Counting Partitions of Graphs

Pavol Hell[1], Miki Hermann[2], and Mayssam Mohammadi Nevisi[1]

[1] School of Computing Science, Simon Fraser University, Burnaby, Canada, `pavol,maysamm@sfu.ca`[⋆]

[2] LIX CNRS UMR 7161, École Polytechnique, Palaiseau, France. `hermann@lix.polytechnique.fr`[⋆⋆]

Abstract. Recently, there has been much interest in studying certain graph partitions that generalize graph colourings and homomorphisms. They are described by a pattern, usually viewed as a symmetric $\{0, 1, *\}$-matrix M. Existing results focus on recognition algorithms and characterization theorems for graphs that admit such M-partitions, or M-partitions in which vertices of the input graph G have lists of admissible parts. In this paper we study the complexity of counting M-partitions. The complexity of counting problems for graph colourings and homomorphisms have been previously classified, and most turned out to be #P-complete, with only trivial exceptions where the counting problems are easily solvable in polynomial time. By contrast, we exhibit many M-partition problems with interesting non-trivial counting algorithms; moreover these algorithms appear to depend on highly combinatorial tools. In fact, our tools are sufficient to classify the complexity of counting M-partitions for all matrices M of size less than four. It turns out that, among matrices not acccounted for by the existing results on counting homomorphisms, all matrices which do not contain the matrices for independent sets or cliques yield tractable counting problems.

Keywords: partitions, polynomial algorithms, #P-completeness, dichotomy, counting problems

1 Introduction

It is well known that the number of bipartitions of a graph G can be computed in polynomial time. Indeed, we can first check, in polynomial time, if G is bipartite, and if not, the answer is 0. If G is bipartite, we can find, in polynomial time, the number c of connected components of G. Since each such component admits exactly two bipartitions, the answer in this case is 2^c. Interestingly, the number of bipartitions can also be counted using linear algebra: if each vertex v is associated with a variable x_v over the field $F_2 = \{0, 1\}$, and each edge uv with the equation $x_u + x_v = 1$ in F_2 (i.e., modulo 2), then the number solutions of this system is precisely the number of bipartitions of G. Thus 2-colourings of graphs can

[⋆] Supported by NSERC Canada

[⋆⋆] Supported by ANR Blanc International ALCOCLAN be counted in polynomial time. It is also known that the counting problem for m-colourings of G with $m > 2$ is #P-complete [5]. This is the *dichotomy* of the counting problems for graph colourings.

A *homomorphism* f of G to H is a mapping $V(G) \to V(H)$ such that $uv \in E(G)$ implies $f(u)f(v) \in E(H)$. If $H = K_m$, a homomorphism of G to H is an m-colouring of G. Dichotomy of counting homomorphisms to graphs H has been established by Dyer and Greenhill [6]. Namely, if each connected component of H is either a reflexive complete graph, or an irreflexive complete bipartite graph, then counting homomorphisms to H can be solved by trivial methods, as in the above example (or, once again, by linear algebra). In all other cases, counting homomorphisms to H is #P-complete [6]. Other dichotomies for homomorphism counting problems, in bounded degree graphs, or for homomorphisms with lists, are discussed in [18]. (In the list version of the problem, the graph G has a list $L(v) \subseteq V(H)$ for each vertex $v \in V(G)$ and only homomorphisms f that satisfy $f(v) \in L(v)$, for all $v \in V(G)$, are counted.) In particular, it is proved in [18], that, as without lists, if each connected component of H is either a reflexive complete graph, or an irreflexive complete bipartite graph, then counting list homomorphisms to H can be solved by easy polynomial time methods, and in all other cases counting list homomorphisms to H is #P-complete [18].

Homomorphisms to H can be viewed as partitions of the input graph G into parts corresponding to the vertices of H. Specifically, if $x \in V(H)$ has no loop, the corresponding part P_x is an independent set in G, and if $xy \notin E(H)$, then there are no edges between the parts P_x and P_y. A further generalization of homomorphisms allows us to specify that certain parts P_x must be cliques, and between certain parts P_x and P_y there must be all possible edges.

Throughout the paper, M will always be assumed to be a symmetric m by m matrix over $0, 1, *$. An *M-partition* of a graph G is a partition $P_1, P_2, \ldots, P_m$ of $V(G)$, such that two distinct vertices in (possibly equal) parts P_i and P_j are adjacent if $M(i,j) = 1$, and nonadjacent if $M(i,j) = 0$; the entry $M(i,j) = *$ signifies no restriction. Since we admit $i = j$, a part P_i is independent if $M(i,i) = 0$, and a clique if $M(i,i) = 1$. (We usually refer to P_i as *the i-th part.*) Note that when M has no 1's, the matrix M corresponds to an adjacency matrix of a graph H, if we interpret $*$ as adjacent and 0 as non-adjacent; and in this case an M-partition of G is precisely a homomorphism of G to H. Thus M-partitions generalize homomorphisms and hence also graph colourings. They are frequently encountered in the study of perfect graphs. A simple example is the matrix $M = \left(\begin{smallmatrix} 0 & * \\ * & 1 \end{smallmatrix}\right)$: in this case a graph G is M-partitionable if and only if it is a split graph. Other examples of matrices M such that M-partitions are of interest in the study of perfect graphs can be found in [8]. They include the existence of a homogeneous set [13] (M has size three, see the next section), the existence of a clique cutset (M has size three), the existence of a skew cutset (M has size four), and many other popular problems [3, 8, 10, 15, 22]. (If M has a diagonal $*$, it is usual for the existence problems to focus on M-partitions with all parts non-empty, otherwise the problems become trivial; this is in particular the case in the previous three examples.) In any event, we emphasize the fact that M-partition problems tend to be difficult and interesting even for small matrices M.

We note for future reference the following *complementarity* of M-partitions. Denote by $\overline{M}$ the matrix obtained from M be replacing each 0 by 1 and vice versa. Then an $\overline{M}$-partition of a graph G is precisely an M-partition of the complement $\overline{G}$.

In the literature there are several papers dealing with algorithms and characterizations of graphs admiting M-partitions (or list M-partitions) [4, 7, 8, 10]; these are detailed in a recent survey [14], cf. also a slightly older survey [19], or the book [16]. We focus on the counting problem for M-partitions. Recall that for homomorphism problems, except for trivial cases, counting homomorphisms turned out to be #P-complete [6]. More generally, in constraint satisfaction problems [9, 20], the situation is similar, and only the counting problems that can be solved using algebraic methods turned out to be tractable [1]. We contrast these facts by exhibiting several counting problems for M-partitions, where highly combinatorial methods seem to be needed. In the process, we completely classify the complexity of counting the number of M-partitions for all matrices of size less than four.

Given a matrix M, we want to know how hard it is to count the number of M-partitions. As a warm-up, we prove the following classification for two by two matrices $M = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$.

Theorem 1. *If c and exactly one of a, b is $*$, then the problem of counting the number of M-partitions is* #P*-complete. Otherwise there is a polynomial algorithm to count the number of M-partitions.*

Proof. If, say, $a = c = *$ and $b = 0$, then the number of M-partitions of G is precisely the number of independent sets in G, which is known to be #P-complete [21]. Similarly, if $a = c = *$ and $b = 1$, we are counting the number of cliques in G, which is also #P-complete [21] (or by complementarity).

If M has no 1, the result follows by [6], so we assume that M contains at least one 1, and, by complementarity, also at least one 0. If $c = 0$, the two parts have no edges joining them, and at least one part is a clique. The number of M-partitions can easily be determined once the connected components of G have been computed. (For instance if $a = 1$, $b = *$, and t connected components of G are cliques, then G has t M-partitions. When $a = 1$, $b = 0$, the counting is even easier.) If $c = 1$, the result follows by complementation.

Thus we assume that $c = *$. By symmetry, we may assume without loss of generality that $a = 0$ and $b = 1$. Such an M-partition of G is called a *split partition.* It follows from [8] that the number of split partitions is polynomial, and can be found in polynomial time (see Theorem 3.1 in [8]). Therefore they can also be counted in polynomial time. $\square$

The two matrices $\left(\begin{smallmatrix} * & * \\ * & 0 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} * & * \\ * & 1 \end{smallmatrix}\right)$ from the first paragraph of the proof will play a role in the sequel, and we will refer to the as the *matrices for independent sets, and cliques.*

2 Decomposition Techniques

The last two by two matrix, corresponding to split partitions, illustrates the fact that there are interesting combinatorial algorithms for counting M-partitions. In this section we examine a few other examples, in this case of three by three matrices, documenting this fact.

In the remainder of the paper, we assume we have the matrix $M = \begin{pmatrix} a & d & e \\ d & b & f \\ e & f & c \end{pmatrix}$.

We begin by discussing some cases related to the example of split partitions at the end of the previous section. In fact, the general technique of Theorem 3.1 from [8] is formulated in the language of so-called *sparse-dense partitions.* If $\mathcal{S}$ and $\mathcal{D}$ are two families of subsets of $V(G)$, and if there exists a constant c such that all intersections $S \cap D$ with $S \in \mathcal{S}, D \in \mathcal{D}$, have at most c vertices, then G with n vertices has at most n^{2c} sparse-dense partitions $V(G) = S \cup D$, with $S \in \mathcal{S}, D \in \mathcal{D}$, and they can be generated in polynomial time [8]. For split partitions, we take $\mathcal{S}$ to be all independent sets, $\mathcal{D}$ all cliques, and $c = 1$. If we take for $\mathcal{S}$ all bipartite induced subgraphs, for $\mathcal{D}$ all cliques, and $c = 2$, we can conclude that any graph G has only a polynomial number of partitions $V(G) = B \cup C$, where B induces a bipartite graph and C induces a clique, and all these partitions can be generated in polynomial time. This result is sufficient to cover a number of polynomial cases.

Theorem 2. *If a, b, c are not all the same and none is $*$, then the number of M-partitions can be counted in polynomial time.*

Proof. Up to symmetry and complementarity we may assume that $a = b = 0, c = 1$. For each sparse-dense partition $V(G) = S \cup D$, we shall test how many partitions of the subgraph induced by S, into the first part and the second part, satisfy the constraints induced by the entries d, e, and f of the matrix M. We impose the constraints due to e and f by introducing lists on the vertices in S. (If, say, $e = 1$, then only vertices completely adjacent to D will have the third part in their lists, and similarly for other values of e and f.) If d is 0 or $*$, this corresponds to counting list homomorphisms to a complete bipartite graph (K_2), which is polynomial by [18], as noted above. When $d = 1$, there are at most two different partitions of S to consider, so we can check these as needed. $\square$

The next result deals with a class of problems related to homogeneous sets and modular decomposision [13]. A *module* (or a *homogeneous set*) in a graph G is a set $S \subseteq V(G)$ such that every vertex not in S is either adjacent to all vertices of S or to none of them. Trivially, each singleton vertex, as well as $V(G)$ and $\emptyset$, are modules. For most applications, these trivial modules are ignored, but we will be counting all modules. In fact, we will show that modules can be counted in polynomial time, even under some additional restrictions.

For our purposes, we will only use the following basic theorem of Gallai [11].

Theorem 3 ([11]). *For any graph G one of the following three cases must occur.*

1. *G is disconnected, with components $G_1, G_2, \dots G_k$.*
   Each union of the sets $V(G_i)$ is a module of G, and the other modules of G are precisely all the modules of individual components G_i.
2. *The complement of G is disconnected, with components $H_1, H_2, \dots, H_\ell$.*
   Each union of the sets $V(H_j)$ is a module of G, and the other modules of G are precisely all the modules of individual subgraphs $\overline{H_j}$.
3. *Both G and its complement are connected. There is a partition $S_1, S_2, \dots, S_r$ of $V(G)$ (which can be computed in linear time), such that all the modules of G are precisely all the modules of individual subgraphs induced by the sets S_t, $t = 1, \dots, r$, plus the module $V(G)$.*

Based on this theorem, one can recursively decompose any graph into modules; this decomposition produces a tree structure called the *modular decomposition tree* of G. Even though the number of modules of G can be exponential (for instance if $G = K_n$), the modular decomposition tree has polynomial size and can be computed in linear time [12].

We will count modules of G, or modules of G with special properties, recursively, using the theorem. This will suffice to provide a polynomial time counting algorithm for these modules. We note however, that there is a natural linear time algorithm that counts these modules directly on the modular decomposition tree. We will describe this algorithm in the full journal version of our paper.

We will call the sets $V(G_i)$, $V(\overline{H_j})$, and S_t from the theorem the *blocks* of G. According to the theorem, all modules are modules of the blocks, except for modules that are unions of (at least two) blocks. We call these latter modules *cross modules.*

We illustrate the technique on a polynomial time algorithm to count the total number $T(G)$ of non-empty modules of a graph G. (This turns out to be more convenient, and one can add 1 for the empty module at the end of the computation.) We first compute the decompositions (1, 2, or 3) in Theorem 3. In cases 1 and 2, we have $T(G) = 2^t - t - 1 + \sum T(B)$, where t is the number of blocks, and the sum is over all blocks B. In case 3, we have $T(G) = 1 + \sum T(B)$. Indeed the number of cross modules is 1 in the case 3 (only the module $V(G)$ is a cross module), and is $2^t - t - 1$ in the other two cases (subtracting one for the empty set and the individual blocks). Since the sizes of the blocks for the recursive calls sum up to n, this yields a recurrence for the running time whose solution in polynomial in n. This shows that counting the number of modules of G is polynomial.

We note that a number of variants can be counted the same way. Consider, for instance, the number of modules that are independent sets. In case 1, if s of the blocks consist of a single vertex, then the number of cross modules changes to $2^s - s - 1$. In case 2, as well as 3, there are no cross module in this case (unless G has no edges). Moreover, it is easy to see that the number of non-empty independent modules of G inside a block B is precisely the number of non-empty independent modules of the graph induced by B, in all three cases. Thus the number of independent non-empty modules of G is $T(G) = 2^s - s - 1 + \sum T(B)$ in case 1, and $T(G) = \sum T(B)$ in cases 2 and 3. Of course, the number of modules that are cliques can be counted in a similar way, or by looking at the complement.

We can in fact handle all restrictions of this type on the module S, its set of neighbours R, and its set of non-neighbours Q. The above examples are respectively: (i) S, R, Q unrestricted; (ii) S independent, R, Q unrestricted. It turns out that all the remaining combinations of restrictions (independent, clique, or unrestricted) for the sets S, Q, R can be treated in similar ways.

Note that the arguments are written so as to count the number of non-empty modules of the various kinds. Of course by adding 1, we can count all such modules.

Theorem 4. *There is a polynomial time algorithm to count the number of modules satisfying any combination of restrictions where the module itself, its set of neighbours, and its set of non-neighbours are an independent set, a clique, or unrestricted.*

It is easy to see that each restricted kind of module corresponds to an M-partition in which $d = 1, e = 0, f = *$, and a is determined by the constraint on S ($a = 0$ if R is to be independent, $a = 1$ if it is to be a clique, and $a = *$ if S is unrestricted), b is determined by the constraint on R, and c by the constraint on Q.

Corollary 5. *If $d = 1, e = 0, f = *$, then the number of M-partitions with non-empty first part can be counted in polynomial time.*

However, the number of M-partitions must also take into account the partitions with the first part empty. By applying Theorem 1, we conclude that the number of M-partitions with empty first part can also be counted in polynomial time, unless the second and third part form the matrix for independent sets or cliques.

Theorem 6. *If d, e, f are all different, and M does not contain, as a principal submatrix, the matrix for independent sets or cliques, then the number of M-partitions can be counted in polynomial time.*

The last kind of decomposition refers to the matrix M. Namely, if two of d, e, f are 0, then the matrix M can be viewed as consisting of two submatrices and the M-partition problem can be reduced to the corresponding problems for these two matrices.

Theorem 7. *Assume that two of d, e, f are 0 and M does not contain as principal submatrix the matrix for independent sets or cliques. Then counting the number of M-partitions is polynomial.*

3 A special polynomial case

Our final example of a polynomial counting problem deals with the following matrix $M = \begin{pmatrix} 0 & * & * \\ * & 0 & 1 \\ * & 1 & 0 \end{pmatrix}$.

We first consider bipartite input graphs G.

Theorem 8. *The number of M-partitions of bipartite input graphs G can be computed in polynomial time.*

Proof. (*Sketch*) Let G be a bipartite graph with parts X and Y. We again call the three parts of an M-partition A, B, C, in that order. Each part X, Y is an independent set, and thus, must be placed either entirely in $A \cup B$ or entirely in $A \cup C$. The number of M-partitions of G with B or C is empty is easily counted, according to Theorem 1. We will add the number of M-partitions with the vertices of X placed in $A \cup B$, and the vertices of Y placed in $A \cup C$ and the number of M-partitions with the opposite assignment (and subtract 1 when G has no edges, because in that case we are counting the one possible solution that places all vertices to A twice.) Thus consider the M-partitions of G with the vertices of X placed in $A \cup B$, and the vertices of Y placed in $A \cup C$: a vertex of $X \cap A$ has no neighbours in B and a vertex of $Y \cap A$ has no neighbours in C. Thus G has some possible edges between $A \cap X$ and $C \cap Y$, all edges between $C \cap Y = C$ and $B \cap Y = B$, and some possible edges between $B \cap Y$ and $A \cap Y$, and no other edges. Such a partition is called a *split* [2]. (In general graphs splits can have edges inside the parts, in our case of bipartite graphs, the parts are independent sets.) Each M-partition of G gives a split, and each split corresponds to two unique M-partitions of G. Thus it will suffice to count the number of splits. Splits form a recursive structure called a *split decomposition tree* [2], akin to the modular decomposition tree discussed earlier. Even though the number of splits can be exponential, the split decomposition tree has polynomially many vertices, and can be computed in linear time [2]. It can be shown that the number of splits of a graph can be computed in linear time from the split decomposition tree. □

We are ready to prove the main result of this section, Theorem 9.

Theorem 9. *The number of M-partitions of any graph G can be computed in polynomial time.*

Proof. By Theorem 8, we may assume that G is not bipartite. If it does not contain a triangle, then the shortest odd cycle has at least five vertices. In such a case, the number of M-partitions of G is zero. Indeed, the largest complete bipartite subgraph of the cycle has three (consecutive) vertices, and hence at least two adjacent vertices of the cycle must be placed in A in any M-partition of G, which is impossible.

Otherwise, we find a triangle uvw in G, in polynomial time, and then add the numbers of M-partitions of G with the six possible assignments of u, v, w to A, B, C. (Since the parts are independent, u, v, w must be placed in distinct parts.) For each such assignment, we shall first extend the assignment by placing vertices that are forced to certain parts uniquely.

In the first phase, we proceed as follows:

– a vertex with neighbours in two different parts is placed in the third part;

- a vertex with a non-neighbour in B as well as a non-neighbour in C is placed in A;
- a vertex with both a neighbour and a non-neighbour in B (respectively C) is placed in A.

It is clear that these are forced assignments in any M-partition of G extending the given assignment on u, v, w. If at any time these assignments (or those below) violate the requirements of an M-partition, the corresponding count is zero.

After the first phase, every vertex is either fully adjacent to B and not adjacent to any vertex of $A \cup C$, forming a set called X, or is fully adjacent to C and not adjacent to any vertex of $A \cup B$, forming a set called Y. Note that the vertices of X must be placed in $A \cup C$ and the vertices of Y in $A \cup B$.

In the second phase, we extend the assignment using the following rules. (In brackets we explain why these rules are forced.)

Assume uv is an edge with $u, v \in X$, and $w \in Y$. (Symmetric rules apply for $u, v \in Y$, and $w \in X$.)

- If w is adjacent to both u and v then w will be placed in B. (This is forced because u and v must be in different parts A and C, and w is adjacent to both of them.)
- if w is nonadjacent to both u and v, then w is placed in A. (This is forced because u and v must be in different parts A and C, and w is not adjacent to either of them, so it is nonadjacent to at least one vertex in C.)
- if w is adjacent to u but not to v, then u is placed in C and v is placed in A. (This is forced because if $v \in C$ then $u \in A$ and $w \in B$, which is impossible as vw is not an edge of G.)

When there are no more choices remaining, either we have X or Y empty, or the graph induced by $X \cup Y$ is bipartite. The former case corresponds to partitions of the remaining vertices into two of the three parts, counted by Theorem 1. The latter case is counted by Theorem 8. $\square$

4 Dichotomy

Our main result is the following dichotomy. Notice that when M does not contain any 1's (or does not contain any 0's), then the dichotomy follows from [6].

Theorem 10. *Suppose M is an m by m matrix with $m < 4$, and assume M contains both a* 0 *and a* 1.

If M contains, as a principal submatrix, the matrix for independent sets, or the matrix for cliques, then the counting problem for M-partitions is #P-*complete.*

Otherwise, counting M-partitions is polynomial.

Proof. We first derive the polynomial cases from our existing results. We assume throughout that M contains both a 0 and a 1.

For $m = 2$ this follows from Theorem 1. Thus we assume that $m = 3$, say $M = \begin{pmatrix} a & d & e \\ d & b & f \\ e & f & c \end{pmatrix}$, and M does not contain as a principal submatrix the matrix for independent sets, or the matrix for cliques. If at least two of d, e, f are 0 (or at least two are 1), then M-partitions are counted by Theorem 7. If d, e, f are all different, then the number of M-partitions are counted by Theorem 6.

Hence, in all the remaining cases, we may assume that $d = e = *$, $f \neq 0$ by symmetry and complementarity. Notice that M contains both a 0 and a 1; hence, $f = 1$ implies that at least one of a, b, c is 0, and $f = *$ implies that at least one of a, b, c is 0 and one is 1. Thus, we note that none of a, b, c is $*$, because M does not contain the forbidden principal submatrices, and we conclude by Theorem 9 ($f = 1$, $a = b = c = 0$) or Theorem 2 (otherwise).

It remains to show that if M contains, as a principal submatrix, the matrix for independent sets or cliques, then counting M-partitions is #P-complete. We shall again use the notation $M' = \left(\begin{smallmatrix} b & f \\ f & c \end{smallmatrix}\right)$. We shall assume that M' is the matrix for independent sets, without loss of generality, say, that $b = f = *$, $c = 0$.

We consider two distinct cases, depending on the value of a. Our proof is completed by the following two lemmas. □

For the purposes of the lemmas we introduce two constructions. The *universal vertex extension* G^* of a graph G is a graph obtained from G by adding a new vertex u, adjacent to all vertices of G. The *isolated vertex extension* G^o of G is a graph obtained from G by adding a new isolated vertex u.

Lemma 11. *If $a \neq 0$, $b = f = *$, $c = 0$, then counting the number of M-partitions is* #P*-complete.*

Proof. In this case, we reduce from the number of independent sets in graph G using the isolated vertex extension of G. Let $\#I(G)$ denote the number of independent sets of G, and $\#M(G)$ the number of M-partitions of G. As before, we will write A, B, C for the first, second, and third part of an M-partition.

Given an input graph G, we first construct G^o. We count the number of M-partitions of G^o according to the placement of the isolated vertex u. When $a = 1$, we consider the two following cases:

We illustrate the proofs on the case when d, e are different from 1. In this case we can show that $\#M(G^o) = \#I(G) + 2\#M(G)$. This implies that counting the number of M-partitions is #P-complete. In all other cases, $\#I(G)$ can also be reduced in polynomial time to $\#M(G)$. □

Lemma 12. *If $a = 0$, $b = f = *$, $c = 0$, then counting the number of M-partitions is* #P*-complete.*

In this case the proofs use the universal vertex extension.

References

1. A.A. Bulatov, Tractable conservative constraint satisfaction problems, LICS (2003) 321 – 330.

2. P. Charbit, F. de Montgolfier, M. Raffinot, Linear time split decomposition revisited, SIAM J. Discrete Math., 26 (2012) 499 – 514.
3. V. Chvátal, Star-cutsets and perfect graphs, J. Comb. Th. B 39 (1985) 189 – 199.
4. M. Cygan, M. Pilipczuk, M. Pilipczuk, J.O. Wojtaszczyk, The stubborn problem is stubborn no more, SODA 2011, 1666–1674.
5. N. Linial, Hard enumeration problems in geometry and combinatorics, SIAM Journal on Algebraic and Discrete Methods 7 (1986) 331 – 335.
6. M. Dyer and C. Greenhill, The complexity of counting graph homomorphisms, SODA 1999, pp. 246 – 255.
7. C.M.H. de Figueiredo, The P versus NP-complete dichotomy of some challenging problems in graph theory, Discrete Applied Math., in press.
8. T. Feder, P. Hell, S. Klein, R. Motwani, List partitions, SIAM J. Discrete Math. 16 (2003) 449 – 478.
9. T. Feder, M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction, SIAM J. Comput. 28 (1999) 57 – 104.
10. C.M.H. de Figueiredo, S. Klein, Y. Kohayakawa and B. Reed, Finding skew partitions efficiently, J. Algorithms 37 (2000) 505 – 521.
11. T. Gallai, Transitiv orientierbare Graphen, Acta Mathematica Hungarica 18 (1967) 25 – 66.
12. M. Habib and C. Paul, A survey of the algorithmic aspects of modular decomposition, Computer Science Review 4 (2010) 41 – 59.
13. M.C. Golumbic, **Algorithmic Graph Theory and Perfect Graphs**, Academic Press, New York (1980).
14. P. Hell, Graph partitions with prescribed patterns, to appear.
15. P. Hell, S. Klein, F. Protti, L. Tito, On generalized split graphs, Electronic Notes in Discrete Math. 7 (2001) 98 – 101.
16. P. Hell and J. Nešetřil, On the complexity of H–colouring, J. Combin. Theory B 48 (1990) 92 – 110.
17. P. Hell, J. Nešetřil, **Graphs and Homomorphisms**, Oxford Univ. Press 2004.
18. P. Hell and J. Nešetřil, Counting list homomorphisms and graphs with bounded degrees, in **Graphs, Morphisms and Statistical Physics** (J. Nešetřil and P. Winkler, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science 63 (2004) 105 – 112.
19. P. Hell, J. Nešetřil, Colouring, constraint satisfaction, and complexity, Computer Science Review 2 (2008) 143 –163.
20. P. Jeavons, On the structure of combinatorial problems, Theoretical Comp. Science 200 (1998) 185 – 204.
21. J.S. Provan and M.O. Ball, The complexity of counting cuts and of computing the probability that a graph is connected, SIAM Journal on Computing 12 (1983) 777 – 788.
22. R. E. Tarjan, Decomposition by clique separators, Discrete Math. 55 (1985) 221 – 232.