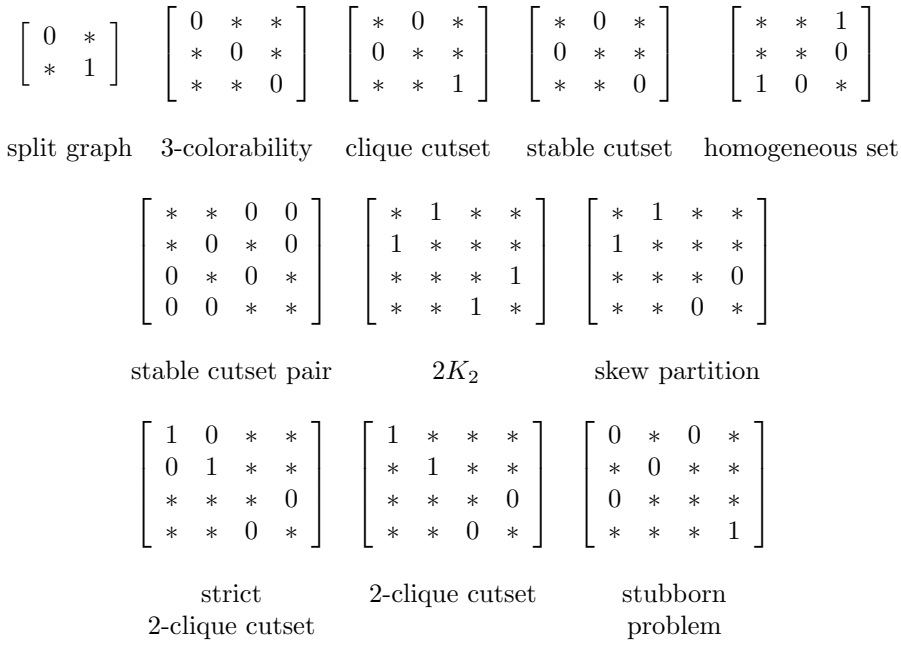

FIG. 1.1. *Some M-partition problems.*

a polynomial-time algorithm to find a strict 2-clique cutset. A *strict 2-clique cutset* is a cutset that induces the disjoint union of two cliques (or, equivalently, induces a complete bipartite graph in the complement). We also classify each list generalized skew partition problem as solvable in polynomial time or NP-complete.

1.3. Significance. Many important graph decomposition problems can be formulated as M-partition problems with additional constraints imposed on the parts. Indeed, the eventual resolution of the Strong Perfect Graph Conjecture by Chudnovsky et al. [5] relies in part on three types of decompositions (a type of skew cutset partition and two generalizations of the homogeneous set partition) that can be formulated as M-partition problems with constraints. Such extra constraints typically are that certain parts be nonempty, have at least a given number of vertices, induce subgraphs that have at least one edge, etc. As discussed later, an instance of the M-partition problem with additional constraints can be reduced to a set of instances of the list M-partition problem. Thus, the list M-partition problem provides a flexible model to capture extra constraints placed on the required partition.

Every list M-partition problem with M of dimension 4 was classified by Feder et al. [22] as either 'solvable in quasi-polynomial time' or NP-complete. Here, quasi-polynomial time is complexity of $O(n^{c \log^t n})$, where t and c are positive constants and n is the number of vertices in the input graph. Complete classification into polynomial-time solvable and NP-complete problems has been obtained for the list M-partition problem under several restrictions on M: when M is a matrix over $\{0, *\}, \{1, *\}$, or $\{0, 1\}$ [16, 19, 20, 22], has dimension 4 and does not contain an asterisk on the main diagonal [22], is the matrix for skew partition [15], has dimension 3 [22], and, trivially, when M has dimension 2. We complete this dichotomy classification (polynomial-

time solvable and NP-complete) for all problems when M has dimension 4, with the exception of the stubborn problem (see Figure 1.1) and its complement. Further, when M has dimension 4, we give polynomial-time algorithms for many list M-partition problems that were previously not known to be solvable in polynomial time [22]. The techniques we employ, obtained by strengthening the techniques used in [15], are general enough that they may prove useful in solving other decomposition problems. For instance, we develop tools that are applicable to list M-partition problems of any dimension.

In general, such dichotomy (into polynomial-time solvable and NP-complete problems) results are uncommon. However, Feder and Vardi [26] have made a dichotomy conjecture in the context of constraint-satisfaction problems which has generated considerable interest and has been proven in several special cases [17]. It is noted in [17, 22] that general list M-partition problems are similar to, but not exactly the same as, list constraint-satisfaction problems. It was conjectured in [22] that every list M-partition problem (with no restriction on dimension of M) is either solvable in quasi-polynomial time or NP-complete. This "quasi-dichotomy" has since been established by Feder and Hell [17].

We show that all the quasi-polynomial-time cases of the Feder et al. [22] quasi-dichotomy result for the list M-partition problem when M has dimension 4 are actually polynomial-time solvable, with the single exception of the stubborn problem (and its complement), for which the best known complexity remains quasi-polynomial time. There is no NP-complete problem that is known to have a quasi-polynomial-time solution, and it is generally believed that problems solvable in quasi-polynomial time are unlikely to be NP-complete. A polynomial-time solution for the stubborn problem, if one exists, appears to be difficult and to require methods different from those presented here and those in [17, 22].

Next, we remark on the attention that the stubborn problem has received subsequent to the appearance of a preliminary version of this paper in [4]. Feder and Hell have independently identified the so-called "edge-free three-coloring problem" (see [17]), in their attempt to classify certain list partition and list constraint satisfaction problems, whose complexity has also eluded classification. Further, it is shown in [17] that the two problems are closely related and also that the latter problem is at least as hard as the stubborn problem. Finally, in a recent work in [24], it was shown that each of these two problems can be solved in $O(n^{O(\frac{\log n}{\log\log n})})$ time, thus improving the bound of $O(n^{O(\log n)})$ established in [22]. This remains the current best complexity for solving the stubborn problem.

1.4. Background and previous work. Feder et al. [22] introduced the M-partition problem and, motivated by the need to capture additional restrictions on the contents of individual parts or the connections between parts, generalized it to the list M-partition problem. Lists also facilitate solving problems by recursing to subproblems with modified lists. We use this technique, which was also employed in the algorithms of [15, 22]. The list M-partition problem generalizes the M-partition, list k-coloring, and list homomorphism (cf. below) problems. An instance of the M-partition problem with certain additional constraints (that certain parts be nonempty, have at least a given number of vertices, induce subgraphs that have at least one edge, etc.) can be reduced to a set of instances of the list M-partition problem. In this manner, the list M-partition problem provides a flexible model to capture extra constraints on the required partition. Many well-known graph theoretic problems correspond to M-partitions with additional constraints. We elaborate on this notion next.

A *clique cutset* in a graph is a cutset that induces a clique. It is easy to see that a connected graph has a clique cutset if and only if its vertex-set can be partitioned into parts A, B, and C, such that C is a clique, there are no edges between parts A and B, and, *further*, each part is nonempty. For a graph G on n vertices, the clique cutset problem can be reduced to $O(n^3)$ instances of the list M-partition problem, where M is the matrix corresponding to the clique cutset problem, as follows: in order to handle the restriction that each of the parts A, B, and C be nonempty, for each triple x, y, z of vertices, we construct an instance with $\mathcal{L}(x) = \{A\}$, $\mathcal{L}(y) = \{B\}$, $\mathcal{L}(z) = \{C\}$, and the list for any other vertex is $\{A, B, C\}$. G has a clique cutset if and only if some such instance has a valid list M-partition. We note that finding a clique cutset and decomposing a graph via clique cutsets have applications in algorithmic graph theory [7, 28], and efficient algorithms exist for these problems [28, 33, 35, 36].

A 2-*clique cutset* is a cutset that is the union of two cliques (equivalently, the set of vertices in the cutset induces a bipartite graph in the complement). As illustrated in Figure 1.1, if parts A and B correspond to the two cliques whose union disconnects part C from part D, then whether a graph admits a 2-clique cutset is again an instance of the M-partition problem with the extra stipulation that each part be nonempty. Hayward and Reed [29] conjectured that every (even hole)-free graph (a graph that does not contain any induced cycle on an even number of vertices ≥ 4) that is not a complete graph contains a vertex whose neighborhood can be partitioned into two cliques. This conjecture implies that an (even hole)-free graph G has chromatic number at most $2\omega(G)$, where $\omega(G)$ is the clique number of G. Hoàng [31] proposed the weaker conjecture that (even hole)-free graphs different from a clique have a 2-clique cutset. Feder et al. [22] provided the first subexponential-time (but, not polynomial-time) algorithm to solve the list M-partition problem where M is the matrix for a 2-clique cutset, and hence, they also solved the 2-clique cutset problem in subexponential time. They posed the question [22] of the existence of a polynomial-time algorithm for the problem, which is answered in the affirmative here. We note that (even hole)-free graphs can be recognized in polynomial time [8, 9].

Analogous to a clique cutset, if we require the cutset to induce a stable set, then we get the stable cutset problem. A *skew partition* of a graph is a partition of its vertex-set into nonempty parts A, B, C, and D such that there are all possible edges between parts A and B and there are no edges between parts C and D. These problems are M-partition problems with the added constraint that each part be nonempty. Both the stable cutset and skew partition problems play prominent roles in the area of perfect graph theory. The interest in the stable cutset problem was motivated by Tucker's result [34] that a minimal imperfect graph, other than a chordless odd cycle, cannot contain a stable cutset. Chvátal conjectured [6] that a minimal imperfect graph does not admit a skew partition. Skew partitions played an important role in the proof of the Strong Perfect Graph Conjecture by Chudnovsky et al. [5]; this work also proved Chvátal's conjecture. Testing whether a graph has a stable cutset is known to be NP-complete [14]. However, Feder et al. [22] gave the first subexponential-time algorithm for the (list) skew partition problem. A polynomial-time algorithm for the (list) skew partition problem was developed subsequently by de Figueiredo et al. [15].

In certain other M-partition problems, there are constraints that there be at least a certain number of vertices in some parts. A *homogeneous set* or *module* in a graph is a set C of vertices such that C has at least two, but not all, of the vertices of the graph, and every vertex not in C is either adjacent to all the vertices in C, or none of the vertices in C. Among vertices not in C, if A is the set of vertices that are

adjacent to all the vertices in C, and B is the set of vertices that are adjacent to none of the vertices in C, then testing for the presence of module is an M-partition problem with the additional requirements that $|C| \geq 2$ and $A \cup B$ is nonempty. We can reduce the homogeneous set problem for a graph G on n vertices to $O(n^3)$ instances of the list M-partition problem, where M is the matrix corresponding to the homogeneous set problem, as follows: for each triple x, y, z of vertices, we set $\mathcal{L}(x) = \{C\}$, $\mathcal{L}(y) = \{C\}$, and $\mathcal{L}(z) = \{A, B\}$, the list of any other vertex to $\{A, B, C\}$, and check if any such instance has a valid list M-partition. Testing for the presence of modules and decomposition of a graph via modules have important applications in algorithmic graph theory, and efficient algorithms exist for these problems [10, 28, 32].

Feder et al. [22] studied the list M-partition problem with the goal of classifying matrices M into those for which the problem is efficiently solvable and those for which an efficient solution is perhaps unlikely. Next, we present results known on restricted versions of the list M-partition problem and then results known on the general list M-partition problem.

A k-coloring of graph G is the same as an M-partition of G where M (with dimension k) has zeros along the main diagonal and all other entries are asterisks. Therefore, the k-colorability problem is an M-partition problem where M is obtained from the 0-1 adjacency matrix of a complete (loopless) graph on k vertices by replacing every 1 with an asterisk. The more general H-coloring problem [30] is derived when M is obtained from the adjacency matrix of an arbitrary graph in the same way. More precisely, in the *H-coloring problem* [30], also called the *homomorphism problem*, given graph G and a specific graph H (possibly containing loops), we are asked whether it is possible to partition $V(G)$ into parts $A_u, u \in V(H)$, such that A_u is a stable set when u does not have a loop in H, and there are no edges between parts A_x and A_y whenever $xy \notin E(H)$. The H-coloring problem is solvable in polynomial time when H is bipartite or when H contains a loop, and is NP-complete otherwise [30].

The list H-coloring problem [16, 19, 20] is the list version of the H-coloring problem where, in addition to being given G and H, for each vertex v of G we are given a list, $\mathcal{L}(v)$ which is a subset of $V(H)$. The problem then asks whether there is an H-coloring subject to the additional restriction that each vertex v of G is placed in a part A_y such that $y \in \mathcal{L}(v)$. Just as the list coloring is a special case of list H-coloring (when H is a complete graph with no loops), list H-coloring is a special case of list M-partition where the matrix M is obtained from the adjacency matrix of the graph H by replacing every 1 with an asterisk.

In a sequence of papers [16, 19, 20], it was established that every list H-coloring problem (namely, every list M-partition problem where M is a matrix over $\{0, *\}$) is either solvable in polynomial time or NP-complete. The complement $\overline{M}$ of a matrix M over $\{0, 1, *\}$ is obtained from M by interchanging the zeros and ones and leaving the asterisks unchanged. Since the list M-partition problem for G, where M is a matrix over $\{1, *\}$, is essentially the same as the list $\overline{M}$-partition problem for the complement of G, it follows that every list M-partition problem, where M is a matrix over $\{1, *\}$, is also either solvable in polynomial time or NP-complete. See Figure 1.1 for definitions of the problems in the following theorems.

THEOREM 1.1 (see [16, 19, 20]). *If M is a matrix over $\{0, *\}$ or $\{1, *\}$, then the list M-partition problem is either solvable in polynomial time or NP-complete.*

The following corollary can be derived from [16, 19, 20].

COROLLARY 1.2 (see [16, 19, 20, 21]). *If M is a matrix over $\{0, *\}$ or $\{1, *\}$ and has dimension 4, then the list M-partition problem is solvable in polynomial time,*

except when M contains the matrix for 3-coloring, stable cutset, or their complements, or M is the matrix for stable cutset pair, $2K_2$, or their complements, in which cases the problem is NP-complete.

Feder et al. [22] proved the following three theorems.

THEOREM 1.3 (see [22]). *When M has dimension 3, the list M-partition problem is solvable in polynomial time, except when M is the matrix for 3-coloring, stable cutset, or their complements, in which cases the problem is NP-complete.*

THEOREM 1.4 (see [22]). *When M has dimension 4 and does not contain a $*$ on the main diagonal, the list M-partition problem is solvable in polynomial time, except when M contains the matrix for 3-coloring, or its complement, in which cases the problem is NP-complete.*

THEOREM 1.5 (see [22]). *When M has dimension 4, the list M-partition problem is solvable in quasi-polynomial time or NP-complete.*

Feder et al. [22] also showed that if M is a matrix over $\{0,1\}$, then the list M-partition problem is polynomial-time solvable. When M has dimension 2, the problem can be reduced to the 2-satisfiability problem and solved in polynomial time using the algorithm of [1].

It was conjectured in [22] that every list M-partition problem (with no restriction on dimension of M) is either solvable in quasi-polynomial time or NP-complete, and this now has been shown to be the case by Feder and Hell [17]. In a recent work [17], it has been shown that every list M-partition problem for directed graphs is either solvable in quasi-polynomial time or NP-complete. Further, when M has dimension at most 3, the quasi-polynomial cases of the list M-partition problem for directed graphs are now known to be polynomial-time solvable [25].

We close this section by referring the reader to [22] for a fine exposition on other graph theoretic problems that can be modeled as list M-partition problems.

2. Tools. We borrow some tools from [15] and [22]. For a vertex v of graph G, $N(v)$ denotes the set of vertices adjacent to v in G, i.e., $N(v)$ is the set of *neighbors of* v in G.

A basic strategy that we employ, much akin to [22] and [15], is replacing an instance $\mathcal{I}$ of the list M-partition problem on graph G by a polynomially bounded number of instances $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$ such that

- The answer to $\mathcal{I}$ is "yes" if and only if the answer to *some* $\mathcal{I}_k$ is "yes."

Moreover, each instance $\mathcal{I}_k$ satisfies *at least one* of the following:

- The longest list of $\mathcal{I}$ is missing in $\mathcal{I}_k$.
- The number of distinct lists in $\mathcal{I}_k$ is fewer than the number of distinct lists in $\mathcal{I}$.
- $\mathcal{I}_k$ is an instance of the list M'-partition problem for graph H where H is an induced subgraph of G and M' is a principal submatrix of M.
- $\mathcal{I}_k$ is easy to resolve.

Next we reproduce and summarize the tools from [22] that we use in this regard.

TOOL 1. *An instance of the list M-partition problem in which the list for every vertex of the input graph has size at most two, is solvable in polynomial time.*

Justification. Such a problem can easily be modeled as an instance of the 2-satisfiability problem (2-SAT) and solved using the algorithm in [1]. □

In the course of dealing with an instance of the list M-partition problem, our methods might decide to place a particular vertex in a specific part of the partition (either because the list of the vertex has size one, or this is one of the many possibilities

that will be tried). The following tool addresses how the instance can then be "cleaned up" to account for the placement of the vertex without altering the outcome.

TOOL 2. *Suppose we have an instance of the list M-partition problem on graph G with lists L, and suppose we decide to place vertex v in part X. Let L' be the lists obtained from L as follows: for all parts Y such that $M_{X,Y} = 0$, remove Y from the lists of neighbors of v. For all parts Y such that $M_{X,Y} = 1$, remove Y from the lists of nonneighbors of v. Then there is a list M-partition of G with respect to lists L and with v in X if and only if there is a list M-partition of G-v with respect to lists L'.*

TOOL 3. *Suppose we have an instance of the list M-partition problem for a graph on n vertices where $M_{X,Y} = 0$ and $M_{X,Z} = 1$. Then we can replace the instance with a set of instances consisting of one instance in which no vertex has X in its list, and at most n other instances in each of which no vertex has both Y and Z in its list such that the original instance admits a list M-partition if and only if some new instance does.*

Justification. If the original instance were to admit a list M-partition, then the possibilities are that either some vertex that had X in its list is placed in part X, or no vertex that had X in its list is placed in part X. The latter case can be covered by creating an instance by deleting X from every list. The former case can be covered by creating, for each vertex v that has X in its list, an instance by placing v in X and then applying Tool 2. □

Following the terminology used in [22], we say part X dominates part Y in matrix M, if for every part Z (including X and Y), we have $M_{X,Z} = M_{Y,Z}$ or $M_{X,Z} = *$.

TOOL 4. *Suppose we have an instance of the list M-partition problem on graph G with lists L, and part X dominates part Y in M. Let L' be the lists obtained from L by removing Y from any list that also contains X. Then there is a list M-partition of G with respect to lists L if and only if there is a list M-partition of G with respect to lists L'.*

Justification. If part X dominates part Y in matrix M, then in any list M-partition of G, a vertex in part Y can also be placed in part X. □

Again, following the terminology in [22], we say that a $k \times k$ matrix M *contains* a $p \times p$ matrix M', $p \leq k$, if M' is a principal submatrix of M.

TOOL 5. *If M contains M' and the list M'-partition problem is NP-complete, then the list M-partition problem is also NP-complete.*

Justification. Clearly, any polynomial-time algorithm for the list M-partition problem can be used, without any changes, to solve the list M'-partition problem in polynomial time. □

Recall that the complement $\overline{M}$ of matrix M is obtained from M by replacing every 0 with a 1, every 1 with a 0, and leaving the asterisks unchanged.

TOOL 6. *Graph G admits a list M-partition with respect to lists L if and only if the complement of G admits a list $\overline{M}$-partition with respect to the lists L.*

The following lemmata can be extracted from the details in [15]; however, they are not explicitly presented as lemmata there. We state them explicitly and present their proofs in their entirety for the sake of completeness. For simplicity of exposition (as was done in [15]) we use the constant 1/10 (and the related constants 7/10, 8/10, and 9/10) in the following lemmata. However, this can be replaced by any constant $1/c$ (and the related constants replaced by $(c-3)/c$, etc.) such that $c \geq 5$.

With respect to graph G and vertex-subset O of G, $\overline{O}$ denotes the subgraph induced by O in $\overline{G}$, the complement of G.

LEMMA 2.1 (see [15]). *Let G be a graph on n vertices and W be the set of those*

vertices of G whose degree is more than $\frac{9n}{10}$. If $|W| > \frac{9n}{10}$, then there is a linear time algorithm that

- *either finds pairwise disjoint vertex subsets O, T, and NT of G such that $|O| + |NT| \geq \frac{n}{10}$, $|T| \geq \frac{n}{10}$, $\overline{O}$ is connected, there are all possible edges between O and T, and each vertex in NT is nonadjacent to a vertex of O,*
- *or finds disjoint vertex subsets O^*, T^* of G such that $|O^*| \geq \frac{n}{10}$, $|T^*| \geq \frac{7n}{10}$, and there are all possible edges between O^* and T^*.*

Proof. Consider the following algorithm that partitions a subset W' of W into sets O, T, and NT, where $|W'| > \frac{8n}{10}$. The algorithm starts with a single vertex in set O and attempts to grow the set.

Algorithm α.
Input:
$W' \subseteq W$ such that $|W'| > \frac{8n}{10}$.

pick vertex $u \in W'$;
$O = \{u\}$;
$T = N(u) \cap W'$;
$NT = W' - T - \{u\}$;
repeat
    pick $v \in NT$;
    move v from NT to O;
    move $T \backslash N(v)$ from T to NT
until $(|O| + |NT| \geq \frac{n}{10})$ **or** $(NT = \emptyset)$

We first set $W' = W$ and invoke Algorithm α. As u is nonadjacent to fewer than $\frac{n}{10}$ vertices of G (hence, of W'), initially $|NT| < \frac{n}{10}$. Suppose the algorithm stops with $|O| + |NT| \geq \frac{n}{10}$. As v is nonadjacent to fewer than $\frac{n}{10}$ vertices of G (and hence, of W'), fewer than $\frac{n}{10}$ new vertices were moved into $O \cup NT$ during the final iteration. Therefore, $\frac{n}{10} \leq |O| + |NT| < \frac{2n}{10}$ and $|T| \geq (|W'| - \frac{2n}{10}) \geq (\frac{8n}{10} - \frac{2n}{10}) \geq \frac{n}{10}$. Further, as any vertex v moved into O is nonadjacent to some vertex of O, $\overline{O}$ remains connected. Clearly, there are all possible edges between O and T and every vertex in NT is nonadjacent to some vertex in O. Therefore, the sets O, T, and NT meet the conditions of the lemma.

On the other hand, suppose the algorithm stops with $|O|+|NT| < \frac{n}{10}$ and $NT = \emptyset$; clearly, $|O| < \frac{n}{10}$ and W was partitioned into O and T, and there are all possible edges between O and $W \backslash O$. We then apply the following algorithm to find the desired sets.

Algorithm β
Input:
$O \subseteq W$ such that $|O| < \frac{n}{10}$ and there are all possible edges between O and $W \backslash O$.

$O^* = O$;
$W' = W \backslash O^*$;
repeat
    Apply Algorithm α to W' to partition it into sets O, T, and NT;
    if $(|O| + |NT| \geq \frac{n}{10})$ **then**
        stop /* O, T, and NT are as desired */

else
{
$O^* = O^* \cup O$;
$W' = W \backslash O^*$
}
until ($|O^*| \geq \frac{n}{10}$);
$T^* = W \backslash O^*$

Note that as Algorithm β begins, $|W'| > \frac{8n}{10}$; also, there are all possible edges between O^* and $W \backslash O^*$. If the algorithm stops with $|O| + |NT| \geq \frac{n}{10}$, then we have found appropriate sets O, T, and NT. Otherwise, $|O| < \frac{n}{10}$ and W' is partitioned into O and T. This implies that at the end of each iteration, there are all possible edges between O^* and $W \backslash O^*$. If $|O^*| < \frac{n}{10}$ (and hence, the loop does not terminate), then $|W'| > \frac{8n}{10}$ for the next iteration, satisfying the precondition for Algorithm α. Suppose Algorithm β stops with $|O^*| \geq \frac{n}{10}$; then, at the end of the penultimate iteration, $|O^*| < \frac{n}{10}$. Since the set O of vertices added to O^* during the final iteration has fewer than $\frac{n}{10}$ vertices, when the algorithm stops, $|O^*| < \frac{2n}{10}$. Taking $T^* = W \backslash O^*$ then guarantees that $|T^*| \geq (|W| - |O^*|) \geq (\frac{9n}{10} - \frac{2n}{10}) \geq \frac{7n}{10}$ and there are all possible edges between O^* and T^*. Finally, the algorithms can easily be implemented to run in linear time. □

LEMMA 2.2 (see [15]). *Let G be a graph on n vertices with a partition of its vertex set into sets S_1, S_2 with $|S_1| = n_1$ and $|S_2| = n_2$. Let X_1 be the set of those vertices in S_1 each of which has fewer than $\frac{n_2}{10}$ neighbors in S_2. If $|X_1| \geq \frac{n_1}{2}$, then there is a linear time algorithm that finds vertex subsets O, M, and NM of G such that*

1. *$O \subseteq X_1$,*
2. *S_2 is partitioned into M and NM,*
3. *there are no edges between O and M,*
4. *every $u \in NM$ has a neighbor $u' \in O$, and*
5. *either $\frac{2n_2}{5} \leq |M| \leq \frac{n_2}{2}$ and $|NM| \geq \frac{n_2}{2}$, or $|O| \geq \frac{n_1}{10}$ and $|M| > \frac{n_2}{2}$.*

Proof. We apply the following linear time algorithm to grow the set $O \subseteq X_1$ starting with a single vertex in O while partitioning S_2 into sets M and NM.

Algorithm γ
Input:
Sets S_1, S_2, and X_1 as specified in Lemma 2.2.

pick vertex $u \in X_1$;
$O = \{u\}$;
$NM = N(u) \cap S_2$;
$M = S_2 \backslash NM$;
repeat
pick $v \in (X_1 \backslash O)$;
move v to O;
move $N(v) \cap M$ from M to NM
until ($|M| \leq \frac{n_2}{2}$) **or** ($|O| \geq \frac{n_1}{10}$)

It is evident from Algorithm γ that there are no edges between O and M, and every vertex in NM has a neighbor in O. As u is adjacent to fewer than $\frac{n_2}{10}$ vertices

of S_2, initially $|M| > \frac{9n_2}{10}$. Suppose $|M| \leq \frac{n_2}{2}$ when the algorithm stops. Since v is adjacent to fewer than $\frac{n_2}{10}$ vertices of S_2 (hence, of M), fewer than $\frac{n_2}{10}$ vertices were moved from M to NM during the final iteration. Therefore, $|M| > (\frac{n_2}{2} - \frac{n_2}{10})$, and we have $\frac{2n_2}{5} \leq |M| \leq \frac{n_2}{2}$. As M and NM partition the set S_2, we also have $|NM| \geq \frac{n_2}{2}$, as desired. On the other hand, suppose the algorithm stops with $|M| > \frac{n_2}{2}$ and $|O| \geq \frac{n_1}{10}$. The conditions of the lemma are then trivially met. ∎

LEMMA 2.3 (see [15]). *Let G be a graph on n vertices with a partition of its vertex set into sets S_1, S_2 with $|S_1| = n_1$ and $|S_2| = n_2$. Let W_1 be the set of those vertices in S_1 each of which has more than $\frac{9n_1}{10}$ neighbors in S_1 and more than $\frac{9n_2}{10}$ neighbors in S_2. Let W_2 be the set of those vertices in S_2 each of which has more than $\frac{9n_2}{10}$ neighbors in S_2 and more than $\frac{9n_1}{10}$ neighbors in S_1. If $|W_1| > \frac{9n_1}{10}$ and $|W_2| > \frac{9n_2}{10}$, then there is a linear time algorithm that*

- *either finds pairwise disjoint vertex subsets O, T, and NT of G such that*
  1. *$\overline{O}$ is connected,*
  2. *there are all possible edges between O and T,*
  3. *each vertex in NT is nonadjacent to a vertex in O,*
  4. *$|T \cap S_1| \geq \frac{n_1}{10}$,*
  5. *$|T \cap S_2| \geq \frac{n_2}{10}$, and*
  6. *either $|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10}$, or $|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10}$,*
- *or finds disjoint vertex subsets O^*, T^* of G such that*
  1. *either $O^* \subseteq S_1$ and $|O^*| \geq \frac{n_1}{10}$, or $O^* \subseteq S_2$ and $|O^*| \geq \frac{n_2}{10}$,*
  2. *$|T^* \cap S_1| \geq \frac{n_1}{10}$,*
  3. *$|T^* \cap S_2| \geq \frac{n_2}{10}$, and*
  4. *there are all possible edges between O^* and T^*.*

Proof. We begin by noting that the proof of Lemma 2.3 is similar in principle to that of Lemma 2.1. Let $W = (W_1 \cup W_2)$, and therefore, $|W \cap S_1| > \frac{9n_1}{10}$ and $|W \cap S_2| > \frac{9n_2}{10}$.

Consider the following algorithm that partitions a subset W' of W into sets O, T, and NT, where $|W' \cap S_1| > \frac{8n_1}{10}$ and $|W' \cap S_2| > \frac{8n_2}{10}$. The algorithm starts with a single vertex in set O and attempts to grow the set.

Algorithm δ
Input:
$W' \subseteq W$ such that $|W' \cap S_1| > \frac{8n_1}{10}$ and $|W' \cap S_2| > \frac{8n_2}{10}$.

pick vertex $u \in W'$;
$O = \{u\}$;
$T = N(u) \cap W'$;
$NT = W' - T - \{u\}$;
repeat
  pick $v \in NT$;
  move v from NT to O;
  move $T \backslash N(v)$ from T to NT
until $(|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10})$ **or** $(|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10})$ **or** $(NT = \emptyset)$

We first set $W' = W$ and invoke Algorithm δ. As u is nonadjacent to fewer than $\frac{n_1}{10}$ vertices of S_1 (hence, of $W' \cap S_1$) and fewer than $\frac{n_2}{10}$ of vertices of S_2 (hence, of $W' \cap S_2$), initially $|NT \cap S_1| < \frac{n_1}{10}$ and $|NT \cap S_2| < \frac{n_2}{10}$.

Suppose when the algorithm stops, $((|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10})$ or $(|O \cap S_2| + |NT \cap$

$S_2| \geq \frac{n_2}{10}))$ is true; without loss of generality, assume that $|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10}$. As v is nonadjacent to fewer than $\frac{n_1}{10}$ vertices of S_1 (hence, of $W' \cap S_1$), fewer than $\frac{n_1}{10}$ new vertices were moved into $(O \cap S_1) \cup (NT \cap S_1)$ during the final iteration. Therefore, $|O \cap S_1| + |NT \cap S_1| < \frac{2n_1}{10}$. For similar reasons, $|O \cap S_2| + |NT \cap S_2| < \frac{2n_2}{10}$. Hence, $|T \cap S_1| \geq (|W' \cap S_1| - (|O \cap S_1| + |NT \cap S_1|)) \geq (\frac{8n_1}{10} - \frac{2n_1}{10}) \geq \frac{n_1}{10}$ and $|T \cap S_2| \geq (|W' \cap S_2| - (|O \cap S_2| + |NT \cap S_2|)) \geq (\frac{8n_2}{10} - \frac{2n_2}{10}) \geq \frac{n_2}{10}$. Further, as any vertex v moved into O is nonadjacent to some vertex of O, $\overline{O}$ remains connected. Clearly, there are all possible edges between O and T and every vertex in NT is nonadjacent to some vertex in O. Therefore, the sets O, T, and NT meet the conditions of the lemma.

On the other hand, suppose the algorithm stops with $|O \cap S_1| + |NT \cap S_1| < \frac{n_1}{10}$, $|O \cap S_2| + |NT \cap S_2| < \frac{n_2}{10}$, and $NT = \emptyset$; clearly, $|O \cap S_1| < \frac{n_1}{10}$, $|O \cap S_2| < \frac{n_2}{10}$, W is partitioned into O and T, and there are all possible edges between O and $W \backslash O$. We then apply the following algorithm to find the desired sets.

Algorithm ϵ
Input:
$O \subseteq W$ such that $|O \cap S_1| < \frac{n_1}{10}$ and $|O \cap S_2| < \frac{n_2}{10}$
and there are all possible edges between O and $W \backslash O$.

$J^* = O$;
$W' = W \backslash J^*$;
repeat
  Apply Algorithm δ to W' to partition it into sets O, T, and NT;
  if $(|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10})$ **or** $(|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10})$ **then**
    stop /* O, T, and NT are as desired */
  else
  {
    $J^* = J^* \cup O$;
    $W' = W \backslash J^*$
  }
until $(|J^* \cap S_1| \geq \frac{n_1}{10})$ **or** $(|J^* \cap S_2| \geq \frac{n_2}{10})$;
if $(|J^* \cap S_1| \geq \frac{n_1}{10})$ **then**
  $O^* = J^* \cap S_1$
else
  $O^* = J^* \cap S_2$;
$T^* = W \backslash J^*$

Note that as Algorithm ϵ begins, $|W' \cap S_1| > \frac{8n_1}{10}$ and $|W' \cap S_2| > \frac{8n_2}{10}$; also, there are all possible edges between J^* and $W \backslash J^*$. If the algorithm stops with $((|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10})$ or $(|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10}))$ being true, then we have found appropriate sets O, T, and NT. Otherwise, $|O \cap S_1| < \frac{n_1}{10}$, $|O \cap S_2| < \frac{n_2}{10}$, and W' is partitioned into O and T. This implies that at the end of each iteration, there are all possible edges between J^* and $W \backslash J^*$. If $|J^* \cap S_1| < \frac{n_1}{10}$ and $|J^* \cap S_2| < \frac{n_2}{10}$ (and hence, the loop does not terminate), then $|W' \cap S_1| > \frac{8n_1}{10}$ and $|W' \cap S_2| > \frac{8n_2}{10}$ for the next iteration, satisfying the precondition for Algorithm δ. Without loss of generality, suppose the loop in Algorithm ϵ terminates with $|J^* \cap S_1| \geq \frac{n_1}{10}$; then, at the end of the penultimate iteration, $|J^* \cap S_1| < \frac{n_1}{10}$ and $|J^* \cap S_2| < \frac{n_2}{10}$. Since the set O of vertices added to J^* during the final iteration has fewer than $\frac{n_i}{10}$ vertices of S_i, $i = 1, 2$,

$|J^* \cap S_i| < \frac{2n_i}{10}$ for $i = 1, 2$. Taking $T^* = W \backslash J^*$ and $O^* = (J^* \cap S_1)$ then guarantees that $|T^* \cap S_1| \geq (|W \cap S_1| - |J^* \cap S_1|) \geq \frac{n_1}{10}$, $|T^* \cap S_2| \geq (|W \cap S_2| - |J^* \cap S_2|) \geq \frac{n_2}{10}$, and there are all possible edges between O^* and T^*. □

3. Three procedures. We assume the input is a graph $G = (V, E)$ with the adjacency requirements on the parts A_i and a set Φ of lists $\mathcal{L}(v)$. We consider the instance Φ as a partition of V into at most $2^k - 1$ sets $S_{\mathcal{L}}$, indexed by the nonempty subsets $\mathcal{L}$ of $\mathcal{Z} = \{A_1, A_2, \ldots, A_k\}$. That is, $S_{\mathcal{L}}$ is the set of vertices with list $\mathcal{L}$. For example, if $\mathcal{L}(v) = \{A_1, A_2\}$, then $v \in S_{\{A_1,A_2\}}$. For simplicity we will drop the set brackets in the subscript, i.e., $S_{A_1A_2} = S_{\{A_1,A_2\}}$. $S_{\mathcal{L}}(\Phi)$ refers to the set $S_{\mathcal{L}}$ defined by Φ. When the context is clear, we write $S_{\mathcal{L}} = S_{\mathcal{L}}(\Phi)$. When we say Φ has a solution, it is assumed the parts are $A_1, A_2, \ldots, A_k$.

Throughout the algorithms used in the proof of the main theorem in section 4, Properties 1 and 2 below are always satisfied by the partition of V according to the sets $S_{\mathcal{L}}$.

Property 1. If the algorithm returns a partition and v is in $S_{\mathcal{L}}$, then the returned part A_i containing v is a part in $\mathcal{L}$.

Property 2. If $v \in S_{\mathcal{L}}$ for some $\mathcal{L}$, then for each $A_i \in \mathcal{L}$ and each S_{A_j}, v is adjacent (resp., nonadjacent) to all vertices in S_{A_j} whenever $M_{A_i,A_j} = 1$ (resp., $M_{A_i,A_j} = 0$). (It is possible that $i = j$.)

Often, we replace an instance Φ by a set of instances $\{\Phi_1, \Phi_2, \ldots, \Phi_p\}$ such that Φ has a solution if and only if some Φ_i has a solution. In this case, we say the set of instances $\{\Phi_1, \Phi_2, \ldots, \Phi_p\}$ is *equivalent* to Φ.

Let $X \subseteq S_{\mathcal{L}}$ and $A_i \in \mathcal{L}$. In creating a new instance Φ_j from Φ, we often say X *drops (part)* A_i. By this we mean for each vertex $v \in X$, $\mathcal{L}(v) = \mathcal{L} - \{A_i\}$, and, consequently, $S_{\mathcal{L}}(\Phi_j) = S_{\mathcal{L}}(\Phi) - X, S_{\mathcal{L}-\{A_i\}}(\Phi_j) = S_{\mathcal{L}-\{A_i\}}(\Phi) \cup X$ and $S_{\mathcal{L}'}(\Phi_j) = S_{\mathcal{L}'}(\Phi)$ for all other subsets $\mathcal{L}'$ of $\mathcal{Z}$. When we say X *gets the list* A_i we mean X drops all parts except A_i (i.e., $X \subseteq S_{A_i}(\Phi_j)$).

The reduction operation. Whenever a new instance Φ_j is created, a set $S_{A_i}(\Phi_j)$ may be a proper superset of $S_{A_i}(\Phi)$, and in any solution of Φ_j we must have $S_{A_i}(\Phi_j) \subseteq A_i$ for all i. If some $v \in S_{\mathcal{L}}(\Phi_j)$, where $A_i \in \mathcal{L}$, is not adjacent to all vertices in $S_{A_j}(\Phi_j)$ and $M_{A_i,A_j} = 1$, then v cannot be in part A_i in any solution. So we can *reduce* to a new problem where v drops the part A_i. In the case that $\mathcal{L}$ is a singleton set, Φ_j has no solution. The case where $M_{A_i,A_j} = 0$ is handled in a similar way. It is easy to see that after $O(n)$ similar reductions, we obtain an equivalent instance satisfying Property 2, or halt because Φ_j has no solution.

We refer to parts A_i, A_j such that $M_{A_i,A_j} = 1$ ($M_{A_i,A_j} = 0$) as *true partners* (*false partners*). We use *partner* without qualification to refer to a true or false partner. Note that a part can be its own partner.

The following two procedures (1 and 2) generalize two procedures in [15]. These generalizations are necessary for the proof of our main result in section 4. Also, these procedures are applicable to more general list partition problems than 4-part problems.

Remark. As in the lemmata of section 2, we assume $k \leq 10$ and use the corresponding constant 1/10 (and the related constants 7/10, 8/10, and 9/10) in the following procedures. However, for arbitrary dimension k, the constant 1/10 can be replaced by any constant $1/c$ (and the related constants replaced by $(c-3)/c$, etc.) such that $c \geq \max\{5, k\}$. Thus, these procedures are applicable to partition problems of any dimension k. The procedures are applied recursively to a given instance Φ to generate an equivalent set of instances (cf. Notes 1, 2, and 3). Taking $c = \max\{5, k\}$

minimizes the number of instances generated for any k. □

Procedure 1.

Input: An instance Φ of the list M-partition problem with set $\mathcal{Z}$ of parts $A_1, A_2, \ldots, A_k$, and a set $\mathcal{L} \subseteq \mathcal{Z}$ such that $S_{\mathcal{L}} \neq \emptyset$ and the parts $A_i \in \mathcal{L}$ can be put into sets $\mathcal{U}$ and $\mathcal{F}$ such that $\mathcal{U} \neq \emptyset, \mathcal{F} \neq \emptyset, \mathcal{U} \cup \mathcal{F} = \mathcal{L}$, but $\mathcal{U} \cap \mathcal{F}$ may or may not be empty, and the following properties hold:

(a) **Clique structure.** $\mathcal{U} = \{U_1, U_2, \ldots, U_u\}$. If $|\mathcal{U}| = 1$, then $M_{U_1,U_1} = 1$; otherwise $M_{U_iU_j} = 1$ for all i and j, $i \neq j$, except possibly when $i = u - 1$ and $j = u$. If $M_{U_{u-1},U_u} \neq 1$, then $M_{U_{u-1},U_{u-1}} = M_{U_u,U_u} = 1$.

(b) $\mathcal{F} = \{F_1, \ldots, F_f\}$. If $|\mathcal{F}| = 1$, then $M_{F_1,F_1} = 0$; otherwise $M_{F_i,F_j} = 0$ for all i, j, $i \neq j$, except possibly when $i = f - 1$ and $j = f$. If $M_{F_{f-1},F_f} \neq 0$, then $M_{F_{f-1},F_{f-1}} = M_{F_f,F_f} = 0$.

As noted above, lists satisfying property (a) are said to have the *clique structure*.

Output: A set of at most k instances, $\{\Phi_1, \Phi_2, \ldots\}$, that is equivalent to Φ, and such that for each i, $|S_{\mathcal{L}}(\Phi_i)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|$, or a proof that Φ has no solution.

Note 1. Given an instance Φ, applying Procedure 1 to Φ produces at most k instances Φ_i with $|S_{\mathcal{L}}(\Phi_i)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|$. Thus, given an instance Φ on a graph G with n vertices (with $k \leq 10$), recursively applying Procedure 1 produces a polynomial number of instances Φ' for which $S_{\mathcal{L}}(\Phi') = \emptyset$, and the set of instances produced is equivalent to Φ. It is easy to see that the number of instances Φ' is at most $k^{\log_{\frac{10}{9}} n} = n^{\log_{\frac{10}{9}} k}$. We shall refer to this process as *eliminating* the set $\mathcal{S}_{\mathcal{L}}$. □

Details of Procedure 1. Let $n = |S_{\mathcal{L}}(\Phi)|$. Any partner referred to here is a partner in $\mathcal{L}$.

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

To cover the possibility that v is placed in part A_i in the solution, we generate instances Φ_i, $i = 1, \ldots, k$, by setting $S_{A_i}(\Phi_i) = \{v\} \cup S_{A_i}(\Phi)$ and reducing so that Property 2 holds. If $A_i \in \mathcal{U}$, then the nonneighbors of v must drop the part $p(A_i)$ (hence, they cannot remain in $S_{\mathcal{L}}$) where $p(A_i)$ is the true partner of A_i. Since there are at least $\frac{n}{10}$ nonneighbors of v, $|S_{\mathcal{L}}(\Phi_i)| \leq \frac{9n}{10}$. Similarly, if $A_i \in \mathcal{F}$, then the neighbors of v must drop the part $p(A_i)$ where $p(A_i)$ is the false partner of A_i; hence, $|S_{\mathcal{L}}(\Phi_i)| \leq \frac{9n}{10}$. Clearly, the set of instances $\{\Phi_1, \ldots, \Phi_k\}$ is equivalent to Φ.

We may now assume that every vertex in $S_{\mathcal{L}}$ has more than $\frac{9n}{10}$ neighbors or fewer than $\frac{n}{10}$ neighbors in $S_{\mathcal{L}}$.

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

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

In any solution to Φ, $|A_i \cap S_{\mathcal{L}}(\Phi)| \geq \frac{n}{k} \geq \frac{n}{10}$ for some A_i; thus, we generate an instance for each A_i to cover the possibility that A_i is such a part. If $|A_i \cap S_{\mathcal{L}}(\Phi)| \geq \frac{n}{10}$ and A_i has a true (false) partner $p(A_i)$, then $p(A_i) \cap X = \emptyset$ ($p(A_i) \cap W = \emptyset$). Properties (a) and (b) ensure that each A_i has either a true or false partner $p(A_i)$. Thus, for $i = 1, \ldots, k$, generate Φ_i in which X drops $p(A_i)$, if $p(A_i)$ is a true partner; otherwise, generate Φ_i in which W drops $p(A_i)$. For each i, $|S_{\mathcal{L}}(\Phi_i)| \leq \frac{9n}{10}$, and the set of instances $\{\Phi_1, \ldots, \Phi_k\}$ is equivalent to Φ.

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

By Lemma 2.1, we can either

(i) find pairwise disjoint subsets O, T, NT of $S_{\mathcal{L}}$ such that $\overline{O}$ is connected, $|O| + |NT| \geq \frac{n}{10}, |T| \geq \frac{n}{10}$, there are all possible edges between O and T, and each vertex in NT is nonadjacent to some vertex in O, or

(ii) find disjoint subsets O^* and T^* of $S_{\mathcal{L}}$ such that $|O^*| \geq \frac{n}{10}, |T^*| \geq \frac{7n}{10}$, and there are all possible edges between O^* and T^*.

Case (i). We create an instance Φ_{A_i} for each part A_i of $\mathcal{L}$ as follows. First, for each A_i of $\mathcal{F}-\mathcal{U}$ with false partner $p(A_i)$, construct Φ_{A_i} by making T drop the part $p(A_i)$.

Now, we may assume that the remaining parts in $\mathcal{L}$ can be named $U_1, U_2, \ldots, U_l$ so that they have the clique structure. We create instances as follows:

1. If $l = 1$, then $M_{U_1,U_1} = 1$. If $|O| > 1$, do not create a new instance. Otherwise, create instance Φ_{U_1} by placing the only vertex of O in part U_1 and making NT drop part U_1.
2. If $l \geq 2$ and $M_{U_i,U_j} = 1$ for all $j \neq i$, create, for each i, Φ_{U_i} from Φ by making $O \cup NT$ drop every part U_j, $j \neq i$.
3. If $l \geq 2$ and $M_{U_iU_j} \neq 1$ for some i, j, then we must have $\{i, j\} = \{l-1, l\}$ and $M_{U_{l-1},U_{l-1}} = M_{U_l,U_l} = 1$. Test whether O has a unique partition into two cliques K_1, K_2. If not, do not create a new instance (see the explanation below). Otherwise, create two instances Φ_1, Φ_2 as follows. In Φ_1, K_1 gets the list U_{l-1} (it drops all other parts) and K_2 gets the list U_l; for each vertex x in NT, x drops part U_{l-1} if x is nonadjacent to some vertex in K_1, or x drops part U_l if x is nonadjacent to some vertex in K_2. The instance Φ_2 is defined similarly with K_1 getting list U_l and K_2 getting list U_{l-1}.

We now show that the set of new instances is equivalent to Φ. Suppose there is a solution $A_1, \ldots, A_k$ to Φ. It must be the case that for some i, $O \cap A_i \neq \emptyset$. If there is an A_i in $\mathcal{F}-\mathcal{U}$ with a false partner $p(A_i)$ such that $O \cap A_i \neq \emptyset$, then $T \cap p(A_i) = \emptyset$; this eventuality is covered by Φ_{A_i}.

Now suppose there is no part in $\mathcal{F}-\mathcal{U}$ that has nonempty intersection with O. Let the parts not in $\mathcal{F}-\mathcal{U}$ be $U_1, \ldots, U_l$ (if they exist). These parts must have the clique structure. If $l = 1$, then we have $M_{U_1,U_1} = 1$ and $O \subseteq U_1$ in the solution. Since $\overline{O}$ is connected, it follows that when $|O| > 1$, there is no solution. Otherwise, the only vertex in O must go to part U_1. As no vertex in NT can now be in part U_1, NT must drop the part U_1; this eventuality is covered by the instance Φ_{U_1}.

Now suppose $l \geq 2$. For any U_i that is a true partner of all U_j with j different from i, if $O \cap U_i \neq \emptyset$, then (as $\overline{O}$ is connected) $O \subseteq U_i$. Since no member of NT can now be placed in a part that is a true partner of U_i, it follows that NT must drop all parts U_j with $i \neq j$; this eventuality is covered by Φ_{U_i}.

Last, we consider the case $M_{U_{l-1},U_{l-1}} = M_{U_l,U_1} = 1$ and every vertex in O belongs to $U_{l-1} \cup U_l$. Since $\overline{O}$ is connected, O must be partitioned uniquely into two cliques K_1, K_2; otherwise, there is no solution. We see that every vertex in NT must drop a part (U_{l-1} or U_l); this eventuality is covered by Φ_1 and Φ_2.

Case (ii). We construct two new instances from Φ as follows. Choose an A_i that has a false partner $p(A_i)$ and create Φ_1 by making T^* drop $p(A_i)$; then create Φ_2 by making O^* drop A_i. This can be justified as follows. In any solution to Φ, if $A_i \cap O^* \neq \emptyset$, then $T^* \cap p(A_i) = \emptyset$; otherwise, $O^* \cap A_i = \emptyset$.

Case 4. $|X| > \frac{9n}{10}$.

This case is similar to Case 3 with G replaced by $\overline{G}$ and M replaced by $\overline{M}$.

It is easily verified that in each instance Γ created, $|S_{\mathcal{L}}(\Gamma)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|$. If no new instances are produced by the above analysis, then Φ has no solution. This completes the description of Procedure 1. ∎

Procedure 2.

Input: Instance Φ of the list M-partition problem with set $\mathcal{Z}$ of parts $A_1, \ldots, A_k$, and two sets $\mathcal{L}$ and $\mathcal{R}$, which are subsets of $\mathcal{Z}$, such that $S_{\mathcal{L}} \neq \emptyset, S_{\mathcal{R}} \neq \emptyset$, $\mathcal{L} \not\subseteq \mathcal{R}$, $\mathcal{R} \not\subseteq \mathcal{L}$, and we can write $\mathcal{L} = \{L_1, L_2, \ldots L_p\}$ ($p \geq 3$) and $\mathcal{R} = \{R_1, R_2, \ldots, R_q\}$ ($q \geq 3$) so

that

1. each L_i has a partner in $\mathcal{R}$,
2. each R_i has a partner in $\mathcal{L}$,
3. some L_i has a true partner in $\mathcal{R}$ (equivalently, some R_i has a true partner in $\mathcal{L}$),
4. some L_i has a false partner in $\mathcal{L} \cup \mathcal{R}$,
5. some R_j has a false partner in $\mathcal{L} \cup \mathcal{R}$,
6. for each i, if L_i has no false partner in $\mathcal{R}$, then L_i has a true partner in $\mathcal{L}$,
7. for each i, if R_i has no false partner in $\mathcal{L}$, then R_i has a true partner in $\mathcal{R}$,
8. if the set $\mathcal{F}$ of parts in $\mathcal{L}$ that have no true partners in $\mathcal{R}$ is not empty, then there is a part R_j that is a false partner of all parts in $\mathcal{F}$,
9. if the set $\mathcal{H}$ of parts in $\mathcal{R}$ that have no true partners in $\mathcal{L}$ is not empty, then there is a part L_i that is a false partner of all parts in $\mathcal{H}$,
10. if the set $\mathcal{U}$ of parts of $\mathcal{L} \cup \mathcal{R}$ that have no false partners in $\mathcal{L} \cup \mathcal{R}$ is not empty, then the parts in $\mathcal{U}$ must have the clique structure, each of them has a true partner in $\mathcal{L}$ and in $\mathcal{R}$, and the two parts in $\mathcal{U}$ that are not true partners (if they exist) must belong to $\mathcal{L} \cap \mathcal{R}$.

Output: A set of at most $2k$ instances $\{\Phi_1, \Phi_2, \ldots\}$ that is equivalent to Φ, and such that for each i, $|S_{\mathcal{L}}(\Phi_i)|\,|S_{\mathcal{R}}(\Phi_i)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|\,|S_{\mathcal{R}}(\Phi)|$, or a proof that Φ has no solution.

Note 2. Given an instance Φ on a graph G with n vertices (with $k \leq 10$) that satisfies the conditions of Procedure 2, recursively applying Procedure 2 produces a polynomial number of instances Φ' for which $S_{\mathcal{L}}(\Phi') = \emptyset$ or $S_{\mathcal{R}}(\Phi') = \emptyset$, and the set of instances produced is equivalent to Φ. It is easy to see that the number of instances Φ' is at most $(2k)^{\log_{\frac{10}{9}} n^2} = n^{2\log_{\frac{10}{9}} 2k}$. □

Details of Procedure 2. Write $S_1 = S_{\mathcal{L}}, S_2 = S_{\mathcal{R}}$. Let $n_1 = |S_1|$ and $n_2 = |S_2|$. For a vertex $v \in S_1 \cup S_2$, let $d_i(v) = |N(v) \cap S_i|$, $i = 1,2$.

Case 1. There is a vertex v in S_1 with $\frac{n_2}{10} \leq d_2(v) \leq \frac{9n_2}{10}$.

For each $L_i \in \mathcal{L}$, let $p(L_i)$ be a partner of L_i in $\mathcal{R}$. For each $L_i \in \mathcal{L}$, construct an instance Φ_i from Φ as follows. If L_i is a true partner of $p(L_i)$, then $S_2 - N(v)$ drops the part $p(L_i)$; otherwise, $S_2 \cap N(v)$ drops part $p(L_i)$. It is a routine matter to verify that the set of new instances is equivalent to Φ.

Case 1′. There is a vertex v in S_2 with $\frac{n_1}{10} \leq d_1(v) \leq \frac{9n_1}{10}$.

This case is symmetric to Case 1.

Case 2. Every vertex v in S_1 satisfies $d_2(v) < \frac{n_2}{10}$ or $d_2(v) > \frac{9n_2}{10}$. Every vertex v in S_2 satisfies $d_1(v) < \frac{n_1}{10}$ or $d_1(v) > \frac{9n_1}{10}$.

Define four sets as follows:

$$X_1 = \left\{ v \in S_1 | d_2(v) < \frac{n_2}{10} \right\},\ X_2 = \left\{ v \in S_2 | d_1(v) < \frac{n_1}{10} \right\},$$

$$W_1 = \left\{ v \in S_1 | d_2(v) > \frac{9n_2}{10} \right\},\ W_2 = \left\{ v \in S_2 | d_1(v) > \frac{9n_1}{10} \right\}.$$

There are three cases to consider.

Case 2.1. $|X_1|, |W_1| \geq \frac{n_1}{10}$.

Create q new instances from Φ as follows. For each $R_j \in \mathcal{R}$, let $p(R_j)$ be a partner of R_j in $\mathcal{L}$. If $p(R_j)$ is a true (resp., false) partner of R_j, then Φ_j is obtained from Φ by making X_1 (resp., W_1) drop the part $p(R_j)$. This is justified as follows. In any

solution to Φ some R_j must have $|R_j \cap S_2| \geq \frac{n_2}{q} \geq \frac{n_2}{k} \geq \frac{n_2}{10}$; if $M_{R_j,p(R_j)} = 1$ (resp., 0), then $X_1 \cap p(R_j) = \emptyset$ (resp., $W_1 \cap p(R_j) = \emptyset$). Thus, the q new instances cover all the eventualities.

Case 2.1′. $|X_2|, |W_2| \geq \frac{n_2}{10}$.

This case is symmetric to Case 2.1.

Case 2.2. $|X_1| > \frac{9n_1}{10}$.

Find the sets O, M, and NM as defined by Lemma 2.2.

Suppose first that $|O| \geq \frac{n_1}{10}$ and $|M| > \frac{n_2}{2}$. Replace Φ by two new instances Φ_1, Φ_2 as follows. Let L_i be a part with a true partner $p(L_i)$ in $\mathcal{R}$. Φ_1 is obtained from Φ by making M drop the part $p(L_i)$ and Φ_2 is obtained from Φ by making O drop the part L_i. This can be justified as follows. Consider any solution of Φ. If $O \cap L_i \neq \emptyset$, then no vertex of M can be in part $p(L_i)$; otherwise, no vertex of O is in part L_i. Thus, the two new instances Φ_1, Φ_2 cover all the eventualities.

Now, we may assume that $\frac{2n_2}{5} \leq |M| \leq \frac{n_2}{2}$ and $|NM| \geq \frac{n_2}{2}$. Let $\mathcal{L}^+$ be the set of parts in $\mathcal{L}$ that have a true partner $p(L_i)$ in $\mathcal{R}$. Construct at most $|\mathcal{L}^+| + 1$ new instances as follows. For each $L_i \in \mathcal{L}^+$, construct Φ_{L_i} from Φ by making M drop the part $p(L_i)$. If $\mathcal{L} - \mathcal{L}^+ \neq \emptyset$, then there is a part R_j in $\mathcal{R}$ that is a false partner of each part in $\mathcal{L} - \mathcal{L}^+$; construct a new instance Φ' from Φ by making NM drop the part R_j. This can be justified as follows. Consider any solution of Φ. For any L_i in $\mathcal{L}^+$, if $O \cap L_i \neq \emptyset$, then $M \cap p(L_i) = \emptyset$. If $O \cap L_i = \emptyset$ for all L_i in $\mathcal{L}^+$, then the vertices of O must be in parts in $\mathcal{L} - \mathcal{L}^+$, so $NM \cap R_j = \emptyset$.

Case 2.2′. $|X_2| > \frac{9n_2}{10}$. This case is symmetric to Case 2.2.

Case 2.3. $|W_1| > \frac{9n_1}{10}, |W_2| > \frac{9n_2}{10}$.

Suppose there is a vertex $v \in W_1$ with $d_1(v) \leq \frac{9n_1}{10}$. Let $\mathcal{L}^-$ be the set of parts in $\mathcal{L}$ that have a false partner $p(L_i)$ in $\mathcal{R}$. Note that each part $L_i \in \mathcal{L} - \mathcal{L}^-$ has a true partner $p(L_i)$ in $\mathcal{L}$. Construct p new instances, corresponding to each of the p parts of $\mathcal{L}$ that v can be placed in, as follows. For each $L_i \in \mathcal{L}^-$, construct Φ_{L_i} from Φ by making $S_2 \cap N(v)$ drop $p(L_i)$. For each $L_i \in \mathcal{L} - \mathcal{L}^-$, construct Φ_{L_i} from Φ by making $S_1 - N(v)$ drop $p(L_i)$. A routine argument shows the set of p new instances is equivalent to Φ.

A symmetrical argument settles the case in which there is a vertex $v \in W_2$ with $d_2(v) \leq \frac{9n_2}{10}$.

Now, we may assume that each $v \in W_i$ has $d_i(v) > \frac{9n_i}{10}$, for $i = 1, 2$. By Lemma 2.3, we can find either the sets (a) or the sets (b) as follows.

(a) Pairwise disjoint vertex subsets O, T, and NT of $S_1 \cup S_2$ such that all the following hold:
   (a) $\overline{O}$ is connected.
   (b) There are all possible edges between O and T.
   (c) Each vertex in NT is nonadjacent to some vertex in O.
   (d) $|T \cap S_1| \geq \frac{n_1}{10}$.
   (e) $|T \cap S_2| \geq \frac{n_2}{10}$.
   (f) Either $|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10}$ or $|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10}$.

(b) Disjoint vertex subsets O^*, T^* of $S_1 \cup S_2$ such that all the following hold:
   (a) Either $O^* \subseteq S_1$ and $|O^*| \geq \frac{n_1}{10}$, or $O^* \subseteq S_2$ and $|O^*| \geq \frac{n_2}{10}$.
   (b) $|T^* \cap S_1| \geq \frac{n_1}{10}$.
   (c) $|T^* \cap S_2| \geq \frac{n_2}{10}$.
   (d) There are all possible edges between O^* and T^*.

Case (a). Consider the case $|O \cap S_1| + |NT \cap S_1| \geq \frac{n_1}{10}$. (The case $|O \cap S_2| + |NT \cap S_2| \geq \frac{n_2}{10}$ is symmetric.) Construct at most $p + q$ new instances from Φ as follows.

For each part A_i in $\mathcal{L} \cup \mathcal{R}$ with a false partner $p(A_i)$ in $\mathcal{L} \cup \mathcal{R}$, create Φ_{A_i} by making $T \cap S_1$ drop the part $p(A_i)$ if $p(A_i) \in \mathcal{L}$, or $T \cap S_2$ drop part $p(A_i)$ if $p(A_i) \in \mathcal{R}$.

Now, the remaining parts of $\mathcal{L} \cup \mathcal{R}$ (if they exist) can be named $U_1, U_2, \ldots, U_l$ such that condition 10 of Procedure 2 is satisfied. We create instances as follows:

1. If $l = 1$, then $M_{U_1,U_1} = 1$. If $|O| > 1$, do not create a new instance. Otherwise, create instance Φ_1 as follows. Let v be the single member of O. If $v \in S_1$ and $U_1 \in \mathcal{L}$, then v gets label U_1 and $NT \cap S_1$ drops part U_1. If $v \in S_2$ and $U_1 \in \mathcal{R}$, then v gets label U_1 and $NT \cap S_1$ drops a part $A_j \in \mathcal{L}$ such that $M_{U_1,A_j} = 1$. If neither of these conditions are satisfied, do not create a new instance.
2. If $l \geq 2$ and $M_{U_i,U_j} = 1$ for all $j \neq i$, define Φ_{U_i} from Φ as follows. First, suppose $U_i \in \mathcal{L}$. If U_i is also in $\mathcal{R}$ or if $O \cap S_2 = \emptyset$, then define Φ_{U_i} by making O get the list U_i and $NT \cap S_1$ drop a part $A_j \in S_1$ such that $M_{U_i,A_j} = 1$; otherwise, create no new instance (Φ would not have a solution in this eventuality). Now, suppose $U_i \in \mathcal{R} - \mathcal{L}$. If $O \cap S_1 = \emptyset$, then make $NT \cap S_1$ drop a part $A_j \in S_1$ such that $M_{U_i,A_j} = 1$; otherwise, make no new instance (Φ would not have a solution in this eventuality).
3. If $l \geq 2$ and $M_{U_i,U_j} \neq 1$ for some i, j, then we must have $\{i, j\} = \{l-1, l\}$, $M_{U_{l-1},U_{l-1}} = M_{U_l,U_1} = 1$. Test whether O has a unique partition into two cliques K_1, K_2 (if this is not the case then we do not create a new instance, see the explanation below). We define two instances Φ_1, Φ_2 as follows. In Φ_1, K_1 gets the list U_{l-1} (it drops all other parts), K_2 gets the list U_l; for each vertex x in NT, x drops the part U_{l-1} if x is nonadjacent to some vertex in K_1, or x drops the part U_l if x is nonadjacent to some vertex in K_2. The instance Φ_2 is defined similarly with K_1 getting list U_l and K_2 getting list U_{l-1}.

We now show that the set of new instances are equivalent to Φ. Suppose there is a solution $A_1, \ldots, A_k$ to Φ. It must be the case that for some i, $O \cap A_i \neq \emptyset$. If there is a part $A_i \in \mathcal{L} \cup \mathcal{R}$ with a false partner $p(A_i) \in \mathcal{L} \cup \mathcal{R}$ such that $O \cap A_i \neq \emptyset$, then $T \cap S_j \cap p(A_i) = \emptyset$, where $j = 1$ if $p(A_i) \in \mathcal{L}$ and $j = 2$ if $p(A_i) \in \mathcal{R}$. This eventuality is covered by Φ_{A_i}.

Now suppose there is no part with a false partner that has nonempty intersection with O. Let $U_1, \ldots, U_l$ be the parts of $\mathcal{L} \cup \mathcal{R}$ with no false partners in $\mathcal{L} \cup \mathcal{R}$ (if they exist). These parts must have the clique structure. If $l = 1$, then we have $M_{U_1,U_1} = 1$ and $O \subseteq U_1$ in the solution. Since $\overline{O}$ is connected, it follows that if $|O| > 1$, there is no solution in this eventuality. Therefore, O has exactly one vertex v and it is in S_1 or S_2. If $v \in S_1$, there is a solution only if $U_1 \in \mathcal{L}$ and v is placed in U_1. Then no vertex of $NT \cap S_1$ can be in U_1. If $v \in S_2$, there is a solution only if $U_1 \in \mathcal{R}$ and v is placed in U_1. Then no vertex of $NT \cap S_1$ can be in a part that is a true partner of U_1. In this case, since $|O \cap S_1| = 0$, we have $|NT \cap S_1| \geq \frac{n_1}{10}$.

We can now assume $l \geq 2$. Consider a U_i that is a true partner of all U_j with j different from i. If $O \cap U_i \neq \emptyset$, then we have $O \subseteq U_i$. If $U_i \in \mathcal{L}$, then for there to be a solution with $O \subseteq U_i$, we must have either $U_i \in \mathcal{R}$ or $O \cap S_2 = \emptyset$ (or both). If $U_i \in \mathcal{R} - \mathcal{L}$, then for there to be a solution with $O \subseteq U_i$, we need $O \cap S_1 = \emptyset$, and in this case we have $|NT \cap S_1| \geq \frac{n_1}{10}$. This eventuality is covered by Φ_{U_i}.

Last, we consider the case $M_{U_{l-1},U_{l-1}} = M_{U_l,U_1} = 1$ (both belong to $\mathcal{L} \cap \mathcal{R}$ by condition 10 of Procedure 2) and every vertex in O belongs to $U_{l-1} \cup U_l$. Since $\overline{O}$ is connected, there is a unique partition of O into two cliques K_1, K_2 (if this is not the case, then this eventuality has no solution and so we do not need to create a new

instance). Since every vertex x in NT is nonadjacent to some vertex in O, x must drop part U_{l-1} or U_l; it follows that this eventuality is covered by Φ_1, Φ_2.

Case (b). Consider the case $O^* \subseteq S_1$, $|O^*| \geq \frac{n_1}{10}$. (The case $O^* \subseteq S_2$, $|O^*| \geq \frac{n_2}{10}$ is symmetric.) We construct two new instances from Φ as follows. Choose an $L_i \in \mathcal{L}$ that has a false partner $p(L_i) \in \mathcal{L} \cup \mathcal{R}$ and create Φ_{L_i} by making $T^* \cap S_1$ drop $p(L_i)$ if $p(L_i) \in \mathcal{L}$, or by making $T^* \cap S_2$ drop $p(L_i)$ if $p(L_i) \in \mathcal{R}$. Then create Φ' by making O^* drop L_i. This can be justified as follows. In any solution to Φ, if for some $L_i \in \mathcal{L}$ we have $L_i \cap O^* \neq \emptyset$, then $T^* \cap S_j \cap p(L_i) = \emptyset$, where $j = 1$ if $p(L_i) \in \mathcal{L}$ and $j = 2$ if $p(L_i) \in \mathcal{R}$; otherwise, $O^* \cap L_i = \emptyset$.

It is easily verified that in each instance Γ created, $|S_{\mathcal{L}}(\Gamma)|\,|S_{\mathcal{R}}(\Gamma)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|\,|S_{\mathcal{R}}(\Phi)|$. If no new instances are produced by the above analysis, then Φ has no solution. This completes the description of Procedure 2. ∎

Procedure 3. We note that our Procedure 3, in principle, is the same as Procedure 4 in [15].

Input: Instance Φ of the list M-partition problem with set $\mathcal{Z}$ of parts $A_1, \ldots, A_k$, and two sets $\mathcal{L}$ and $\mathcal{R}$, which are subsets of $\mathcal{Z}$, such that $S_{\mathcal{L}} \neq \emptyset, S_{\mathcal{R}} \neq \emptyset$, $\mathcal{L} \not\subseteq \mathcal{R}$, $\mathcal{R} \not\subseteq \mathcal{L}$, and we can write $\mathcal{L} = \{L_1, L_2\}$ and $\mathcal{R} = \{R_1, \ldots, R_q\}$ $(q \geq 2)$ so that L_1 has a false partner in $\mathcal{R}$ and L_2 has a true partner in $\mathcal{R}$.

Output: The set of instances $\{\Phi_1, \Phi_2\}$ that is equivalent to Φ, and such that $|S_{\mathcal{L}}(\Phi_i)|\,|S_{\mathcal{R}}(\Phi_i)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|\,|S_{\mathcal{R}}(\Phi)|$, or a proof that Φ has no solution.

Note 3. Given an instance Φ on a graph G with n vertices that satisfies the conditions of Procedure 3, recursively applying Procedure 3 produces a polynomial number of instances Φ' for which $S_{\mathcal{L}}(\Phi') = \emptyset$ or $S_{\mathcal{R}}(\Phi') = \emptyset$, and the set of instances Φ' is equivalent to Φ. It is easy to see that the number of instances Φ' is at most $(2)^{\log_{\frac{10}{9}} n^2} = n^{2\log_{\frac{10}{9}} 2}$. ∎

Details of Procedure 3. Write $S_1 = S_{\mathcal{L}}, S_2 = S_{\mathcal{R}}$. Let $n_1 = |S_1|$ and $n_2 = |S_2|$. For a vertex $v \in S_1 \cup S_2$, let $d_i(v) = |N(v) \cap S_i|$, $i = 1, 2$. Let $p(L_i) \in \mathcal{R}$ be the partner of L_i, $i = 1, 2$.

Case 1. There is a vertex v in S_1 with $\frac{n_2}{10} \leq d_2(v) \leq \frac{9n_2}{10}$.

Construct two instances from Φ corresponding to v being placed in L_i, $i = 1, 2$. One instance is constructed by making $S_2 \cap N(v)$ drop the part $p(L_1)$ and another is constructed by making $S_2 - N(v)$ drop the part $p(L_2)$. It is a routine matter to verify that the set of new instances is equivalent to Φ.

Case 2. Every vertex in $S_{\mathcal{L}}$ satisfies $d_2(v) < \frac{n_2}{10}$ or $d_2(v) > \frac{9n_2}{10}$.

Define two sets as follows:

$$X_1 = \left\{ v \in S_1 | d_2(v) < \frac{n_2}{10} \right\},\ W_1 = \left\{ v \in S_1 | d_2(v) > \frac{9n_2}{10} \right\}.$$

There are two cases to consider.

Case 2.1. $|X_1| \geq \frac{n_1}{2}$.

Find the sets O, M, and NM as defined in Lemma 2.2.

Suppose first that $|O| \geq \frac{n_1}{10}$ and $|M| > \frac{n_2}{2}$. Replace Φ with two new instances Φ_1, Φ_2 constructed as follows. Φ_1 is obtained from Φ by making M drop the part $p(L_2)$; Φ_2 is obtained from Φ by making O drop the part L_2. This can be justified as follows. Consider any solution of Φ. If $O \cap L_2 \neq \emptyset$, then no vertex in M can be in part $p(L_2)$; otherwise, no vertex of O is in L_2. Thus, the two new instances Φ_1, Φ_2 cover all the eventualities.

Now, we may assume that $\frac{2n_2}{5} \leq |M| \leq \frac{n_2}{2}$ and $|NM| \geq \frac{n_2}{2}$. Replace Φ with two new instances Φ_1, Φ_2 constructed as follows. Φ_1 is obtained from Φ by making

M drop the part $p(L_2)$ and Φ_2 is obtained from Φ by making NM drop the part $p(L_1)$. This can be justified as follows. Consider any solution of Φ. If $O \cap L_2 \neq \emptyset$, then no vertex in M can be in part $p(L_2)$. Otherwise, no vertex of O is in part L_2; hence, every vertex of O is placed in L_1. Since every vertex in NM has a neighbor in O, this implies $NM \cap p(L_1) = \emptyset$. Thus, the two new instances Φ_1, Φ_2 cover all the eventualities.

Case 2.2. $|X_1| < \frac{n_1}{2}$; hence, $|W_1| \geq \frac{n_1}{2}$.

Observe that in this situation, with respect to the adjacencies in the complement of the graph under consideration, we have $|X_1| \geq \frac{n_1}{2}$. Therefore, we can construct a set of two instances equivalent to Φ in this case by using the logic for Case 2.1 in the complement of the given graph using $\overline{M}$ and by simply reversing the roles played by L_1 and L_2.

Finally, it can be easily verified that for each instance Γ created, $|S_{\mathcal{L}}(\Gamma)|\,|S_{\mathcal{R}}(\Gamma)| \leq \frac{9}{10}|S_{\mathcal{L}}(\Phi)|\,|S_{\mathcal{R}}(\Phi)|$. If no new instances are produced by the above analysis, then Φ has no solution. This completes the description of Procedure 3. ∎

4. The main theorem. In this section we focus on the main result of the paper which concerns all list M-partition problems where M is a symmetric 4×4 matrix over $\{0, 1, *\}$. In the following we will refer to the four parts of the partition as A, B, C, and D. Recall from section 1 that the *stubborn problem* is the list M-partition problem where $M_{A,A} = 0$, $M_{B,B} = 0$, $M_{D,D} = 1$, $M_{A,C} = M_{C,A} = 0$, and all other entries are asterisks (see Figure 1.1). The stubborn problem has been shown to be solvable in quasi-polynomial time in [22]; hence, it is unlikely to be NP-complete.

THEOREM 4.1. *Suppose M with dimension 4 is neither the matrix for the stubborn problem nor its complement. Then the list M-partition problem is solvable in polynomial time or NP-complete. In particular, the list M-partition problem is solvable in polynomial time, except when M contains the matrix for 3-colorability, stable cutset, or their complements, or M is the matrix for stable cutset pair, $2K_2$, or their complements, in which cases the problem is NP-complete.*

In proving Theorem 4.1 we employ the tools and procedures described in the previous sections. Given an instance $\mathcal{I}$ of the list M-partition problem, Procedures 1, 2, and 3 are recursively applied to create a polynomial number of new instances $\mathcal{I}_i$ that together are equivalent to the given instance. The resulting instances $\mathcal{I}_i$ are each such that there is a list $\mathcal{L}$ for which the set of vertices $S_{\mathcal{L}}(\mathcal{I}_i)$ with list $\mathcal{L}$ is empty, whereas $S_{\mathcal{L}}(\mathcal{I})$ was not empty. Care must be taken in applying the procedures and tools not to recreate vertices with list $\mathcal{L}$ and thus, reintroduce $S_{\mathcal{L}}$ into subsequent instances of the problem. This can happen as a result of the procedures and tools themselves or the reduction operation that is applied whenever a new instance is created. If any list is (re-)introduced, this list will be a proper subset of a list involved in the operation. This can be easily verified by examining the details of the procedures and the reduction operation.

For simplicity, we write $\mathcal{L}$ (without set brackets) for $\mathrm{S}_{\mathcal{L}}$; for example, $ABC = S_{ABC}$. Theorem 4.1 will be proved via a sequence of lemmata, similar to the treatment in [22].

Proof of Theorem 4.1. If M is a matrix over $\{0, *\}$ or $\{1, *\}$, the result follows from Corollary 1.2 [16, 19, 20, 21]. We can therefore assume that M has at least one 0 and at least one 1. By Theorem 1.3, the only 3-part subproblems that are NP-complete are the stable cutset problem, the 3-colorability problem and their complements, and all others are solvable in polynomial time. By Tool 5, if M contains the matrix for any of these NP-complete subproblems, then the problem is NP-complete. Otherwise,

the following lemmata show that the problem can be reduced to a polynomial number of instances that are together equivalent to the given instance, and such that each instance can be solved in polynomial time. The NP-completeness results we employ are well known [14, 27].

The next two lemmata cover the cases when M has an off-diagonal 0 and an off-diagonal 1.

LEMMA 4.2. *Suppose $M_{A,B} = 1$ and $M_{C,D} = 0$. Then the list M-partition problem is solvable in polynomial time or NP-complete.*

Proof. Recall Notes 1, 2, and 3. Given the original instance $\mathcal{I}$, if $ABCD$ is not empty, we recursively apply Procedure 1 with $\mathcal{L} = \{A, B, C, D\}$, $\mathcal{U} = \{A, B\}$, and $\mathcal{F} = \{C, D\}$ to obtain a polynomial number of instances $\mathcal{I}_i$ that together are equivalent to $\mathcal{I}$ such that, for each i, $ABCD(\mathcal{I}_i) = \emptyset$. We now consider the instances $\mathcal{I}_i$.

For each instance $\mathcal{I}_i$, first recursively apply Procedure 2 with $\mathcal{L} = \{A, B, C\}$ and $\mathcal{R} = \{A, B, D\}$, and then (working in $\overline{G}$ using $\overline{M}$) recursively apply Procedure 2 to the resulting instances with $\mathcal{L} = \{A, C, D\}$ and $\mathcal{R} = \{B, C, D\}$ to obtain a polynomial number of instances $\mathcal{J}_j$ that together are equivalent to $\mathcal{I}_i$ and such that, for each j, either $ABC(\mathcal{J}_j) = \emptyset$ or $ABD(\mathcal{J}_j) = \emptyset$, and either $ACD(\mathcal{J}_j) = \emptyset$ or $BCD(\mathcal{J}_j) = \emptyset$.

We now consider the resulting instances $\mathcal{J}_j$. There are four types:

1. $ABC, ACD \neq \emptyset$,
2. $ABC, BCD \neq \emptyset$,
3. $ABD, ACD \neq \emptyset$,
4. $ABD, BCD \neq \emptyset$.

Since the four types are symmetric, we only need to consider instances $\mathcal{J}_j$ of type 1. In this case the possible remaining nonempty sets are ABC, ACD, AB, AC, AD, BC, BD, CD. Recursively, apply Procedure 3 to each $\mathcal{J}_j$ and then to the resulting instances with pairs $\mathcal{L}, \mathcal{R}$, in the following sequence: step (a) $\mathcal{L} = \{B, D\}$ and $\mathcal{R} = \{A, C, D\}$, step (b) $\mathcal{L} = \{B, D\}$ and $\mathcal{R} = \{A, B, C\}$, step (c) $\mathcal{L} = \{A, D\}$ and $\mathcal{R} = \{A, B, C\}$, until one of the two sets involved is empty. This will produce (and will be justified shortly) a polynomial number of instances $\mathcal{K}_k$ that together are equivalent to $\mathcal{J}_j$ and such that each instance $\mathcal{K}_k$ has possible remaining nonempty sets as in one of the following cases:

Case 1. AB, AC, AD, BC, BD, CD,

Case 2. ABC, AB, AC, BC, CD,

Case 3. ABC, ACD, AB, AC, AD, BC, CD.

After step (a), the new instances $\mathcal{K}_k$ either have $BD = \emptyset$ (Case 3) or $ACD = \emptyset$. After step (b), we have either $ABC = \emptyset$ (Case 1) or $BD = \emptyset$ (Case 3 again). In the latter case, we proceed to step (c), after which we have either $AD = \emptyset$ (Case 2) or $ABC = \emptyset$ (Case 1). (Note that if there are vertices with lists of length 3, then the reduction operation may produce a vertex with a list of length 1 or 2 that can be derived from the length 3 list by dropping parts.)

Now we consider the instances $\mathcal{K}_k$.

Case 1. This case can be formulated as a 2-satisfiability problem (2-SAT) and solved in polynomial time (see Tool 1).

Case 2. In the case that $M_{A,C} = 1$ or $M_{B,C} = 1$, we recursively apply Procedure 3 with $\mathcal{L} = \{C, D\}$ and $\mathcal{R} = \{A, B, C\}$. This will create instances in each of which either ABC is empty or CD is empty. In the former case, the problem reduces to 2-SAT. In the latter case, in every instance created there is no vertex with a list containing part D. Thus, the problem is reduced to a 3-part list M-partition problem (3-part problem).

If $M_{A,C} = 0$ ($M_{B,C} = 0$), we recursively apply Procedure 1 with $\mathcal{L} = \{A, B, C\}$, $\mathcal{U} = \{A, B\}$, and $\mathcal{F} = \{A, C\}$ ($\mathcal{L} = \{A, B, C\}$, $\mathcal{U} = \{A, B\}$, and $\mathcal{F} = \{B, C\}$). This will create instances in which ABC is empty, and thus, the problem is reduced to 2-SAT.

Therefore, we can now assume that $M_{A,C} = M_{B,C} = *$.

If $M_{C,C} = 1$, by Tool 3, we create one instance in which no vertex has part C in its list, and at most n instances in each of which no vertex has both C and D in its list. In these cases, the problem is reduced to a 3-part problem.

If $M_{C,C} = 0$, then we recursively apply Procedure 1 with $\mathcal{L} = \{A, B, C\}$, $\mathcal{U} = \{A, B\}$, and $\mathcal{F} = \{C\}$. This will create instances which can be solved using 2-SAT.

Hence, we can now assume that $M_{C,C} = *$.

If $M_{A,A} = 0$ ($M_{B,B} = 0$), by Tool 3, we create one instance in which no vertex has part A (part B) in its list, and at most n instances in each of which no vertex has both A and B in its list. In these cases, the problem is reduced to 2-SAT.

If $M_{A,A} = *$, then as $M_{C,C} = M_{A,C} = *$, the instance can be solved trivially by first placing the vertices whose lists contain A in part A, and then placing any remaining vertices whose lists contain C in part C. Similarly, if $M_{B,B} = *$, the problem can be solved trivially.

So, we can now assume that $M_{A,A} = M_{B,B} = 1$.

If $M_{B,D} = 0$ ($M_{A,D} = 0$), then C dominates B (C dominates A). We can then use Tool 4 to derive an equivalent instance where no vertex has the list ABC, and hence can be solved using 2-SAT.

If $M_{B,D} = 1$ ($M_{A,D} = 1$), by Tool 3, we create one instance in which no vertex has part D in its list, and at most n instances in each of which no vertex has both B and C (both A and C) in its list. In these cases, we either have an instance that is a 3-part problem or can be solved using 2-SAT.

We finally can assume that $M_{A,D} = M_{B,D} = *$. Since B dominates A, we can use Tool 4 to derive an equivalent instance where no vertex has the list ABC, and hence, can be solved using 2-SAT.

Case 3. Suppose we are able to produce an equivalent set of instances in each of which $ACD = \emptyset$, and hence, the possible nonempty sets are ABC, AB, AC, AD, BC, CD. Then, recursively applying Procedure 3 with $\mathcal{L} = \{A, D\}$ and $\mathcal{R} = \{A, B, C\}$ will produce instances each of which either can be solved using 2-SAT or has $AD = \emptyset$, which is settled by Case 2. A similar analysis can be made when an equivalent set of instances can be produced in each of which $ABC = \emptyset$. Suppose $ABC = \emptyset$, recursively applying Procedure 3 with $\mathcal{L} = \{B, C\}$ and $\mathcal{R} = \{A, C, D\}$ will produce instances each of which either can be solved using 2-SAT or has $BC = \emptyset$. The latter case is reduced to Case 2 by working in $\overline{G}$ in place of G and using $\overline{M}$ in place of M. Therefore, we aim to produce equivalent instances in each of which either $ABC = \emptyset$ or $ACD = \emptyset$.

If $M_{A,C} = 0$, then the lists $\mathcal{L} = \{A, B, C\}$ and $\mathcal{R} = \{A, C, D\}$ fail to satisfy the conditions for Procedure 2. However, with respect to $\overline{G}$ and $\overline{M}$, they do satisfy the conditions for Procedure 2. Hence, we recursively apply Procedure 2 with $\mathcal{L}$ and $\mathcal{R}$ in $\overline{G}$ using $\overline{M}$ to create instances in each of which either $ABC = \emptyset$ or $ACD = \emptyset$.

If $M_{A,C} = 1$, then recursively apply Procedure 2 with $\mathcal{L} = \{A, B, C\}$ and $\mathcal{R} = \{A, C, D\}$ to create instances in each of which either $ABC = \emptyset$ or $ACD = \emptyset$.

We can therefore assume that $M_{A,C} = *$.

If $M_{A,A} = 0$ ($M_{B,B} = 0$), using Tool 3, we create one instance in which no vertex has part A (part B) in its list, and at most n instances in each of which no vertex has both A and B in its list; hence, $ABC = \emptyset$.

If $M_{A,A} = 1$, recursively apply Procedure 1 with $\mathcal{L} = \{A, C, D\}$, $\mathcal{U} = \{A\}$, and $\mathcal{F} = \{C, D\}$ to produce instances in each of which $ACD = \emptyset$.

Therefore, we can assume that $M_{A,A} = *$.

If $M_{C,C} = 1$, using Tool 3, we create one instance in which no vertex has part C in its list, and at most n instances in each of which no vertex has both C and D in its list; hence, $ACD = \emptyset$.

If $M_{C,C} = 0$, recursively apply Procedure 1 with $\mathcal{L} = \{A, B, C\}, \mathcal{U} = \{A, B\}$, and $\mathcal{F} = \{C\}$ to produce instances in each of which $ABC = \emptyset$.

Therefore, we can assume that $M_{C,C} = *$.

We now have instances in which $M_{A,A} = M_{C,C} = M_{A,C} = *$. Such an instance can be solved trivially by first placing the vertices whose lists contain A in part A, and then placing any remaining vertices whose lists contain C in part C. □

Recall that the *list generalized $\mathcal{P}$ problem* is the list M'-partition problem where M' is obtained from the matrix M for list partition problem $\mathcal{P}$ by changing some asterisks to either 0 or 1.

COROLLARY 4.3. *Each list generalized skew partition problem is solvable in polynomial time, except when it contains the stable cutset problem or its complement, in which cases the problem is NP-complete.*

Proof. Observe (via Theorem 1.3) that the only possible 3-part subproblems that are NP-complete are the stable cutset problem and its complement, and all others are solvable in polynomial time. By Tool 5, if M contains the matrix for the stable cutset problem or its complement, then the problem is NP-complete. Otherwise, the problem is polynomial-time solvable by Lemma 4.2. □

From here on, we write the proofs in an abbreviated style. When Tool 3 is applied an instance that is a 3-part problem is always created; this will now be assumed and not explicitly stated. The full details can be written in the same manner as the proof of Lemma 4.2.

LEMMA 4.4. *Suppose $M_{A,B} = 0$ and $M_{A,D} = 1$. Then the list M-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, either the problem is reduced to a 3-part problem (no A), or no list contains $\{B, D\}$. Assuming the latter, the possible nonempty sets are ABC, ACD, AB, AC, AD, BC, CD. By Lemma 4.2 we may assume $M_{C,D} \neq 1$ and $M_{B,C} \neq 0$.

Suppose $M_{A,C} = 1$. Then no list contains $\{B, C\}$ (using Tool 3). Apply Procedure 3 to the pair AB, ACD. If AB becomes empty, then we have a 3-part problem on $\{A, C, D\}$. Otherwise, the instance can be solved using 2-SAT.

Suppose $M_{A,C} = 0$, and no list contains $\{C, D\}$ (using Tool 3). Apply Procedure 3 to the pair AD, ABC to get either 3-part problems or instances solvable using 2-SAT. Therefore, $M_{A,C} = *$.

If $M_{C,D} = 0$, then no list contains $\{A, C\}$ (using Tool 3) and we get instances solvable using 2-SAT. Therefore, $M_{C,D} = *$.

If $M_{B,C} = 1$, then no list contains $\{A, C\}$ (using Tool 3) and we get instances solvable using 2-SAT. Therefore, $M_{B,C} = *$.

If $M_{C,C} = *$, then C dominates parts A, B, and D, and we get an instance solvable using 2-SAT.

If $M_{C,C} = 0$, apply Procedure 1 to ACD so that ACD becomes empty. Then apply Procedure 3 to the pair AD, ABC. If ABC becomes empty, we get instances solvable using 2-SAT. Otherwise, now apply Procedure 3 to the pair CD, ABC to get 3-part problems or instances solvable using 2-SAT.

Now we have $M_{C,C} = 1$. Apply Procedure 1 to ABC so that ABC becomes empty. Then apply Procedure 3 to the pair AB, ACD. If ACD becomes empty, we get instances solvable using 2-SAT. Otherwise, now apply Procedure 3 to the pair BC, ACD to get 3-part problems or instances solvable using 2-SAT. □

Graphs for which the vertex-set can be partitioned into two stable sets and two cliques are called (2,2)-graphs. Brandstädt [2, 3] introduced this class and gave the first polynomial-time algorithm for recognition. Recognition of (2,2)-graphs is the M-partition problem where $M_{A,A} = 1$, $M_{B,B} = 1$, $M_{C,C} = 0$, and $M_{D,D} = 0$, and all other entries are asterisks. The following result was proved in [22]; we provide a proof using different techniques.

LEMMA 4.5. *All list generalized* (2,2)*-graph recognition problems are solvable in polynomial time.*

Proof. Repeatedly apply Procedure 1 to the following sets to eliminate them, one by one: $ABCD$, ABC, ABD, ACD, BCD. Then use 2-SAT. □

Based on the previous lemmata and Tool 6, we can now assume that 1 occurs only on the diagonal and that the off-diagonal entries are either 0 or $*$. We first consider the case that there are at least two 1's on the diagonal.

LEMMA 4.6. *Suppose there are at least two* 1*'s on the diagonal and all off-diagonal entries are* $*$*. Then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. If one of the diagonal entries, say $M_{A,A}$, is $*$, A dominates the other parts; hence, the problem can be reduced to a 3-part problem on $\{B, C, D\}$. On the other hand, suppose none of the diagonal entries are $*$. When there are two 1's and two 0's on the diagonal we get a problem solvable in polynomial time (see Lemma 4.5). Otherwise, the problem is NP-complete via the complement of 3-colorability and Tool 5. (This subcase is also covered by Theorem 1.4.) □

We can now assume that there is at least one off-diagonal entry that is 0. The next three lemmata cover the possible position of the off-diagonal 0 with respect to the two or more 1's assumed to be on the diagonal.

LEMMA 4.7. *Suppose all off-diagonal entries are* 0 *or* $*$*,* $M_{B,B} = M_{D,D} = 1$*, and* $M_{A,C} = 0$*. Then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Apply Procedure 1 to eliminate set $ABCD$.

Apply Procedure 1 to eliminate set ABC.

Apply Procedure 1 to eliminate set ACD.

Apply Procedure 2 to the pair ABD, BCD so that one of the sets is eliminated.

Assume $BCD = \emptyset$. (The other case is similar.)

Apply Procedure 3 to the pair BC, ABD so that one of the sets is eliminated.

Apply Procedure 3 to the pair CD, ABD so that one of the sets is eliminated.

We now can assume that the remaining nonempty sets are ABD, AB, AC, AD, BD; otherwise, we can use 2-SAT.

If $M_{A,B} = 0$, then, using Tool 3, no list contains $\{A, B\}$; we can now use 2-SAT.

If $M_{A,D} = 0$, then, using Tool 3, no list contains $\{A, D\}$; we can now use 2-SAT.

Otherwise, the hypothesis of the lemma implies $M_{A,B} = M_{A,D} = *$.

If $M_{A,A} = 0$, then apply Procedure 1 to ABD; we can now use 2-SAT.

If $M_{A,A} = 1$, then, by Tool 3, no list contains $\{A, C\}$; we get a 3-part problem.

Therefore, $M_{A,A} = *$.

If $M_{B,C} = 0$, then A dominates B and no list contains $\{A, B\}$; we can now use 2-SAT.

If $M_{C,D} = 0$, then A dominates D and no list contains $\{A, D\}$; we can now use 2-SAT.

Otherwise, the hypothesis of the lemma implies $M_{B,C} = M_{C,D} = *$.

If $M_{C,C} = 0$, then A dominates C and no list contains $\{A, C\}$; we get a 3-part problem.

If $M_{C,C} = 1$, then, by Tool 3, no list contains $\{A, C\}$; we get a 3-part problem.

Therefore, $M_{C,C} = *$.

Place vertices with list AC in the part A to get a 3-part problem on $\{A, B, D\}$ in which A dominates other parts; we can now use 2-SAT. □

COROLLARY 4.8. *The list* 2*-clique cutset problem is solvable in polynomial time.*

Proof. Lemma 4.7 covers the list 2-clique cutset problem: $M_{B,B} = M_{D,D} = 1$, $M_{A,C} = M_{C,A} = 0$, and all other entries are asterisks. It can be verified (via Theorem 1.3) that every 3-part problem produced in that case in the proof of Lemma 4.7 is solvable in polynomial time. □

COROLLARY 4.9. *Each list generalized* 2*-clique cutset problem is solvable in polynomial time, except when it contains the complement of the* 3*-colorability problem, in which case it is NP-complete.*

Proof. Observe (via Theorem 1.3) that in this case the only possible 3-part subproblem that is NP-complete is the complement of 3-colorability, and all others are solvable in polynomial time. By Tool 5, if M contains the matrix for the complement of 3-colorability, then the problem is NP-complete. Otherwise, the problem is polynomial-time solvable by Lemmata 4.2, 4.4, and 4.7. □

LEMMA 4.10. *Suppose all off-diagonal entries are* 0 *or* $*$*,* $M_{A,A} = M_{B,B} = 1$*, and* $M_{A,C} = 0$*. Then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, C\}$. The possible nonempty sets are ABD, BCD, AB, AD, BC, BD, CD.

By previous lemmata, $M_{C,D} = *$ and $M_{D,D} \neq 1$.

If $M_{D,D} = 0$, then apply Procedure 1 to ABD. Then, apply Procedure 3 to the pair AB, BCD. If BCD is eliminated we can use 2-SAT; otherwise, apply Procedure 1 to AD to get a 3-part problem.

Therefore, $M_{D,D} = *$.

If $M_{A,D} = 0$, then, using Tool 3, no list contains $\{A, D\}$. Apply Procedure 3 to the pair AB, BCD to get a 3-part problem or we can use 2-SAT.

Therefore, $M_{A,D} = *$.

If $M_{B,D} = 0$, then, using Tool 3, no list contains $\{B, D\}$; we can now use 2-SAT.

Therefore, $M_{B,D} = *$.

Now, D dominates the other parts; we get a 3-part problem. □

LEMMA 4.11. *Suppose all off-diagonal entries are* 0 *or* $*$*,* $M_{A,A} = M_{C,C} = 1$*, and* $M_{A,C} = 0$*. Then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. From previous lemmata, $M_{A,B} = M_{A,D} = M_{B,C} = M_{B,D} = M_{C,D} = *$, $M_{B,B} \neq 1$, and $M_{D,D} \neq 1$. Using Tool 3, no list contains $\{A, C\}$.

If $M_{B,B} = *$, then B dominates other parts; we get a 3-part problem.

If $M_{D,D} = *$, then D dominates other parts; we get a 3-part problem.

Therefore, $M_{B,B} = M_{D,D} = 0$.

Apply Procedure 1 to ABD. Then, apply Procedure 1 to BCD; we can now use 2-SAT. □

In the remaining case M has exactly one 1 and it is on the diagonal; say $M_{D,D} = 1$. Following [22], we define a *separating statement* for $X = A, B$, or C to be "$M_{X,D} = 0$ or $M_{X,X} = 0$." We divide the remaining cases based on the number of separating statements that hold being three, two, or at most one. The following four lemmata cover the cases when three separating statements hold.

LEMMA 4.12. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{A,A} = M_{B,B} = M_{C,C} = 0$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. If the subproblem on the parts $\{A, B, C\}$ corresponds to 3-colorability, then we have an NP-complete problem by Tool 5. Therefore, without loss of generality, we can assume $M_{A,B} = 0$.

If $M_{B,C} = 0$, then the following reduces the instance to 3-part problems: Apply Procedure 1 to $ABCD$, apply Procedure 1 to ABD, apply Procedure 1 to ACD, apply Procedure 1 to BCD, apply Procedure 1 to AD, apply Procedure 1 to BD, and then apply Procedure 1 to CD.

Therefore, $M_{B,C} = *$. Similarly, $M_{A,C} = *$.

As there is a single 1 in M, each of $M_{A,D}$ and $M_{B,D}$ is constrained to be 0 or $*$. In any such case, A dominates B or B dominates A, and no list contains $\{A, B\}$. Apply Procedure 1 to ACD. Then, apply Procedure 1 to BCD. We can now use 2-SAT. □

LEMMA 4.13. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{B,B} = M_{C,C} = M_{A,D} = 0$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, D\}$. The following will then produce 3-part problems on $\{A, B, C\}$: apply Procedure 1 to BCD, apply Procedure 1 to BD, and then apply Procedure 1 to CD. □

LEMMA 4.14. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{C,C} = M_{A,D} = M_{B,D} = 0$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, D\}$ and also no list contains $\{B, D\}$. Apply Procedure 1 to CD to get 3-part problems on $\{A, B, C\}$. □

LEMMA 4.15. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{A,D} = M_{B,D} = M_{C,D} = 0$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, D\}$, no list contains $\{B, D\}$, and also no list contains $\{C, D\}$. We get 3-part problems on $\{A, B, C\}$. □

The next three lemmata cover the cases when exactly two separating statements hold, say for A and B.

LEMMA 4.16. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{A,D} = M_{B,D} = 0$ *and* $M_{C,C} = M_{C,D} = *$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, D\}$ and also no list contains $\{B, D\}$. C dominates D and no list contains $\{C, D\}$. We get 3-part problems on $\{A, B, C\}$. □

LEMMA 4.17. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{A,A} = M_{B,B} = 0$, $M_{C,C} = M_{C,D} = *$, *and the list* M*-partition problem is different from the stubborn problem, then it is solvable in polynomial time or NP-complete.*

Proof. If $M_{A,C} = M_{B,C} = *$, then C dominates all other parts, so we obtain a 3-part problem. Therefore, without loss of generality, assume $M_{A,C} = 0$.

Suppose $M_{B,C} = 0$. Then we can apply Procedure 1 to eliminate the following sets in sequence: $ABCD, ABD, ACD, BCD, AD, BD$. Let $X = AB$ and Y be the

union of sets ABC, AC, BC, and CD (i.e., X is the set of vertices with list $\{A, B\}$ and Y is the set of vertices with any of the possible remaining lists). Suppose we had $u \in X$ and $v \in Y$ such that u and v are adjacent. Since $M_{A,C} = M_{B,C} = 0$, in any solution to the problem, v cannot be placed in part C. Therefore, by making such vertices v drop the part C from their lists (hence, leave the set Y), we get instances where there are no edges between vertices in X and vertices in Y. We can then solve such an instance by placing every vertex in Y in part C and testing whether X induces a bipartite graph. Therefore, we can assume that $M_{B,C} = *$ (and $M_{A,C} = 0$). C dominates A, so no list contains $\{A, C\}$.

Apply Procedure 1 to ABD, then to AD, and then to BD. The possible remaining nonempty sets are BCD, AB, BC, CD.

If $M_{A,B} = 0$, then C dominates B, so no list contains $\{B, C\}$; we can now use 2-SAT.

If $M_{A,D} = 0$, then C dominates D, so no list contains $\{C, D\}$; we can now use 2-SAT.

If $M_{B,D} = 0$, then (using Tool 3) no list contains $\{B, D\}$; we can now use 2-SAT.

Therefore, $M_{A,B} = M_{A,D} = M_{B,D} = *$ and we are left with the stubborn problem. □

We note that the proof of Lemma 4.17 shows that the stubborn problem can be reduced to an equivalent set of instances where for each instance the only possible lists are A, B, C, D, AB, BC, CD, and BCD.

LEMMA 4.18. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{A,D} = M_{B,B} = 0$, $M_{A,A} = M_{B,D} = *$, *and* $M_{C,C} = M_{C,D} = *$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Using Tool 3, no list contains $\{A, D\}$. Therefore, the possible remaining nonempty sets are ABC, BCD, AB, AC, BC, BD, CD.

If $M_{A,C} = *$ or $M_{A,B} = 0$, then C dominates B, so no list contains $\{B, C\}$, and we can use 2-SAT. Therefore, $M_{A,C} = 0$ and $M_{A,B} = *$.

If $M_{B,C} = *$, then the subproblem on $\{A, B, C\}$ is the stable cutset problem. Therefore, $M_{B,C} = 0$.

Apply Procedure 1 to BCD and then to BD. Now the possible remaining nonempty sets are ABC, AB, AC, BC, CD. Let $X = AB$ and Y be the union of sets ABC, AC, BC, and CD (i.e., X is the set of vertices with list $\{A, B\}$ and Y is the set of vertices with any of the possible remaining lists). Suppose we had $u \in X$ and $v \in Y$ such that u and v are adjacent. Since $M_{A,C} = M_{B,C} = 0$, in any solution to the problem, v cannot be placed in part C. Therefore, by making such vertices v drop the part C from their lists (hence, leave the set Y), we get instances where there are no edges between vertices in X and vertices in Y. We can then solve such an instance by placing every vertex in Y in part C, and placing every vertex in X in part A. □

The only remaining case is when the only 1 is at $M_{D,D}$ and at most one separating statement holds, say, for part A.

LEMMA 4.19. *Suppose the only* 1 *is at* $M_{D,D}$. *If* $M_{B,B} = M_{B,D} = M_{C,C} = M_{C,D} = *$, *then the list* M*-partition problem is solvable in polynomial time or NP-complete.*

Proof. Suppose $M_{B,C} = *$. If $M_{A,B} = *$ ($M_{A,C} = *$), then B (C) dominates all the other parts to yield a 3-part problem. On the other hand, if $M_{A,B} = M_{A,C} = 0$, then rows B and C in M are identical; hence, parts B and C can be identified.

Therefore, we can assume $M_{B,C} = 0$. We divide the cases based on the value of

the triple $(M_{A,A}, M_{A,B}, M_{A,C})$.

Case $(*,*,*)$. If $M_{A,D} = *$, then A dominates all other parts to yield a 3-part problem. Hence, $M_{A,D} = 0$, and by Tool 3, no list contains $\{A, D\}$.

Apply Procedure 1 to BCD. Apply Procedure 3 to the pair BD, CD. Solve the problem as follows: Place vertices with lists $\{A, B, C\}$, $\{A, B\}$, or $\{A, C\}$ in part A. Then, if BD is empty, place vertices with lists $\{B, C\}$ or $\{C, D\}$ in part C, and if CD is empty, place vertices with lists $\{B, C\}$ or $\{B, D\}$ in part B.

Case $(*, 0, *)$. If $M_{A,D} = *$, then rows A and C in M are identical; hence, parts A, C can be identified. Therefore, we can assume $M_{A,D} = 0$. Then, by Tool 3, no list contains $\{A, D\}$. Also, C dominates A and no list contains $\{A, C\}$. Apply Procedure 1 to BCD and then use 2-SAT.

Case $(0, 0, 0)$, $(0, 0, *)$. C dominates A, so no list contains $\{A, C\}$. Apply Procedure 1 to ABD, then to BCD, and use 2-SAT.

Case $(0, *, *)$. This case contains the stable cutset problem; hence, by Tool 5, it is NP-complete.

Case $(*, 0, 0)$. Apply Procedure 1 to the following sets one by one: $ABCD$, ABD, ACD, and BCD. If we had $u \in AB$ (BC, AC) and $v \in ABC$ such that u and v are adjacent, then v must drop C (A, B, respectively) and leave ABC. Therefore, there are no edges between ABC and any of AB, BC, or AC. Now, apply Procedure 3 to AD, CD, then to AD, BD, and finally to CD, BD. We can now assume that exactly one of AD, BD, CD is nonempty.

Suppose AD is nonempty. If $M_{A,D} = 0$, then use Tool 3 to eliminate the set AD and obtain a 3-part problem. If $M_{A,D} = *$, then place vertices with list ABC in part A and solve using 2-SAT. If BD is nonempty, then place vertices with list $\{A, B, C\}$ in part B and solve using 2-SAT. If CD is nonempty, then place vertices with list $\{A, B, C\}$ in part C and solve using 2-SAT.

Case $(0, *, 0)$, $(*, *, 0)$. B dominates A, so no list contains $\{A, B\}$. Apply Procedure 1 to ACD, then to BCD, and solve using 2-SAT. □

Thus, Theorem 4.1 is proved via the following cases.

1. M is a matrix over $\{0, *\}$ or $\{1, *\}$: Corollary 1.2.
2. M has at least one 0 and at least one 1:
   - 2.1 M has an off-diagonal 0 and an off-diagonal 1: Lemmata 4.2 and 4.4.
   - 2.2 M has 1 (resp., 0) only on the diagonal and all off-diagonal entries are either 0 or $*$ (resp., 1 or $*$):
     - 2.2.1 M has at least two 1's (resp., 0's) on the diagonal:
       - 2.2.1.1 all off-diagonal entries are $*$: Lemma 4.6.
       - 2.2.1.2 at least one off-diagonal entry is 0 (resp., 1): Lemmata 4.7, 4.10, and 4.11.
     - 2.2.2 M has exactly one 1 (resp., 0) on the diagonal: Lemmata 4.12 through 4.19. □

Note added in proof. In recent related work the list partition problem on some special classes of graphs [18, 23] and some specific graph partition problems with all parts nonempty [11, 12, 13] have been studied.

REFERENCES

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

[2] A. Brandstädt, *Partitions of graphs into one or two independent sets and cliques*, Discrete Math., 152 (1996), pp. 47–54; Corrigendum, Discrete Math., 186 (1998), p. 295.

[3] A. Brandstädt, V. B. Le, and T. Szymczak, *The complexity of some problems related to graph 3-colorability*, Discrete Appl. Math., 89 (1998), pp. 59–73.
[4] K. Cameron, E. M. Eschen, C. T. Hoàng, and R. Sritharan, *The list partition problem for graphs*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004, ACM, New York, pp. 391–399.
[5] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, *The strong perfect graph theorem*, Ann. of Math. (2), 164 (2006), pp. 51–229.
[6] V. Chvátal, *Star-cutsets and perfect graphs*, J. Combin. Theory Ser. B, 39 (1985), pp. 189–199.
[7] V. Chvátal, *Website on Perfect Graph Problems*, http://athos.rutgers.edu/∼chvatal/perfect/problems.html.
[8] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vusković, *Even-hole-free graphs. I. Decomposition theorem*, J. Graph Theory, 39 (2002), pp. 6–49.
[9] M. Conforti, G. Cornuéjols, A. Kapoor, and K. Vusković, *Even-hole-free graphs. II. Recognition algorithm*, J. Graph Theory, 40 (2002), pp. 238–266.
[10] A. Cournier and M. Habib, *A new linear algorithm for modular decomposition*, in Trees in Algebra and Programming—CAAP' 94, Lecture Notes in Comput. Sci. 787, Springer, Berlin, 1994, pp. 68–84.
[11] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein, *Finding H-partitions efficiently*, Theor. Inform. Appl., 39 (2005), pp. 133–144.
[12] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein, *Extended skew partition problem*, Discrete Math., 306 (2006), pp. 2438–2449.
[13] S. Dantas, C. M. H. de Figueiredo, S. Klein, S. Gravier, and B. A. Reed, *Stable skew partition problem*, Discrete Appl. Math., 143 (2004), pp. 17–22.
[14] C. M. H. de Figueiredo and S. Klein, *The NP-completeness of multipartite cutset testing*, Congr. Numer., 119 (1996), pp. 217–222.
[15] C. M. H. de Figueiredo, S. Klein, Y. Kohayakawa, and B. A. Reed, *Finding skew partitions efficiently*, J. Algorithms, 37 (2000), pp. 505–521.
[16] T. Feder and P. Hell, *List homomorphisms to reflexive graphs*, J. Combin. Theory Ser. B, 72 (1998), pp. 236–250.
[17] T. Feder and P. Hell, *Full constraint satisfaction problems*, SIAM J. Comput., 36 (2006), pp. 230–246.
[18] T. Feder, P. Hell, and W. Hochstattler, *Generalized colourings (matrix partitions) of cographs*, in Graph Theory in Paris, A. Bondy et al., eds., Trends Math., Birkhäuser Verlag, Basel, 2007, pp. 149–167.
[19] T. Feder, P. Hell, and J. Huang, *List homomorphisms and circular arc graphs*, Combinatorica, 19 (1999), pp. 487–505.
[20] T. Feder, P. Hell, and J. Huang, *Bi-arc graphs and the complexity of list homomorphisms*, J. Graph Theory, 42 (2003), pp. 61–80.
[21] T. Feder, P. Hell, and J. Huang, *private communicaton*.
[22] T. Feder, P. Hell, S. Klein, and R. Motwani, *List partitions*, SIAM J. Discrete Math., 16 (2003), pp. 449–478.
[23] T. Feder, P. Hell, S. Klein, L. T. Nogueira, and F. Protti, *List matrix partitions of chordal graphs*, Theoret. Comput. Sci., 349 (2005), pp. 52–66.
[24] T. Feder, P. Hell, D. Král, and J. Sgall, *Two algorithms for general list matrix partitions*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005, ACM, New York, pp. 870–876.
[25] T. Feder, P. Hell, and K. M. Tucker-Nally, *Digraph matrix partitions and trigraph homomorphisms*, Discrete Appl. Math., 154 (2006), pp. 2458–2469.
[26] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104.
[27] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, 1979.
[28] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
[29] R. Hayward and B. A. Reed, *Forbidding holes and antiholes*, in Perfect Graphs, Wiley-Intersci. Ser. Discrete Math. Optim., J. L. Ramirez Alfonsin and B. A. Reed, eds., John Wiley & Sons, Chichester, 2001, pp. 113–137.
[30] P. Hell and J. Nešetřil, *On the complexity of H-coloring*, J. Combin. Theory Ser. B, 48 (1990), pp. 92–110.
[31] Website of the American Institute of Mathematics, *The Perfect Graph Conjecture Workshop*, Oct. 30–Nov. 3, 2002, http://www.aimath.org/WWN/perfectgraph/.

[32] R. M. McConnell and J. P. Spinrad, *Linear time modular decomposition and efficient transitive orientation of comparability graphs*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (Arlington, VA), 1994, ACM, New York, 1994, pp. 536–545.
[33] R. E. Tarjan, *Decomposition by clique separators*, Discrete Math., 55 (1985), pp. 221–232.
[34] A. Tucker, *Coloring graphs with stable cutsets*, J. Combin. Theory Ser. B, 34 (1983), pp. 258–267.
[35] S. Whitesides, *An algorithm for finding clique cutsets*, Inform. Process. Lett., 12 (1981), pp. 31–32.
[36] S. Whitesides, *A method for solving certain graph recognition and optimization problems, with applications to perfect graphs*, Ann. Discrete Math., 21 (1984), pp. 281–297.