RAIRO-Inf. Theor. Appl. **39** (2005) 133-144
DOI: 10.1051/ita:2005008

FINDING H-PARTITIONS EFFICIENTLY*

Simone Dantas[1], Celina M.H. de Figueiredo[2], Sylvain Gravier[3] and Sulamita Klein[2]

Abstract. We study the concept of an H-partition of the vertex set of a graph G, which includes all vertex partitioning problems into four parts which we require to be nonempty with only external constraints according to the structure of a model graph H, with the exception of two cases, one that has already been classified as polynomial, and the other one remains unclassified. In the context of more general vertex-partition problems, the problems addressed in this paper have these properties: non-list, 4-part, external constraints only (no internal constraints), each part non-empty. We describe tools that yield for each problem considered in this paper a simple and low complexity polynomial-time algorithm.

Mathematics Subject Classification. 05C85, 68R10.

Introduction

Consider an undirected, finite, simple graph $G = (V(G), E(G))$ and the problem of finding a partition of $V(G)$ into subsets satisfying certain constraints *internal* or *external*. An internal constraint refers to constraints within the parts as to be a clique, an independent set, sparse, dense, etc. An external constraint refers to

Keywords and phrases. Structural graph theory, computational difficulty of problems, analysis of algorithms and problem complexity, perfect graphs, skew partition.

* *This research was partially supported by CNPq, FAPERJ, CAPES (Brazil)/COFECUB (France), project 359/01/03.*

[1] Instituto de Computação, Universidade Estadual de Campinas, Caixa Postal 6176, CEP 13084-971, Campinas, SP, Brasil; sdantas@ic.unicamp.br

[2] Instituto de Matemática and COPPE, Universidade Federal do Rio de Janeiro, Caixa Postal 68530, CEP 21945-970, Rio de Janeiro, RJ, Brasil; celina@cos.ufrj.br & sula@cos.ufrj.br

[3] CNRS, GeoD research group, "Maths à modeler" project, Laboratoire Leibniz, France; sylvain.gravier@imag.fr

© EDP Sciences 2005

constraints between different parts, for example, some parts must be completely adjacent or nonadjacent to other parts.

The skew partition problem was defined by Chvátal [3] as finding a partition of the vertex set of a given graph into four nonempty parts A, B, C, D such that there are all possible edges between A and B, and no edges between C and D. The skew partition problem was defined in the context of perfect graphs and it has a key role in the recent celebrated proof of the strong perfect graph conjecture by Chudnovsky *et al.* [2]. De Figueiredo, Klein, Kohayakawa and Reed [4] presented a polynomial-time algorithm for solving the skew partition problem. Our goal is to contribute to a better understanding of this high complexity polynomial-time algorithm. Note the skew partition problem has only external constraints.

An *H-partition* is a partition of the vertex set $V(G)$ of a graph G into four nonempty parts A, B, C, D such that the adjacencies between vertices placed in distinct parts satisfy constraints given by the edges of a *model* graph $H = (V(H), E(H))$. We call a *model* graph $H = (V(H), E(H))$, a complete graph with 4 vertices $V(H) = \{a, b, c, d\}$ and with 6 edges $E(H) = \{ab, ac, ad, bc, bd, cd\}$, where each vertex $v \in V(H)$ represents one part of the H-partition, and each edge $e \in E(H)$ represents an external adjacency constraint between the two distinct parts of the H-partition corresponding to the endpoints of e. In addition, the edges of a *model* graph H are classified into three types: full edge, dotted edge or non-constraint edge. A *full* edge $ab \in E(H)$ represents the requirement that every vertex of part A is adjacent to every vertex of part B. A *dotted* edge $ab \in E(H)$ represents the requirement that every vertex of part A is nonadjacent to every vertex of part B. A *non-constraint edge* $ab \in E(H)$ represents that there are no adjacency constraints between the vertices of parts A and B. Using this notation, the skew partition is the particular H-partition corresponding to the model graph H depicted on the left of Figure 1, where a full edge is represented by a continuous line, a dotted edge is represented by a dotted line, and a non-constraint edge is omitted.

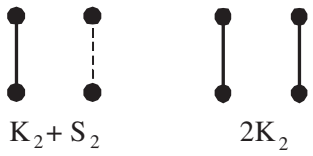

FIGURE 1. Examples of model graphs.

The *complement* $\overline{H}$ of a model graph H is the graph obtained from H by replacing each full edge by a dotted edge and each dotted edge by a full edge (non-constraint edges remain unchanged). Note that a graph G admits an H-partition if and only if its complement $\overline{G}$ admits an $\overline{H}$-partition.

The H-partition problem asks, given a graph G, whether G admits an H-partition, and is a particular case of the *M-partition problem* introduced by Feder, Hell, Klein and Motwani [6, 7]. An M-partition is a partition of a graph

into at most k parts $A_1, A_2, ..., A_k$ where the requirements are encoded by a symmetric k-by-k matrix M in which the diagonal entry $M_{i,i}$ is 0 if A_i is required to be independent, 1 if A_i is required to be a clique and $*$ otherwise (*i.e.*, in case we have no constraints). Similarly, the off-diagonal entry $M_{i,j}$ is 0, $*$ or 1, if A_i and A_j are required to be completely nonadjacent, have arbitrary connections, or are required to be completely adjacent, respectively. In this way, the H-partition is the particular case when the matrix M is a 4-by-4 matrix with only $*$'s in its main diagonal, *i.e.*, it does not impose internal constraints, and with the additional constraint that the parts of the partition are required to be nonempty.

In [6], the authors classified all generalized list H-partition problems which include 4-part problems with arbitrary input lists, internal and/or external constraints, and possibly empty-parts in the solution as quasipolynomial time solvable or NP-complete. Cameron *et al.* [1] further studied and classified all generalized list H-partition problems but one and its complement as polynomial solvable or NP-complete.

Feder and Hell [5] showed that the list homomorphism problems that correspond to list H-partition problems with model graphs 1 through 9 in Figure 2 (that have only full and unconstrained edges, or only dotted and unconstrained edges) can be solved by a polynomial time reduction to 2-SAT. To obtain the list homomorphism to the reflexive graph H'-problem that corresponds to the complement of the list H-partition problem with model graph of Figure 2, take H' to be the complement of the model graph (*i.e.*, include exactly the unconstrained edges, including self-loops). The relevant theorem of reference [5] is Theorem 2.5: if H' is an interval graph then list homomorphism to H' is polynomial time solvable.

In the sequel, we develop simple tools which allow us to analyze most of partitioning problems of the vertex set of a graph G into four nonempty parts with only external constraints. These tools yield low complexity algorithms for all H-partition problems with the exception of two: the skew partition problem, whose model graph H we refer to as $H = K_2 + S_2$, where K_2 is the usual notation for the complete graph with two vertices and S_2 denotes the graph with two vertices and no edges, and the H-partition problem where the model graph H contains a full edge between vertices a and b, a full edge between vertices c and d, and the other 4 edges of H are non-edges, which we refer to as $H = 2K_2$. The first one of these two problems, not studied in this paper, has already been classified. In [4], a polynomial-time algorithm of high complexity was described for the skew partition problem. Considering the second problem, Feder and Hell [5] have shown that its List H-partition version, where $H = 2K_2$, is NP-complete. Unfortunately, this doesn't give us any answer. Our study of H-partition problems leaves as open problems the classification of H-partition problem where $H = 2K_2$, and the task of finding a lower complexity algorithm for the skew partition problem.

Theorem 0.1. *All H-partition problems with the exception of the case $H = 2K_2$ can be solved in polynomial time. Moreover, if $H \neq 2K_2, K_2 + S_2$, then the algorithms are of low complexity.*

1. List H-partition problems

We consider undirected, finite, simple graphs. The H-partition problem is defined as follows:

H-partition problem
Input: a graph $G = (V(G), E(G))$.
Question: is there an H-partition A, B, C, D of $V(G)$?

A convenient way to express the constraints determined by H and the constraint that all parts must be nonempty is to specify for each vertex of G the set of parts of the H-partition in which it is allowed to be. Given a graph G and for each vertex $v \in V(G)$ a list $L(v) \subseteq \{A, B, C, D\}$, a *list H-partition* of G with respect to the lists $\{L(v) : v \in V(G)\}$ is an H-partition A, B, C, D of G in which each $v \in V(G)$ belongs to a part $P \in L(v)$. In other words, the List H-partition problem asks for an H-partition of the input graph G in which each vertex is placed in a part which is in its list. In order to ensure the constraint that A, B, C and D must be nonempty, given the model graph H, we consider for each set of four vertices x_A, x_B, x_C, x_D of $V(G)$ for which the bijection $x_A \mapsto a$, $x_B \mapsto b$, $x_C \mapsto c$, $x_D \mapsto d$ satisfies: each full edge of H corresponds to an edge of G and each dotted edge of H corresponds to a non-edge of G, the following decision problem:

Nonempty part list H-partition problem
Input: a graph $G = (V(G), E(G))$, four vertices x_A, x_B, x_C, x_D of $V(G)$, and for each $v \in V(G)$ a subset $L(v) \subseteq \{A, B, C, D\}$ as follows: $L(x_A) = \{A\}$, $L(x_B) = \{B\}$, $L(x_C) = \{C\}$, $L(x_D) = \{D\}$, and $L(x) = \{A, B, C, D\}$, for all remaining $x \in V(G) \setminus \{x_A, x_B, x_C, x_D\}$.
Question: Is there an H-partition A, B, C, D of $V(G)$ such that each v is contained in some part of $L(v)$?

Let L be a subset of $\{A, B, C, D\}$. We shall drop the brackets and refer to a subset $L = \{A, C, D\}$ simply as $L = ACD$. We shall also denote by L the following subset of $V(G)$: $L = \{v \in V(G) : L(v) = L\}$. We call all lists of size one *trivial lists*. A vertex v such that $L(v) = A$ is said to be *placed* in part A or $v \in A$. Whereas for lists of larger size, for instance a vertex v such that $L(v) = AB$, vertex v is said to be *positioned in* $A \cup B$ or $v \in AB$.

Note that the input of a nonempty part list H-partition problem partitions $V(G)$ into 5 sets, namely $A = \{x_A\}$, $B = \{x_B\}$, $C = \{x_C\}$, $D = \{x_D\}$, $ABCD = V(G) \setminus \{x_A, x_B, x_C, x_D\}$. Vertices x_A, x_B, x_C and x_D of G are placed and the remaining vertices of $V(G)$ will have their lists reduced during the algorithm so that a solution corresponds to a successful reduction of each list to a unitary list so that $V(G)$ is partitioned into four nonempty sets of vertices corresponding to the possible four unitary lists. Note that if a vertex $v \in AB$, then v cannot be placed in part C nor in part D.

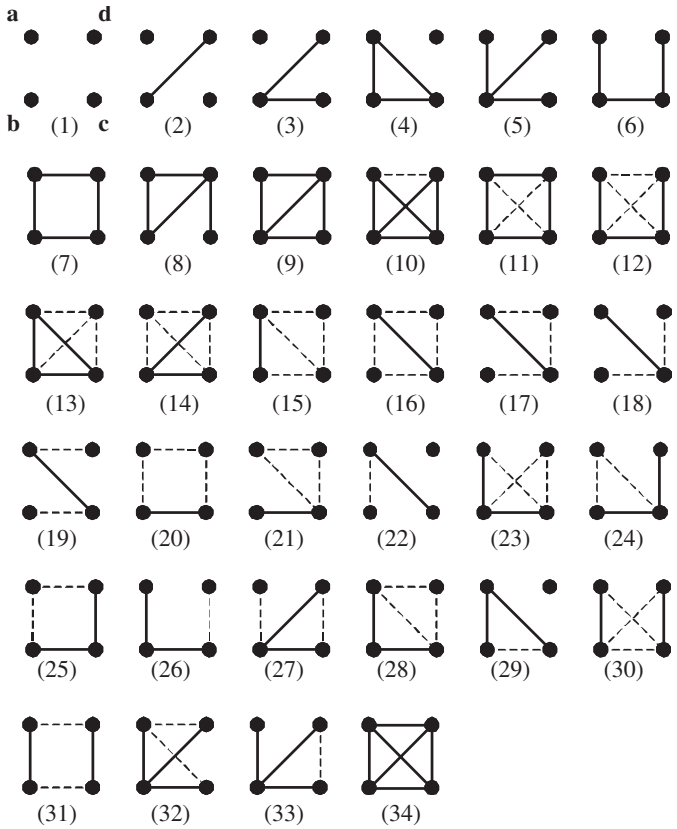

FIGURE 2. List of model graphs.

2. Structure of model graph H

We have a corresponding algorithm for each nonempty part list-H-partition problem, determined by the structure of the model graph H. In this section we describe operations which will identify which type of algorithm we may use in Section 3.

Recall that a *model* graph $H = (V(H), E(H))$ is a complete graph with 4 vertices $V(H) = \{a, b, c, d\}$ and with 6 edges $E(H) = \{ab, ac, ad, bc, bd, cd\}$. The edges of a *model* graph H are classified into three types: full edge, dotted edge or non-constraint edge. Define, for $r \in V(H)$ its *full neighbourhood* $N_F(r) = \{h \in H : hr \text{ is a } full \text{ edge}\}$; and its *dotted neighbourhood* $N_D(r) = \{h \in H : hr \text{ is a dotted edge}\}$. Let L be a subset of $\{A, B, C, D\}$. Extend the concept of

full and dotted neighbourhood accordingly:

$$N_F(L) = \{h \in H : \textit{ there exists } l \in L, \textit{ such that } hl \textit{ is a full edge}\};$$
$$N_D(L) = \{h \in H : \textit{ there exists } l \in L, \textit{ such that } hl \textit{ is a dotted edge}\}.$$

We shall refer to the possible nonempty part list-H-partition problems according to the following numbering:

Lemma 2.1. *All possible model graphs H for the nonempty part list H-partition problem, up to isomorphisms, are presented in Figures 1 and 2.*

2.1. Isolated vertex operation

An *isolated* vertex in a model graph H, is a vertex that is not the end vertex of a full edge nor a dotted edge of H. In case H has an isolated vertex p, the solution is obtained by placing all $x \in V(G) \setminus \{x_A, x_B, x_C, x_D\}$ in the corresponding part P.

2.2. Remaining operations

When we are not able to apply the Isolated vertex operation, we try to use a reduction operation described next, which will reduce H to a smaller model graph H'.

Next, for all cases where the Isolated vertex operation is not used, and considering its reduced model graph, we will generate for a given graph H, its set of refined lists, by applying the operations impossible lists and conflicting lists defined in the sequel.

Reduction operation

Another operation determined by the structure of the model graph H is the *reduction operation* which reduces a given List H-partition problem to a nonempty part list H'-partition problem such that H' has fewer vertices than H.

Two vertices r and s of H are *twins* if rs is a non-constraint edge of H, and such that $N_F(r) = N_F(s)$ and $N_D(r) = N_D(s)$ in H. The reduction operation defines a smaller model graph H' from H as follows: given a pair of twins r and s in H, the model graph H' is a complete graph on vertex set $V(H') = V(H) \setminus \{r, s\} \cup \{s'\}$. The classification of the edges in $E(H')$ into full, dotted and non-constraint is given by the classification of the edges in $E(H)$ as follows: if $e' \in E(H')$ is not incident to s', then e' was present in $E(H)$, and e' has the same classification as in $E(H)$; whereas if e' is incident to s', then $e' = s'x$, corresponds to $e = sx \in E(H)$ and e' has the same classification as e.

For example, when H is the model graph of Figure 3, we can group the vertices c and d into a vertex $c' = c \cup d$ and reduce the problem to a model graph H' with just three vertices a', b', c'.

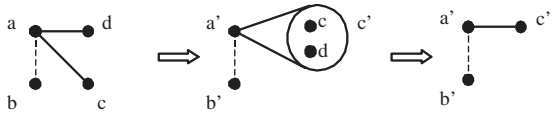

FIGURE 3. Reduction operation.

We present in Table 1 the model graphs after the reduction operation.

TABLE 1. New model graphs after reduction operation.

Subcases	Model graph H'
(5)	$a' = b$, $b' = a \cup c \cup d$
(7)	$a' = a \cup c$, $b' = b \cup d$
(9)	$a' = a \cup c$, $b' = b$, $c' = d$
(16)	$a' = a$, $b' = b \cup d$, $c' = c$
(18)	$a' = a$, $b' = b \cup d$, $c' = c$
(25)	$a' = a$, $b' = b \cup d$, $c' = c$
(32)	$a' = a$, $b' = b$, $c' = c \cup d$

Impossible lists

The structure of a model graph H can determine a set of lists that are *impossible* for this model graph H.

Given a model graph H, the operation impossible lists computes its corresponding set of possible lists by considering for all non-trivial lists L, the neighbourhoods $N_F(L)$ and $N_D(L)$.

Let L_i be a non-trivial list. If $L_i \subset L_j$ and $L_i \neq L_j$, for $i \neq j$, and $N_F(L_i) = N_F(L_j)$ and $N_D(L_i) = N_D(L_j)$, then L_i is an *impossible list*.

For example, consider the model graph H subcase (34) of Figure 2. Clearly, $N_F(ABCD) = V(H)$, and $N_D(ABCD) = \emptyset$. Given any non-trivial list L properly contained in $ABCD$, we have $N_F(L) = V(H)$, and $N_D(L) = \emptyset$. So the only non-trivial possible list for the subcase (34) is $ABCD$.

Conflicting lists

On the other hand, the set of possible lists may be further reduced by verifying which lists are conflicting. This tool was used extensively in [1, 7].

A *conflicting* list, with respect to a model graph H, is a list that imposes two conflicting constraints on a vertex, *i.e.*, by having this list a vertex of G should be simultaneously adjacent and nonadjacent to all vertices already placed in the same part. We call *non-conflicting* lists the lists that are not conflicting.

Given a model graph H, after the operation impossible lists is performed, we apply the operation conflicting lists which computes the corresponding set of non-conflicting lists of the model graph H by considering for each $p \in V(H)$ whether

there exist in H a full edge rp and a dotted edge sp incident to p. In such a case, every list L such that $R, S \in L$ is a conflicting list.

For example, consider the model graph H subcase (11) of Figure 2. Vertex a has incident edges ab which is full and ac which is dotted. So, by definition, BC is a conflicting list and so are lists $ABC, BCD, ABCD$, as they contain list BC. Indeed, assume that during the algorithm a vertex v is in BC. So, since $B \in L(v)$, v should be adjacent to all vertices placed in A and since $C \in L(v)$, v should be nonadjacent to all vertices placed in A. In this case, v should be adjacent and nonadjacent to each vertex already placed in A, a contradiction. Similarly, for subcase (11), lists AB, AD, BC, CD, ABD, ACD can be shown conflicting.

In order to obtain the set of refined lists, we performed first the operation impossible lists, and then we performed the operation conflicting lists. The following lemma shows that this order is not relevant.

Lemma 2.2. *Let R be the set of lists obtained by performing first the operation impossible lists followed by performing the operation conflicting lists. Let R' be the set of lists obtained by performing first the operation conflicting lists followed by performing the operation impossible lists. Then $R = R'$.*

Proof. Clearly, $R \subseteq R'$. Suppose $R \subset R'$. Then there exist lists $L' \in R' \setminus R$, $L \notin R'$ such that: $|L'| = 2$, $|L| = 3$, L' is not a conflicting list and L is conflicting list, $L' \subset L$, $N_F(L') = N_F(L)$ and $N_D(L') = N_D(L)$.

Without loss of generality, let $L' = AC$, $L = ABC$. Since ABC is a conflicting list, we may assume with no loss of generality that AB is a conflicting list.

We distinguish now two cases, and both lead to a contradiction.

First assume ad is a dotted edge, and bd is a full edge. Now $d \in N_F(ABC) = N_F(AC)$ says cd is a full edge, which in turn implies AC is a conflicting list, a contradiction.

Second assume ac is a dotted edge, and bc is a full edge. Now $\{b, c\} \subseteq N_F(ABC)$ and $c \notin N_F(AC)$ is a contradiction. □

3. Efficient nonempty part list H-partition algorithm

We have as input of the nonempty part list H-partition problem a graph $G = (V(G), E(G))$, with $|V(G)| = n$, $|E(G)| = m$, and four initial vertices x_A, x_B, x_C and x_D of $V(G)$ which have as lists A, B, C and D, respectively, and all other vertices $x \in V(G)$ with list $ABCD$.

If the model graph H has an isolated vertex as in one of the subcases (1), (2), (3), (4), (22), (29) presented in Figure 2 then the algorithm solves the problem by placing the vertices in a part corresponding to an isolated vertex of H, as described in Subsection 2.1.

In all remaining subcases listed and numbered in Figure 2, the algorithm proceeds by trying to position the vertices $v \in V(G) \setminus \{x_A, x_B, x_C, x_D\}$ into one of the refined lists of the model graph H as follows.

We initialize the position of all vertices of $V(G)$ as follows: $A = x_A$, $B = x_B$, $C = x_C$, $D = x_D$, $ABCD = V(G) \setminus \{x_A, x_B, x_C, x_D\}$ and all other refined lists

determined by the structure of H are empty. For each vertex v of $V(G)$, v can be placed or v can be positioned in one of the non-trivial refined lists. This is made by means of keeping as invariant the property below:

Property 3.1. *If ab is a full edge of H and $A \in L(v)$, then v sees every vertex with list B. If $B \in L(v)$, then v sees every vertex with list A.*

In the same way, if ab is a dotted edge of H and $A \in L(v)$ then v is nonadjacent to every vertex with list B. If $B \in L(v)$ then v is nonadjacent to every vertex with list A.

Once a vertex v is placed, it is necessary to update as follows all non-trivial refined lists in order to keep the Property 3.1 as invariant. Let ab be a full edge and $v \in L$ with $A \in L$. If v does not see a vertex in B, v cannot be in a set L containing A. Then, we move v to $L \setminus A$. Similarly, if ab is a dotted edge and $v \in L$ with $A \in L$. If v is adjacent to a vertex in B, we move v to $L \setminus A$.

If during the process of updating $L(v)$, the list $L(v)$ is reduced to the empty set, then the algorithm stops with the answer NO, and the instance in question (this graph G with those four particular vertices placed as indicated by their four trivial initial lists) does not have a corresponding nonempty part list H-partition.

Else, we succeed in updating all $L(v)$, positioning all vertices of G in one of the refined lists according to Property 3.1. If all $L(v)$ are reduced to a trivial list, the algorithm stops with the answer YES and the instance in question (this graph G with those four particular vertices placed as indicated by their four trivial initial lists) does have a corresponding nonempty part list H-partition.

Note that for the subcase (12) in Figure 2, the set of refined lists contains only lists A, B, C and D. This means that we always have a decision after running the positioning procedure.

Else and finally, we succeed in updating all $L(v)$, positioning all vertices of G in one of the refined lists according to Property 3.1, but not all $L(v)$ are reduced to a trivial list. We show next that these nonempty part list H-partition problems always have a simple solution by applying the tool List transversal, as described below.

List transversal

A *list transversal* is a list L_T that intersects all lists of size at least 2, and such that L_T is trivial, or nontrivial having no constraints between the parts contained in L_T.

The existence of a list transversal L_T leads to a solution because each vertex v can be placed in a part $P \in L(v) \cap L_T$, for all $|L(v)| \geq 2$, and $P \in \{A, B, C, D\}$.

For example, let H be the model graph subcase (6) of Figure 2. The refined lists for this subcase are: A, B, C, D, AC, AD, BD, ABD, ACD, $ABCD$. List AD satisfies the definition of a list transversal. Hence, a solution for this nonempty part list H-partition problem is given by: $A = \{v \in V(G) | A \in L(v)\}$, $B = \{x_B\}$, $C = \{x_C\}$, and $D = \{v \in V(G) | D \in L(v) \text{ and } A \notin L(v)\}$.

We present in Table 2 the refined lists and the list transversal for the final subcases.

TABLE 2. Refined lists and list transversals.

Subcases	Refined lists	Transversal
(5)	A', B', A'B'	A' or B'
(6)	A, B, C, D, AC, AD, BD, ABD, ACD, ABCD	AD
(7)	A', B', A'B'	A' or B'
(8)	A, B, C, D, AC, BC, ABC, ABCD	C
(9)	A', B', C', A'B'C'	A' or B' or C'
(13)	A, B, C, D, ABC	A or B or C
(14)	A, B, C, D, CD	C or D
(15)	A, B, C, D, AB, AD, CD	BD
(16)	A', B', C', A'C'	A' or C'
(17)	A, B, C, D, AC, BC, BD	AB
(18)	A', B', C', A'B', B'C'	B'
(19)	A, B, C, D, AC, AD, BC, BD	AB or CD
(20)	A, B, C, D, AB, AD, BC, CD	AC or BD
(21)	A, B, C, D, AD, BC, ACD	AB or BD
(23)	A, B, C, D, AC	A or C
(24)	A, B, C, D, BD, BCD	B or D or BD
(25)	A', B', C', A'B', B'C'	B'
(26)	A, B, C, D, AB, AC, AD, ABC, ACD	A or AC
(27)	A, B, C, D, AB, CD	AC or AD
(28)	A, B, C, D, ACD	A or C or D
(30)	A, B, C, D, AB, CD	AD
(31)	A, B, C, D, AB, AD, BC, CD	AC or BD
(32)	A', B', C', A'C'	A' or C'
(33)	A, B, C, D, AB, AC, AD, ACD	AC or AD
(34)	A, B, C, D, ABCD	A or B or C or D

One can extend the list transversal tool in order to solve the remaining cases (10) and (11). In fact, if there exists a solution to nonempty part list H-partition problems (10) and (11) then there exists one for which all vertices having A in its list are placed in part A. Therefore, one can consider the following *transversals*:

TABLE 3. Refined lists and List transversals.

Subcases	Refined lists	Transversal
(10)	A, B, C, D, AD, BC	AB or AC or DB or DC
(11)	A, B, C, D, AC, BD	AB or AD or CB or CD

Now, it is sufficient to check if these transversals define a solution (*i.e.* if there is no violated constraint).

Complexity analysis

For each $O(n^4)$ feasible assignment of one vertex to each part, we update in $O(n^2)$ time the set of non-trivial refined lists in order to keep the Property 3.1 as invariant. Finally, we place in $O(n)$ time the vertices according to corresponding list transversal listed in Tables 2 and 3. For the problems listed in Table 3, we need to check whether there is no violated constraint in additional $O(n^2)$ time.

Therefore, the simple proposed solution for each nonempty part list H-partition problems of Figure 2 decides in time $O(n^2)$ whether a given feasible assignment of one vertex to each part yields a solution.

4. CONCLUSION

We have presented polynomial-time algorithms for all H-partition problems, with the exception of the cases $H = 2K_2$, $H = K_2+S_2$. The described polynomial-time algorithms are simple and of low complexity.

Recently, Cameron *et al.* [1] studied List H-partition problems that allow the partitions to have at most 4 parts. In addition, they considered adding inner constraints to the parts as to be a clique or an independent set. We refer to this problem as a generalized list H-partition problem.

Clearly, a polynomial-time algorithm for such a generalized list H-partition problem also solves the corresponding non list H-partition problem (where the partition is required to have 4 nonempty parts). Nevertheless, the converse is not true. For instance, [1] classified as an NP-complete problem any generalized List H-partition with parts A, B and C required to be independent sets, part D with no inner constraints, and no external constraints between the parts. The corresponding non list partition problem with 4 nonempty parts is trivially a polynomial problem as any graph with at least 4 vertices admits such a partition.

It is clear that similar simple tools as the ones developed in the present paper can be applied even if we add internal constraints to the parts of the desired H-partition. For instance, Cameron *et al.* [1] call *stubborn* problem the partition of a graph G into at most four parts A, B, C, D, with the following constraints: the only external constraint is that every vertex of part A is nonadjacent to every vertex of part C; the internal constraints are part A and part B are required to be independent sets, and part D is required to be a clique. The stubborn problem (and its complement) is the only problem of the class studied

in [1] that was not classified as NP-complete or polynomial time solvable. The stubborn problem such that all but four special vertices x_A, x_B, x_C, x_D have lists $ABCD$ can be easily solved by noticing that its refined set of nontrivial lists is $\{ABCD, ABC, ACD, BCD, AC, BC, BD, CD\}$. Indeed, a vertex containing A in its list must contain also C. Therefore, we may place all vertices containing C in their nontrivial lists into C, and verify if the remaining graph (subgraph induced by the vertices with list BD) is a split graph which can be done by applying 2-SAT.

Acknowledgements. We are very grateful to Pavol Hell and to the referee who helped us to integrate our results in the subject of list partition problems and its existing bibliography.

References

[1] K. Cameron, E.M. Eschen, C.T. Hoàng and R. Sritharan, The list partition problem for graphs, in *Proc. of the ACM-SIAM Symposium on Discrete Algorithms – SODA 2004*. ACM, New York and SIAM, Philadelphia (2004) 384–392.

[2] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas, Strong Perfect Graph Theorem, in *Perfect Graph Conjecture workshop*. American Institute of Mathematics (2002).

[3] V. Chvátal, Star-Cutsets and Perfect Graphs. *J. Combin. Theory Ser. B* **39** (1985) 189–199.

[4] C.M.H. de Figueiredo, S. Klein, Y. Kohayakawa and B. Reed, Finding Skew Partitions Efficiently. *J. Algorithms* **37** (2000) 505–521.

[5] T. Feder and P. Hell, List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B* **72** (1998) 236–250.

[6] T. Feder, P. Hell, S. Klein and R. Motwani, Complexity of graph partition problems, in *Proc. of the 31st Annual ACM Symposium on Theory of Computing - STOC'99*. Plenum Press, New York (1999) 464–472.

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

To access this journal online:
www.edpsciences.org