Two Algorithms for General List Matrix Partitions

Tomás Feder* Pavol Hell† Daniel Král'‡ Jiří Sgall§

Abstract

List matrix partitions are restricted binary list constraint satisfaction problems which generalize list homomorphisms and many graph partition problems arising, e.g., in the study of perfect graphs. Most of the existing algorithms apply to concrete small matrices, i.e., to partitions into a small number of parts. We focus on two general classes of partition problems, provide algorithms for their solution, and discuss their implications.

The first is an $O(n^{r+2})$-algorithm for the list M-partition problem where M is any r by r matrix over subsets of $\{0, 1\}$, which has the "bisplit property". This algorithm can be applied to recognize so-called k-bisplit graphs in polynomial time, yielding a solution of an open problem from [2].

The second is an algorithm running in time $(rn)^{O(\log r \log n/ \log\log n)} n^{O(\log^2 r)}$ for the list M-partition problem where M is any $r \times r$ matrix over subsets of $\{0, 1, \ldots, q-1\}$, with the "incomplete property". This algorithm applies to all non-NP-complete list M-partition problems with $r = 3$, and it improves the running time of the quasi-polynomial algorithm for the "stubborn problem" from [5], and for the "edge-free three-coloring problem" from [12].

1 Introduction.

1.1 List Matrix Partition Problems. In a *list matrix partition problem*, we have a fixed structure H, consisting of a set $V(H)$ with r elements and q binary relations $E_0(H), E_1(H), \ldots, E_{q-1}(H)$. The structure H can also be viewed as a directed multigraph (with loops allowed) on the vertex set $V(H)$ whose arcs are colored with colors $0, \ldots, q-1$. An instance of the *list matrix partition problem for H* is an arc-colored complete digraph without loops with a list given for each vertex. More precisely, an instance is a structure G consisting of a set $V(G)$ of size $n = |V(G)|$ with a list $L(v) \subseteq V(H)$ for each $v \in V(G)$, with q irreflexive binary relations $E_0(G), \ldots, E_{q-1}(G)$ such that for any two (distinct) vertices $v, v' \in G$, there exists a unique $i = 0, \ldots, q-1$ such that $vv' \in E_i(G)$. The *solution* to the instance is a mapping $\varphi : V(G) \to V(H)$ which is a *list homomorphism* of the structure G to H with respect to the lists L, i.e., it has the following properties:

- $\varphi(v) \in L(v)$ for every vertex $v \in V(G)$, and
- $\varphi(v)\varphi(v') \in E_i(H)$ for all v, v' and i such that $vv' \in E_i(G)$.

In other words, each vertex of G is mapped to a vertex from its list and each arc of G is mapped to an arc of H with the same color. In the sequel, we omit the word "list" if $L(v) = V(H)$ for each vertex v of G. We also consider the *surjective* version of the problem [6, 17], in which the homomorphism $\varphi : V(G) \to V(H)$ is required to be onto $V(H)$.

The *adjacency matrix* of the target structure H with $r = |V(H)|$ is an $r \times r$ matrix M whose elements are subsets of the set $\{0, \ldots, q-1\}$. The element $M_{hh'}$ of the matrix contains an index $i \in \{0, \ldots, q-1\}$ if and only if $hh' \in E_i(H)$. Note that the matrix M need not to be symmetric. The list matrix partition problems for H is called the *list M-partition problem*, where M is the adjacency matrix of H. A list homomorphism φ of G to H with respect to L can also be understood as a partition of the set $V(G)$ into subsets $\varphi^{-1}(h)$, $h \in V(H)$. Such a partition is called a *list M-partition of G*. With this interpretation in mind, the elements of $V(H)$ are called *parts* throughout the paper. Thus the surjective version of the problem requires all parts to be nonempty.

In the most studied case, $q = 2$ with symmetric relations $E_0(H), E_1(H)$ (i.e., the case of symmetric matrices M over the subsets of $\{0, 1\}$), the instances of the list M-partition problem, which are 2-edge-colored complete graphs G, can be viewed as ordinary graphs G' [5, 12, 15, 16]: the edges of the colors 1 and 0

*268 Waverley Street, Palo Alto, CA 94301, USA. E-mail: tomas@theory.stanford.edu.

†School of Computing Science, Simon Fraser University 8888 University Drive, Burnaby, B.C., Canada V5A 1S6. E-mail: pavol@cs.sfu.ca.

‡Institute for Theoretical Computer Science (ITI), Charles University, Malostranské náměstí 25, CZ-11800 Praha 1, Czech Republic. E-mail: kral@kam.mff.cuni.cz. The author has been a postdoctoral fellow at Institute for Mathematics of the Technical University Berlin since October 2004.

§Mathematical Institute, Academy of Sciences of the Czech Republic, Žitná 25, CZ-11567 Praha 1, Czech Republic. E-mail: sgall@math.cas.cz.

in G correspond to the edges and non-edges in G', respectively. Note that we impose some restrictions on the images of both the edges and the non-edges of G'. Hence, graph M-partition problems seek partitions of the graph G' into parts which are complete subgraphs, independent sets, and arbitrary sets, and certain pairs of the parts are completely adjacent or nonadjacent as described by the matrix M. Since any two vertices of G are in one of the relations, all the entries of a matrix M can be required without loss of generality to be nonempty subsets of $\{0, 1\}$. For the sake of brevity, the entries of M which are $\{0\}$, $\{1\}$ and $\{0, 1\}$ are denoted by 0, 1 and $*$, respectively. List homomorphisms of different structures, and graphs in particular, form an important theoretical model for other problems in the area; the reader is referred to a recent monograph [18] for various examples.

In the closely related *binary list constraint satisfaction problems*, we remove the restriction that each pair of distinct vertices of G belongs to a unique relation $E_i(G)$. Again, each structure H defines a separate binary list constraint satisfaction problem. Bulatov [4] has recently classified the complexity of all list constraint satisfaction problems. In particular, it follows from his results that each binary list constraint satisfaction problem is NP-complete or solvable in polynomial time.

1.2 Our Results. Most of the existing algorithms for list M-partition problems focus on small matrices M [5, 8, 6, 17, 15, 16]. Some exceptions are [12, 13, 14, 7]; the algorithms in [13] apply to restricted classes of input graphs, while the algorithms in [7] only apply to one concrete family of matrices M. By contrast, in this short paper we describe two general properties of matrices M, which guarantee the existence of certain list M-partition algorithms.

Our first algorithm is a polynomial-time algorithm for the list M-partition problem for $q = 2$ in the special case where the matrix M has the bisplit property, defined later in Section 2.

This result is motivated by an even more restricted case that includes bisplit and k-bisplit graphs from [2, 3]. A graph G is *bisplit* if the vertices of G can be partitioned into three (possibly empty) independent sets X, Y and Z such that the subgraph of G induced by Y and Z is a complete bipartite graph, i.e., G contains all edges yz for $y \in Y$ and $z \in Z$. In brief, the graph G can be partitioned into an independent set and a complete bipartite graph. In general, a graph G is a said to be *k-bisplit* if it can be partitioned into an independent set and k independent complete bipartite graphs. In other words, the set $V(G)$ can be partitioned into $2k+1$ (possibly empty) independent sets $X, Y_1, Z_1, \ldots, Y_k, Z_k$ such that $yz \in E(G)$ for all pairs $y \in Y_i, z \in Z_i$, and $uv \notin E(G)$ if $u \in Y_i \cup Z_i, v \in Y_j \cup Z_j$ for all $i \neq j$. Observe that a graph G is bisplit if and only if it admits an M-partition for the following matrix M:

$$M = \begin{pmatrix} 0 & * & * \\ * & 0 & 1 \\ * & 1 & 0 \end{pmatrix}$$

Similarly, a graph G is k-bisplit if and only if it admits an M-partition for a certain $(2k+1) \times (2k+1)$ matrix M.

Brandstädt et al. [2] showed that the question of whether there exists an integer k such that G is k-bisplit is NP-complete. For the case $k = 1$, they found an algorithm with running time $O(nm)$ recognizing bisplit graphs [2]. They asked whether k-bisplit graphs for a fixed $k \geq 2$ can also be recognized in polynomial time. We provide such an algorithm running in time $O(n^{2k+1})$; the same bound applies also to the surjective version. Since the problem is NP-complete if k is part of the input, it is very unlikely that there is an algorithm whose running time is polynomial in both n and k. In addition, the surjective version of the problem includes the k-clique problem, which is known to be $W[1]$-complete [9, 10], and therefore one cannot even expect to find an algorithm where the exponent of n would not depend on k [11]. We have recently learned that since the first version [2], Brandstädt et al. [3] have also obtained a polynomial time algorithm for recognizing k-bisplit graphs with a similar running time.

Generalizing notion of k-bisplit graphs, we define (A, B)-bisplit graphs. Let A and B be any $k \times k$ and any $\ell \times \ell$ matrices, respectively, whose entries are 0 and 1. A graph G is said to be *(A, B)-bisplit* if the vertices of G can be partitioned into $k + \ell$ sets $X_1, \ldots, X_k, Y_1, \ldots, Y_\ell$ such that the following two conditions hold:

- If $v \in X_i$ and $v' \in X_j$, then vv' is an edge of G if and only if $A_{ij} = 1$.
- If $v \in Y_i$ and $v' \in Y_j$, then vv' is an edge of G if and only if $B_{ij} = 1$.

Thus, k-bisplit graphs are (A, B)-bisplit graphs where A is the 1×1 matrix whose single entry is 0 and B is the $(2k) \times (2k)$ matrix whose all entries are 0 except for the entries with coordinates $(2i-1, 2i)$ and $(2i, 2i-1)$, $i = 1, \ldots, k$, which are equal to 1. Note that A and B are symmetric matrices in this case, but we do not require them to be symmetric in the general case of (A, B)-bisplit graphs. All the problems of recognition of (A, B)-bisplit graphs are treated as list M-partition problems with $q = 2$ where the $(k + \ell) \times (k + \ell)$ matrix M consists of blocks A and B on the diagonal and $*$ in all the other entries.

The bisplit property, which generalizes all the above notions, is somewhat technical. The exact formulation of it, as well as the statement of our result (and algorithm) in full generality is postponed to Section 2.

Our second algorithm is a subexponential algorithm for the list M-partition problem with general q, such that each relation E_i forbids, for any two triples of parts $B, B' \subseteq V(H)$ (not necessarily disjoint), at least one combination of a part from B with a part from B'. Formally, a matrix M has the *incomplete property*, if for any two three-element subsets B and B' of $V(H)$, and any i, we have $B \times B' \not\subseteq E_i(H)$. We give an algorithm with running time $(rn)^{O(c \log r \log n / \log \log n)} n^{O(\log^2 r)}$ for list-M partition problems for $r \times r$ matrices with the incomplete property. Note that the complexity of our algorithm remains quasi-polynomial even for unbounded r. Also, the second term is irrelevant for r constant (or small compared to n). Actually, our algorithm solves a more general class of problems, the exact statement of which is again postponed to Section 3 where the algorithm is presented. This algorithm also implies that all problems with three parts (i.e., $r = 3$, q arbitrary) have complexity $n^{O(\log n / \log \log n)}$ unless they are NP-complete.

Our motivation for considering this type of problems comes from the classification of Feder and Hell [12] of all list partition problems as having either quasi-polynomial complexity $n^{O(\log n)}$ or being NP-complete. In the special case of list digraph partition problems with three parts (i.e., $q = 2$ and $r = 3$), each problem is either polynomial or NP-complete, as shown by Feder, Hell, and Tucker-Nally [16]. For the special case of list graph partition problems with at most four parts ($q = 2$ and $r \leq 4$), all quasi-polynomial problems turned out to be polynomial, with the exception of a single four-part problem dubbed the *stubborn problem* [5]. A similar three-part problem with $q = 3$ was given in [12], with an $n^{O(\log n)}$ algorithm (again, no polynomial algorithm is known for this problem.) Our algorithm improves the bound for both these partition problems by a factor of $\log \log n$ in the exponent.

2 The Bisplit Property.

In this section, we present our polynomial-time recognition algorithm for (A, B)-bisplit graphs and the generalization to list M-partition problems on graphs for M with the bisplit property defined below.

A *cross* of a matrix of M is a pair formed by some row $M_{i\bullet}$ and the corresponding column $M_{\bullet i}$. We say that the cross i *dominates* the cross j if, for every s, $M_{js} \subseteq M_{is}$ and $M_{sj} \subseteq M_{si}$. (Recall our convention that $*$ is a shortcut for $\{0, 1\}$ and 0 and 1 are shortcuts for the corresponding singletons.) Note that if cross i dominates cross j, then any vertex of G mapped to a j can be also mapped to i. (However, in the surjective case we need to keep one vertex of G in each class.)

Two crosses i and j of a matrix M are said to be *independent* if the sets of $*$'s in the corresponding crosses are incomparable by inclusion. More precisely, there exists s such that $M_{is} = * \neq M_{js}$ or $M_{si} = * \neq M_{sj}$, and there exists s such that $M_{is} \neq * = M_{js}$ or $M_{si} \neq * = M_{sj}$. The matrix M is said to have the *bisplit property* if it does not contain three independent crosses. Clearly, all matrices M corresponding to recognizing (A, B)-bisplit graphs have the bisplit property, as the $*$'s in M consist of two symmetric rectangles and thus at most two crosses can be independent.

Note that for any two crosses i and j, (at least) one of the following three possibilities occurs: (i) the crosses i and j are independent, (ii) or they dominate each other (i.e., i dominates j or j dominates i) or (iii) the crosses i and j are incomparable, where two crosses i and j are said to be *incomparable*, if there exists one s such that M_{is} and M_{js} are incomparable (i.e., one of them is equal to 0 and the other one is equal to 1), or M_{si} and M_{sj} are incomparable.

If we allow three independent crosses, in general, the problem becomes NP-hard. For example, note that a complement of graph G is 3-colorable if and only if it admits an M-partition for

$$M = \begin{pmatrix} 1 & * & * \\ * & 1 & * \\ * & * & 1 \end{pmatrix}$$

We are now prove the main result of this section:

THEOREM 2.1. *Let M be an $r \times r$ matrix over the non-empty subsets of $\{0, 1\}$. If the matrix M has the bisplit property, then the list M-partition problem can be solved in time $O(n^{r+2})$. This also applies to the surjective version of the problem.*

Proof. Given a graph G, we first select at most one representative vertex $v_h \in V(G)$ for each element $h \in V(H)$ and map v_h to h; in the surjective version of the problem we choose exactly one representative for each $h \in V(H)$. Note that there $O(n^r)$ ways in which this can be done.

For each such choice, we delete from the matrix M all the crosses corresponding to the parts in $V(H)$ that do not have a representative, and we delete, from any list $L(v)$, all parts that have no representatives.

Note that the resulting matrix M' still has the bisplit property. Furthermore, whenever the list of a vertex v of G contains i and j such that the cross i dominates the cross j in M', remove j from the list of v. Clearly, if the original list partition problem had a solution, the new one has a solution as well: a solution which maps v to j can be changed so that v is mapped to i. Finally, we remove from the list of each vertex v of G all the parts to which v cannot be mapped because of the structure of M and the preselected representatives. Specifically, if v is adjacent to v_h but $M(h,k) = 0$, then remove k from the list $L(v)$, and if u is nonadjacent to v_h but $M(h,k) = 1$, then remove k from the list $L(u)$. Since the size of H is constant, the restriction described above can be performed in linear time.

We claim that after this procedure, every two crosses $i, j \in L(v)$ are independent, for every $v \in V(G)$. They cannot dominate each other, as dominated crosses have been removed. They also cannot be incomparable, as otherwise there exists a cross s where the crosses i and j are incomparable, and based on the existence of an edge between v and the representative of s from the preselected vertices one of i and j was removed from $L(v)$. As noted above, if two crosses neither dominate one another nor are incomparable, they are independent.

Since M has the bisplit property, $L(v)$ has at most two elements, for each $v \in V(G)$. We may thus assign to each vertex a Boolean variable which represent whether the vertex v is mapped to the first or the second part contained in $L(v)$. If the list contains just a single part, we introduce a corresponding clause of length 1. Given vertices v and v', the parts chosen for v and v' are possibly constrained by M depending on whether v and v' are joined by an edge or not. This constraint gives one or more clauses with the two Boolean variables corresponding to v and v'. The corresponding instance of the 2-satisfiability problem can be solved in time $O(n^2)$ [1]. Since there are $O(n^r)$ choices of the initial partition, the total running time of our algorithm is $O(n^{r+2})$.

If the matrix M corresponds to the recognition problem for (A, B)-bisplit graphs, the running time of the algorithm can be improved by a factor of n^2. The key observation is that we can omit assigning a representative to one part from A and to one part from B.

THEOREM 2.2. *For any matrices A and B with dimensions $k \times k$ and $\ell \times \ell$, respectively, (A, B)-bisplit graphs can be recognized in time $O(n^{k+\ell})$. In particular, for any integer $k \geq 1$, k-bisplit graphs can be recognized in time $O(n^{2k+1})$.*

Proof. First, if the matrix A contains two equal crosses (i.e., such that $A_{is} = A_{js}$ and $A_{si} = A_{sj}$ for all s), remove one of them. We proceed similarly for the matrix B.

Given a set U of crosses of A, call two crosses i and j in A separated by U if $A_{is} \neq A_{js}$ or $A_{si} \neq A_{sj}$ for some $s \in U$. We claim that there exists a $(k-1)$-element set U of crosses of A such that every two crosses i and j in A are separated by U. Observe that the relation of "not being separated by U" is an equivalence on crosses of A for each U and adding elements to U can only refine this equivalence. We start with U empty and sequentially add crosses of A to the current set U. Pick up two crosses i and j of A not separated by U. There exists s such that $A_{is} \neq A_{js}$ or $A_{si} \neq A_{sj}$, since there are no equal crosses in A. Add one such s to U. Clearly, this separates i and j and thus increases the number of equivalence classes by at least one. After at most $k-1$ steps, every two crosses of A are separated and the claim follows.

Similarly, for the matrix B, there exists a set U' of at most $\ell - 1$ crosses that distinguish all crosses of B.

Now follow the previous algorithm with the following modification: if less than k crosses of U should have a representative, we proceed as in the general case. If all crosses of A should have a representative vertex, choose a representative vertices only for the at most $k-1$ crosses in U, but keep the remaining cross(es) available in the restricted matrix M'. Similarly, proceed with B. Remove now the dominated crosses of M' from all the lists $L(v)$. After this, each list contains at most one element from A and at most one element from B; to show this if M contains all the crosses of A or B, we use the fact that chosen U distinguishes all the rows. Solve the corresponding 2-satisfiability problem.

There are $O(n^{k+\ell-2})$ choices of the representatives and the total running time of the algorithm is $O(n^{k+\ell})$.

The statement for k-bisplit graphs immediately follows.

3 The Incomplete Property.

In this section, we present our second algorithm. We actually treat the problem in a more general, multi-domain, setting: An instance consists of n variables x_i that range over respective sets A_i of size at most r. In addition, each pair of distinct variables x_i, x_j is constrained by some $C_{ij} \subseteq A_i \times A_j$ which has the following property: If $B_i \subseteq A_i$ and $B_j \subseteq A_j$ are subsets of size exactly 3, then $B_i \times B_j \not\subseteq C_{ij}$. Clearly, every instance of a list M-partition problem is of the above type if the matrix M has the incomplete property. Indeed, in this case $A_i = V(H)$.

THEOREM 3.1. *Consider instances of an r-part list partition problem with n variables where each variable is constrained by a list A_i and each pair of two distinct variables x_i and x_j is constrained by some relation $C_{ij} \subseteq A_i \times A_j$ with the property that for each $B_i \subseteq A_i$ and $B_j \subseteq A_j$ with $|B_i| = |B_j| = 3$, we have $(B_i \times B_j) \nsubseteq C_{ij}$. This r-part list partition problem can be solved in time $(rn)^{O(\log r \log n/\log\log n)} n^{O(\log^2 r)}$.*

Proof. Let $t = \log n/\log\log n$. If $t \leq 6$, then n is constant and the problem can be solved by brute force in time bounded by a polynomial in r.

The algorithm proceeds by double recursion. The outer recursion decreases r by a factor of $2/3$ in each level. More precisely, let p be the maximal size of any list $|A_i|$ at the beginning of a level of the outer recursion. Then at the end of the level, in all the calls to the next level of the outer recursion, the size of all the lists $|A_i|$ is at most $\lfloor 2(p+1)/3 \rfloor$. Consequently, after $O(\log r)$ levels, the size of the lists is at most 2. Such instances are instances of the 2-satisfiability problem and can be solved in time $O(n^2)$.

List A_i is said to be *large* if $|A_i| > 2(p+1)/3$, where p is the maximal size of any list $|A_i|$ at the beginning of the current level of the outer recursion. The inner recursion gradually decreases the number of variables with large lists.

Let s denote the number of large lists; note that $s \leq n$. We define $f(s)$ to be the maximal number of calls within one level of the outer recursion taken over all instances with at most s large lists. The value $f(s)$ also implicitly depends on r and n which are fixed for the whole proof. If $s = 1$, assign all possible values of the last large list to the corresponding variable and recurse to the next level of the outer recursion; thus $f(1) \leq n$.

For each i and $a \in A_i$, let $g_i(a)$ be the number of large lists A_j such that there more than $|A_j|/3 - 1$ elements $d \in A_j$ with $ad \notin C_{ij}$. Thus, if x_i is assigned a, the size of at least $g_i(a)$ large lists is decreased to less than $|A_j| - |A_j|/3 + 1 \leq 2p/3 + 1$, i.e., they are no longer large, and s decreases to at most $s - g_i(a)$.

We claim that for any i and any three distinct $a, b, c \in A_i$, we have $g_i(a) + g_i(b) + g_i(c) \geq s - 1$. Otherwise there exists one large list A_j, $j \neq i$, such that there are at most $|A_j|/3 - 1$ elements $d \in A_j$ with $ad \notin C_{ij}$, at most $|A_j|/3 - 1$ elements $d \in A_j$ with $bd \notin C_{ij}$, and at most $|A_j|/3 - 1$ elements $d \in A_j$ with $cd \notin C_{ij}$. Thus there exists a three-element subset $B \subseteq A_j$ such that $\{a, b, c\} \times B \subseteq C_{ij}$, contradicting the incompleteness assumption.

We now distinguish two cases according to the number of elements $a \in A_i$ with $g_i(a) \leq s/t$.

Assume first that each list A_i contains at most one element a_i with $g_i(a_i) \leq s/t$. Assign to each x_i the element $a_i \in A_i$ with the smallest $g_i(a_i)$ and check whether this is a valid assignment. If yes, we are done with no recursion. If no, there must exist i and j such that $a_i a_j \notin C_{ij}$; fix one such pair i, j. Recurse on $|A_i| - 1$ problems obtained by assigning to x_i all elements $c_i \in A_i \setminus \{a_i\}$ and on $|A_j| - 1$ problems obtained by assigning to x_j all elements $c_j \in A_j \setminus \{a_j\}$. Since $a_i a_j \notin C_{ij}$, this covers all possible solutions. We now bound the number of recursive calls. Observe that $g_i(b_i) < (s-1)/3$ for at most one value of $b_i \in A_i \setminus \{a_i\}$, as otherwise we would get, together with a_i, three elements of A_i with values of g_i smaller than $(s-1)/3$, contradicting the previous claim. Furthermore, for this value of b_i we have $g_i(b_i) > s/t$. Using the same argument for j, we have at most two recursive calls with s decreased to at most $s - s/t$ and at most $2r$ calls with s decreased to at most $(2s+1)/3$. Consequently, the number of calls to the next level of the outer recursion is at most

$$2r \cdot f\left(\frac{2s+1}{3}\right) + 2 \cdot f\left(s\left(1 - \frac{1}{t}\right)\right).$$

The remaining case is that there is a list A_i with two elements $a, b \in A_i$ such that $g_i(a) \leq s/t$ and $g_i(b) \leq s/t$. By the above claim, each element $c \in A_i \setminus \{a, b\}$ has $g_i(c) \geq s(1 - 2/t) - 1$. We now form $|A_i| - 1$ recursive calls. First we assign to x_i all elements $c \in A_i \setminus \{a, b\}$ and recurse on the remaining instance with at most $1 + 2s/t$ large lists. If none of these instances has a solution, we can restrict the list A_i to $\{a, b\}$ and eliminate x_i as follows: For every two indices $j, j' \neq i$ and each $c \in A_j$ and $c' \in A_{j'}$ such that $ca \notin C_{ji}$ and $bc' \notin C_{ij'}$, remove cc' from $C_{jj'}$ (if $cc' \in C_{jj'}$). Clearly, the new instance (without the variable x_i) has a solution if and only if the original one has a solution with $x_i \in \{a, b\}$. The new instance is solved by a recursive call; note that it has at most $s - 1$ large lists. The number of calls to the next level of the outer recursion is at most

$$r \cdot f\left(1 + \frac{2s}{t}\right) + f(s-1).$$

Unwinding the last call of the recursion in the second case until the first case occurs we obtain the following bound for $f(s)$:

$$\begin{aligned} f(s) \quad &\leq \quad rn \cdot f\left(1 + \frac{2s}{t}\right) \\ &\qquad + 2r \cdot f\left(\frac{2s+1}{3}\right) + 2 \cdot f\left(s\left(1 - \frac{1}{t}\right)\right). \end{aligned}$$

Solving this recurrence we obtain, for a suitable constant C, and using $s \leq n$ and $t = \log n/\log\log n$ in the second inequality,

$$
\begin{aligned}
f(s) &\leq (rn)^{C \log_t s}(2r)^{C \log s} 2^{Ct \log s} \\
&\leq (rn)^{O\left(\frac{\log n}{\log \log n}\right)} \cdot n^{O(\log r)}
\end{aligned}
$$

Since there are $O(\log r)$ levels of the outer recursion and all the other computations are polynomial-time for each recursive call, the overall number of recursive calls and the running time of the presented algorithm are $(rn)^{O(\log r \log n/ \log\log n)} n^{O(\log^2 r)}$ as claimed in the theorem.

Given a three-part list partition problem, if one of the q binary constraints allows all pairs, then the problem is just a list constraint satisfaction problem, and thus polynomial or NP-complete by Bulatov's classification [4]. If each of the q binary constraints forbids at least one pair, then the problem satisfies assumptions of Theorem 3.1. We thus have the following:

Corollary 3.1. *Every three-part list partition problem has either complexity at most* $n^{O(\log n/ \log\log n)}$ *or is NP-complete.*

The complexity of the problems from the preceding Corollary is either polynomial or NP-complete in the special case of list digraph partition problems as shown in [16], and noted in the introduction.

The graph list partition problems with at most four parts ($q = 2$, $r \leq 4$) were classified as polynomial or NP-complete, with the exception of the so-called *stubborn problem*, which is quasi-polynomial [5]. This is the problem with parts $0, 1, 2, 3$ where one symmetric binary relation forbids just the loop at 0, and the other symmetric binary relation forbids only the loops at 2 and 3, and the edge 13. As 1 dominates 3, the lists may be assumed to not contain both 1 and 3, i.e., the given lists are contained in $\{0, 1, 2\}$ or $\{0, 2, 3\}$. We thus have a four-part partition problem without the relation that allows all combinations of two choices of three parts: on $\{0, 1, 2\}$ one relation forbids 00 and the other 22, on $\{0, 2, 3\}$ one relation forbids 00 and the other 33, and between $\{0, 1, 2\}$ and $\{0, 2, 3\}$ one relation forbids 00 and the other 13. Theorem 3.1 yields the following bound.

Corollary 3.2. *The stubborn problem can be solved by an algorithm with running time* $n^{O(\log n/ \log\log n)}$.

The *edge-free three-coloring problem* from [12] is closely related to the stubborn problem (cf. [12]), but has only three parts and $q = 3$; it can be formulated as follows. An instance of the edge-free three-coloring problem is a complete graph with edges colored by $0, 1, 2$. The solution to such an instance is a three-coloring of the vertices of the complete graph without a monochromatic edge, i.e., an edge of color i having both end vertices colored i. (Note that the analogous *edge-free two-coloring problem* is the recognition problem for the class of split graphs, and hence solvable in polynomial time; the *edge-free four-coloring problem* is easily seen to be NP-complete [12].) Clearly, the edge-free three-coloring problem is a restriction (to the case where all lists are $\{0, 1, 2\}$) of the list M-partition problem with the following matrix M (we again let $*$ denote $\{0, 1, 2\}$):

$$
M = \begin{pmatrix} \{1,2\} & * & * \\ * & \{0,2\} & * \\ * & * & \{0,1\} \end{pmatrix}
$$

Theorem 3.1 also yields the following bound (which also applies to the general list M-partition problem with the above M).

Corollary 3.3. *The edge-free three-coloring problem can be solved by an algorithm with running time* $n^{O(\log n/ \log\log n)}$.

Acknowledgments.
Part of this research was conducted during the Dagstuhl seminar on Graph Colorings in September 2003 in which the last three authors took part. They would like to thank the International Conference and Research Center Schloss Dagstuhl for support from the program High Level Scientific Conferences of the European Union. The first three authors acknowledge support from an NSERC Discovery Grant. The third author would also like to thank Ladislav Stacho from Simon Fraser University for his kind hospitality and support during his visit at SFU in May 2004. The last author acknowledges support by the Institute for Theoretical Computer Science, Prague, and grant IAA1019401 of GA AV ČR. Institute for Theoretical Computer Science in Prague is supported by the Ministry of Education of the Czech Republic as project LN00A056.

References

[1] B. Aspvall, M. R. Plass, and R. E. Tarjan, *A linear-time algorithm for testing the truth of certain quantified Boolean formulas*, Information Processing Letters 8 (1979), pp. 121–123.

[2] A. Brandstädt, P. L. Hammer, V. Bang Le, and V. V. Lozin, *Bisplit graphs*, DIMACS Technical Report 2002-44 and RUTCOR Research Report 2002-28, Rutgers University, Piscataway, NJ, 2002.

[3] A. Brandstädt, P. L. Hammer, V. Bang Le, and V. V. Lozin, *Bisplit graphs*, to appear in Discrete Mathematics.

[4] A. A. Bulatov, *Tractable conservative constraint satisfaction problems*, in: Proc. 18th IEEE Symposium on Logic in Computer Science (LICS) 2003, pp. 321–330.

[5] K. Cameron, E. E. Eschen, C. T. Hoang, and R. Sritharan, *The list partition problem for graphs*, in: Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2004, pp. 391–399.

[6] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein, *On H-partition problems*, to appear in RAIRO - Theoretical Informatics and Applications. Tech. report PESC UFRJ ES-579/02, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil, 2002.

[7] S. Dantas, C. M. H. de Figueiredo, S. Gravier, and S. Klein, *Extended skew partition problems*, to appear in Discrete Mathematics.

[8] S. Dantas, C. M. H. de Figueiredo, S. Gravier, S. Klein, and B. Reed, *Stable skew partition problem*, Discrete Applied Mathematics 143 (2004), pp. 17–22.

[9] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness II: On completeness for $W[1]$*, Theoretical Computer Science 141 (1995), pp. 109–131.

[10] R. G. Downey and M. R. Fellows, *Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy*, in: Complexity Theory: Current Research, Cambridge University Press (1993), pp. 191–226.

[11] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, NY, 1999.

[12] T. Feder and P. Hell, *List constraint satisfaction and list partition*, submitted.

[13] T. Feder and P. Hell, *Matrix partitions of perfect graphs*, submitted.

[14] T. Feder, P. Hell, and J. Huang, *Bi-arc graphs and the complexity of list homomorphisms*, J. Graph Theory 42 (1999), pp. 61–80.

[15] T. Feder, P. Hell, S. Klein, and R. Motwani, *List partitions*, SIAM J. Discrete Math. 16 (2003), pp. 449–478.

[16] T. Feder, P. Hell, and K. Tucker-Nally, *List partitions and trigraph homomorphisms*, submitted.

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

[18] P. Hell and J. Nešetřil, *Graphs & homomorphisms*, Oxford University Press, Oxford, UK, 2004.