On H-partition problems

Abstract

We study the concept of H-partition the vertex set of a graph G which includes all vertex partitioning problems into four nonempty parts with only external restrictions according to the structure of a model graph H.

author: Simone Dantas **(Student)**
e-mail: sdantas@cos.ufrj.br
Affiliation: Programa de Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro.
Adress: Caixa Postal 68511 - CEP: 21945-970 - Rio de Janeiro - RJ.

author: Celina M. H. de Figueiredo
e-mail: celina@cos.ufrj.br
Affiliation: Instituto de Matemática and COPPE, Universidade Federal do Rio de Janeiro.

author: Sylvain Gravier
e-mail: Sylvain.Gravier@imag.fr
Affiliation: CNRS, Laboratoire Leibniz.

author: Sulamita Klein
e-mail: sula@cos.ufrj.br
Affiliation: Instituto de Matemática and COPPE, Universidade Federal do Rio de Janeiro.

On H-partition problems[*]

Simone Dantas[†] Celina M. H. de Figueiredo[‡] Sylvain Gravier[§]
Sulamita Klein[‡]

Abstract

We study the concept of H-partition the vertex set of a graph G which includes all vertex partitioning problems into four nonempty parts with only external restrictions according to the structure of a model graph H.

1 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 restrictions *internal* or *external*. An internal restriction refers to restrictions within the parts as to be a clique, an independent set, sparse, dense, etc. An external restriction refers to restrictions between different parts, for example, some parts must be completely adjacent or nonadjacent to other parts.

The Skew Partition Problem was defined by Chvátal [2] 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. So the Skew Partition Problem has only external restrictions. De Figueiredo, Klein, Kohayakawa and Reed [3] presented a polynomial-time algorithm for solving the Skew Partition Problem.

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 restrictions given by the edges of a *model* graph $H = (V(H), E(H))$ such that $V(H) = \{a, b, c, d\}$ and $E(H) = \{ab, ac, ad, bc, bd, cd\}$. Each vertex of H represents one part of the H-partition, and each edge of H represents an external adjacency restriction.

[*] This research was partially supported by CNPq, MCT/FINEP PRONEX Project 107/97, CAPES (Brazil)/COFECUB (France), project number 213/97, FAPERJ.

[†] COPPE, Universidade Federal do Rio de Janeiro, Brazil.

[‡] Instituto de Matemática and COPPE, Universidade Federal do Rio de Janeiro, Brazil.

[§] CNRS, Laboratoire Leibniz, France.

The edges of $E(H)$ can be of three types: full edge, dotted edge or non-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-edge* $ab \notin E(H)$ represents that there are no adjacency restrictions 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 of Figure 1.

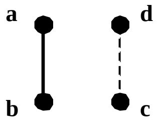

Figure 1: Skew partition - Model Graph H.

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-edges remain unchanged). Note that 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 [5]. An M-Partition is a partition of a graph into 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, 2 if A_i is required to be a clique and 1 otherwise (i.e., in case we have no restrictions). Similarly, the off-diagonal entry $M_{i,j}$ is 0, 1 or 2, if A_i and A_j are required to be completely nonadjacent, have arbitrary connections, or are required to be completely adjacent, respectively. In this way, the H-partition is the particular case when the matrix M is a 4-by-4 matrix with only 1's in its main diagonal, i.e., it does not impose internal restrictions, and with the additional hypothesis that the parts of the partition are required to be nonempty.

In the sequel, we develop tools which allow us to analyse all partitioning problems of the vertex set of a graph G into four nonempty parts with only external restrictions. These techniques are related to those applied by De Figueiredo, Klein, Kohayakawa and Reed for solving the Skew Partition Problem [3].

2 Refining an H-Partition Problem

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 *General 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 General 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.

General List H-Partition Problem
Input: a graph $G = (V(G), E(G))$ and, for each vertex $v \in V$, a subset $L(v)$ of $\{A, B, C, D\}$.
Question: Is there an H-partition A, B, C, D of $V(G)$ such that each v is contained in some part of the corresponding $L(v)$?

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:

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 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 by *trivial lists* and we denote $\{A\}$ by A. A vertex v such that $L(v) = A$ is said to be *placed* in a set A or $v \in A$. We use an analogous notation for lists of larger size, for instance a vertex v such that $L(v) = AB$ is said to be in $A \cup B$ or $v \in AB$.

Note that the input of a 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 four sets of vertices having unitary lists. Note that if during the algorithm $v \in AB$, then v will not be placed in C or D. We observe next some properties of each problem determined by the structure of its model graph H.

Conflicting lists

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

For instance, consider the model graph H depicted in Figure 2. For this H, we have that *ab* is a full edge and that *bc* is a dotted edge. We will show that AC is a conflicting list for this model H. Assume that during the algorithm a vertex v is in AC.

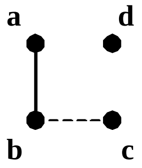

Figure 2: AC: Conflicting List for List H-Partition Problem.

So, since $A \in L(v)$, v should be adjacent to all vertices placed in B, and since $C \in L(v)$, v should be nonadjacent to all vertices placed in B. In this case, v should be adjacent and nonadjacent to each vertex already placed in B, a contradiction. Thus, we conclude that AC is a conflicting list.

Given a List H-Partition Problem, as a first step, our algorithm computes its corresponding set of non-conflicting lists by considering for each $p \in V(H)$ whether there exist in H a full edge and a dotted edge incident to p. The existence of a full edge rp and of a dotted edge sp both incident to p leads to the conflicting lists L such that $R, S \in L$.

Impossible lists

On the other hand, the set of non-conflicting lists can be further reduced by verifying which lists are *impossible* to occur during the algorithm.

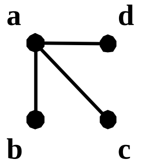

Figure 3: AP: Impossible Lists for List H-Partition Problem.

Let $N_F(L)$ be the set of neighbors in H by full edges of the vertices of H corresponding to the list L. We define $N_D(L)$ in an analogous way by considering its set of neighbors in H by dotted edges. Given a List H-Partition Problem, as a second step, our algorithm computes its corresponding set of possible lists by considering for all non-conflicting and non-trivial lists L, $N_F(L)$ and $N_D(L)$. 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.

Let H be the model graph with full edges ab, ac, ad and non-edges bc, bd and cd (see Figure 3). Suppose that $L(v)$ is a non-trivial list such that there exist $P, Q \in \{B, C, D\}$ such that $P \in L(v)$ and $Q \notin L(v)$. This implies that v is not adjacent to w placed in A and this is turn contradicts $P \in L(v)$, so the only non-trivial possible lists are $ABCD$ and BCD.

Thus, before starting to solve the problem we eliminate all conflicting and impossible lists. We call this procedure *Refining Operation* and we call *Refined Lists* the set of remaining lists after the refining operation.

3 Positioning Vertices

We have in the input of the List H-Partition Problem 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 $ABCD$. 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.

So, 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 (in one of the trivial lists) 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 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 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 a vertex of G can be placed in no refined lists then the algorithm stops with the answer NO, and the instance in question does not have a corresponding list H-partition.

4 Solution Tools

Let H be one of the model graphs depicted in Figure 8 and let all vertices of G after the positioning according to Property 1 be in one of the refined lists. We show next that these List H-Partition Problems have always a solution by applying one of the following solution tools:

Isolated Vertex Operation

An *isolated* vertex in a model graph H, is a vertex that is not end vertex of a full edge or of a dotted edge of H. In case H has an isolated vertex p we place all $x \in V(G) \setminus \{x_A, x_B, x_C, x_D\}$ in the corresponding part P.

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 restrictions 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 $|L(v)| \geq 2$, $P \in \{A, B, C, D\}$.

For example, let H be the model graph depicted in Figure 4. We observe that after refining the set of lists according to the model graph H, the refined lists are A, B, C,

D, AB, AD, BC, CD. List AC satisfies the definition of a list transversal. Hence, a solution for this List H-Partition Problem is presented in Figure 4: $A = \cup_{A \in L} L$, $B = \{x_B\}$, $C = \cup_{C \in L} L$ and $D = \{x_D\}$.

Using the same argument, we have that BD is another list transversal for this model graph H.

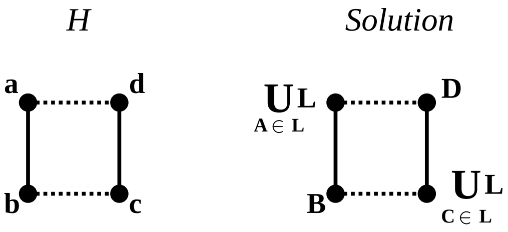

Figure 4: AC is a List Transversal for List H-Partition Problem.

Reduction Operation

Another tool is the *Reduction Operation* which reduces a given List H-Partition Problem to an H'-Partition Problem such that H' has fewer vertices than H.

Two vertices r and s of H are *twins* if there is no full edge rs and no dotted edge rs in 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 vertex set $V(H') = V(H) \setminus \{r, s\} \cup \{s'\}$; the edge set $E(H') = E(H) \setminus \{\mathit{all\ edges\ incident\ to\ r\ or\ s}\} \cup \{s't : t\ \mathit{is\ adjacent\ to}\ s\}$;

For example, when H is the model graph of Figure 5, 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'.

The solution now is obtained by the following Lemma:

Lemma 2 *Let H be a model graph. The problem of determining whether a graph G has an H-partition where $|V(H)| < 4$ can be solved in polynomial-time.*

Proof. If $|V(H)| = 1$, there is nothing to do. When $|V(H)| = 2$ the problem is solved by the 2-SAT algorithm of Aspvall, Plass and Tarjan [1] because all lists have size at most 2. For details of this solution we refer to [4].

If $|V(H)| = 3$, then we have the cases (and the complementary cases) presented in Figure 6. In the model graphs (2) and (3), we apply the isolated vertex operation. In the model graph (1), we apply the reduction operation contracting the pair of twins a and c.

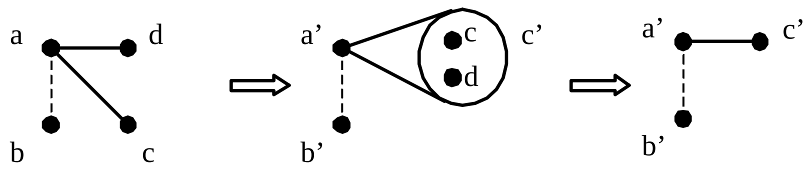

Figure 5: Reduction Operation.

The remaining cases have a solution because, after applying the refining operation, we can always find a list transversal. The refined lists of these three graphs are respectively $\{A, B, C, ABC\}$ for graph (4), $\{A, B, C, AB, BC, ABC\}$ for graph (5) and $\{A, B, C, AC\}$ for graph (6). The corresponding list transversals are A or B or C; B; A or C, respectively. ■

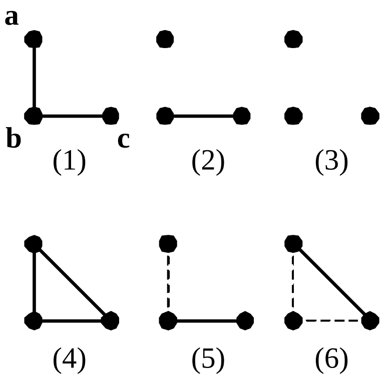

Figure 6: The model graphs with three vertices.

5 Main Theorem

Lemma 3 *All possible model graphs H for the List H-Partition Problem, up to isomorphisms, are presented in Figures 7 and 8.* ■

Theorem 4 *The List H-Partition Problem is polynomial-time solvable when H is one of the graphs of Figure 8.*

Proof. The subcases (2), (3), (4), (22), (29), (34) are solved by the isolated vertex operation. Table 1 presents the subcases for which the solution is obtained by the reduction operation. For the subcase (12), the set of refined lists contains only lists A, B, C and D. This means that we have a decision after running the positioning procedure.

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

Table 1: New model graph after reduction operation.

In the remaining cases, we apply the refining operation. In case we do not find in the refined lists a list transversal, we observe that all refined lists have size at most 2 and the problem is solved by the 2-SAT algorithm of Aspvall, Plass and Tarjan [1]. These cases are shown by Table 2.

Subcases	Refined lists
(10)	A, B, C, D, AD, BC
(11)	A, B, C, D, AC, BD
(16)	A, B, C, D, AC, BD

Table 2: Subcases solved by 2-SAT algorithm.

Otherwise, we present in Table 3 the refined lists and the list transversal for the final subcases. So, all H-Partition Problems corresponding to the model graphs H depicted in Figure 8 can be solved in polynomial time. ■

By Lemma 3, the remaining cases are expressed by the graphs listed in Figure 7 and their complements. The model graph on the left is the graph H for Skew partition and it was solved in polynomial time by de Figueiredo, Klein, Kohayakawa and Reed [3].

Subcases	Refined lists	Transversal
(1)	A, B, C, D, AC, AD, BD, ABD, ACD, ABCD	AD
(6)	A, B, C, D, ABCD	A or B or C or D
(8)	A, B, C, D, AC, BC, ABC, ABCD	C
(13)	A, B, C, ABC	A or B or C
(14)	A, B, C, D, CD	C or D
(15)	A, B, C, D, AB, BC, CD	AC
(17)	A, B, C, D, AC, BC, BD	AB
(19)	A, B, C, D, AC, AD, BC, CD	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
(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
(33)	A, B, C, D, AB, AC, AD, CD, ACD	AC or AD

Table 3: Refined lists and List transversals.

6 Conclusion

A solution for the List H-Partition Problem, when H is the model graph (1) depicted in Figure 7 was presented in [3]. We have presented a solution for the List H-Partition Problem for all model graphs H depicted in Figure 8. We leave the solution of the only remaining List H-Partition Problem, the model graph (2) depicted in Figure 7, as an open problem.

References

[1] B. Aspvall, F. Plass and R. E. Tarjan. A Linear-time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Information Processing Letters*, 8 (1979) 121–123.

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

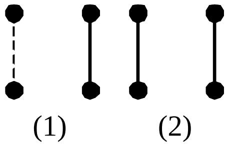

Figure 7: Remaining cases of theorem.

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

[4] H. Everett, S. Klein and B. Reed. An Optimal Algorithm for Finding Clique-Cross Partitions. *Congr. Numer.*, 135 (1998) 171-177.

[5] T. Feder, P. Hell, S. Klein, and R. Motwani. Complexity of graph partition problems. In F. W. Thatcher and R. E. Miller, eds., *Proceedings of the 31st Annual ACM Symposium on Theory of Computing - STOC'99*, pages 464–472. Plenum Press, New York, 1999.

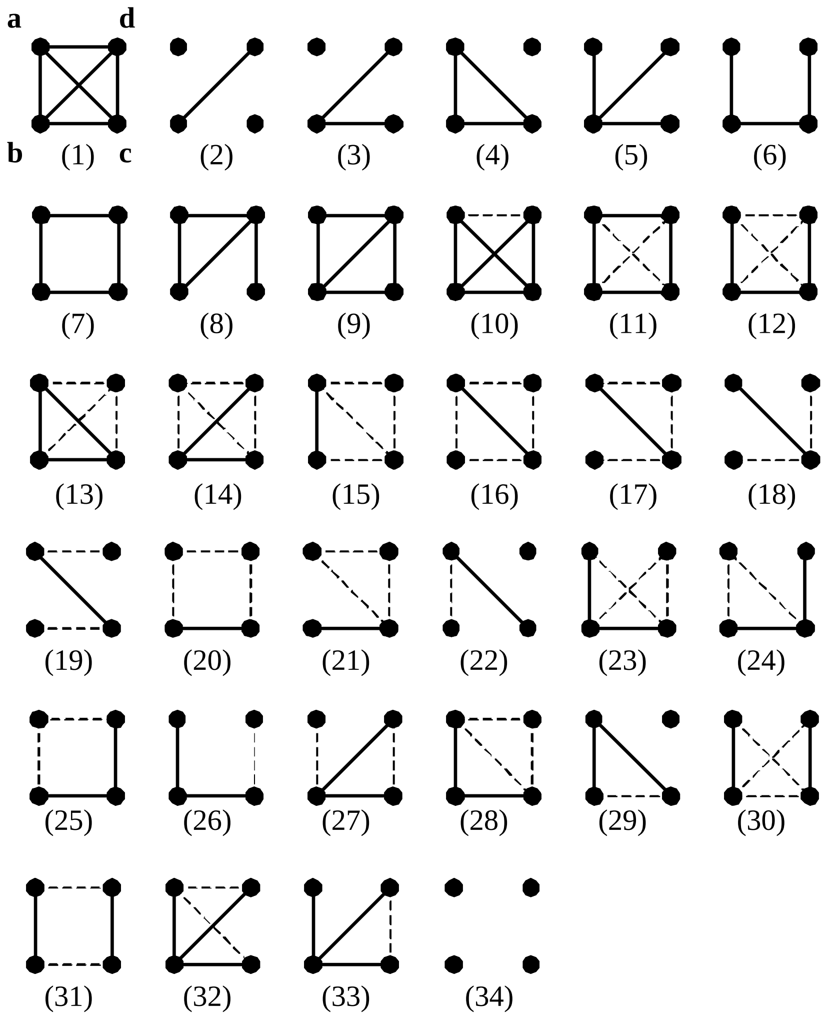

Figure 8: List of model graphs for Theorem 4.