Maximum-Size Subgraphs of P4-Sparse Graphs Admitting a Perfect Matching

Stavros D. Nikolopoulos and Leonidas Palios

Department of Computer Science, University of Ioannina,
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr

Abstract. In this paper, we address the problem of computing a maximum-size subgraph of a P_4-sparse graph which admits a perfect matching; in the case where the graph has a perfect matching, the solution to the problem is the entire graph. We establish a characterization of such subgraphs, and describe an algorithm for the problem which for a P_4-sparse graph on n vertices and m edges, runs in $O(n+m)$ time and space. The above results also hold for the class of complement reducible graphs or cographs, a well-known subclass of P_4-sparse graphs.

Keywords: Perfect graphs, P_4-sparse graphs, cographs, maximum-size subgraphs, maximum matchings, perfect matching.

1 Introduction

The class of P_4-*sparse* graphs was introduced by Hoàng in his doctoral dissertation [11], as the class of graphs for which every set of five vertices induces at most one P_4 (chordless path on four vertices). Hoàng gave a number of characterizations of these graphs and showed that the P_4-sparse graphs are perfect in the sense of Berge (a graph G is perfect if for every induced subgraph H of G, the chromatic number of H equals the clique number of H), and in fact perfectly orderable in the sense of Chvátal [1,9]. The class of P_4-sparse graphs generalizes the well known class of *complement reducible* graphs, also known as *cographs* [14].

The study of P_4-sparse graphs and cographs led naturally to constructive characterizations that implied several linear-time recognition algorithms and also enabled the construction of unique, up to isomorphism, tree representations [2,4,12,13]. In addition, since P_4-sparse graphs and cographs are perfect, many interesting optimization problems in graph theory, which are NP-complete in general graphs, admit polynomial sequential solutions; their tree representations are used by many researchers to develop algorithms for such problems (see [1,9]). In particular, Jamison and Olariu [13] proposed linear-time algorithms for solving five optimization problems on the class of P_4-sparse graphs: maximum-size clique, maximum-size stable set, minimum coloring, minimum covering by cliques, and minimum fill-in. Moreover, in [12] the same authors provided efficient solutions to other classical optimization problems; that is, finding the

P. Bozanis and E.N. Houstis (Eds.): PCI 2005, LNCS 3746, pp. 68–79, 2005.

clique number, the stable number, the chromatic number and the clique cover number of a P_4-sparse graph. Giakoumakis and Vanherpe [8] obtained linear-time algorithms for the maximum weight clique and for the maximum weight stable set problems on P_4-sparse graphs using the modular decomposition tree representation [5,15]. Yang and Yu [18] exhibited a linear time algorithm for the maximum matching problem in cographs, while Fouquet, Parfenoff, and Thuillier [7] extended this algorithm to P_4-tidy graphs, a class containing both P_4-sparse graphs and cographs.

A *matching* M of a graph G is a subset of the edge set $E(G)$ such that no two edges in M share a common endpoint; M is a *maximum matching* if it contains a maximum number of edges; M is a *perfect matching* if every vertex of G is an endpoint of an edge in M. The best known algorithm for solving the maximum matching problem in general graphs is due to Micali and Vazirani [16] and has $O(\sqrt{n}\,m)$ time complexity; recall that in P_4-sparse graphs and cographs, the same problem is solved in linear time due to the algorithm of Fouquet, Parfenoff, and Thuillier [7].

In this paper, we are interested in solving the problem of finding a maximum-size subgraph of a P_4-sparse graph which has a perfect matching. In other words, we want to find and remove the smallest number of edges so that the graph which we obtain if we ignore any isolated vertices has a perfect matching. The problem belongs to the class of problems in which we are asked to remove as few edges or vertices as possible so that the resulting subgraph has some particular properties.

We show that any maximum-size subgraph of a graph G which has a perfect matching is a subgraph induced by the vertices of a maximum matching of G. In this way, we reduce the problem to that of finding a maximum-size subgraph induced by the vertices of a maximum matching of G. Then, we establish a characterization of such subgraphs which by means of the modular decomposition tree representation of the P_4-sparse graphs enables us to obtain a linear-time solution to the problem we consider.

2 Preliminaries

We consider finite undirected graphs with no loops or multiple edges. For a graph G, we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively. Let S be a subset of the vertex set of a graph G. Then, the subgraph of G induced by S is denoted by $G[S]$. Moreover, we denote by $G - S$ the graph $G[V(G) - S]$.

Modular Decomposition

A subset M of vertices of a graph G is said to be a *module* of G, if every vertex outside M is either adjacent to all vertices in M or to none of them. The emptyset, the singletons, and the vertex set $V(G)$ are *trivial* modules and whenever G has only trivial modules it is called a *prime* (or *indecomposable*) *graph*. A module M of G is called a *strong module* if, for any module M' of G, either $M' \cap M = \emptyset$ or one module is included into the other. Furthermore, a module in G is also a module in $\overline{G}$ (i.e., the complement of the graph G).

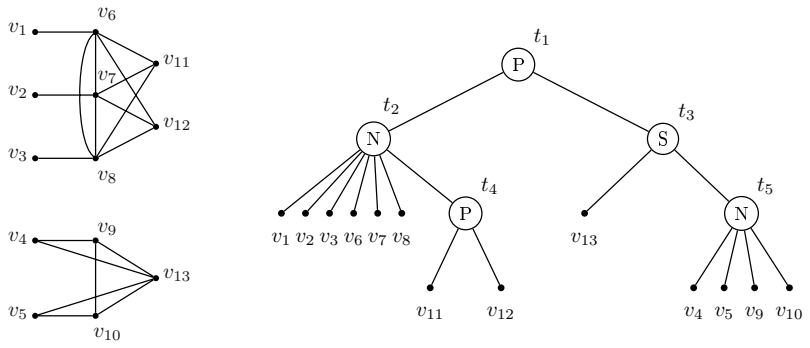

Fig. 1. A disconnected P_4-sparse graph on 13 vertices and its md-tree

The *modular decomposition* of a graph G is a linear-space representation of all the partitions of $V(G)$ where each partition class is a module. The *modular decomposition tree* $T(G)$ of the graph G (or *md-tree* for short) is a unique labeled tree associated with the modular decomposition of G in which the leaves of $T(G)$ are the vertices of G and the set of leaves associated with the subtree rooted at an internal node induces a strong module of G. Thus, the md-tree $T(G)$ represents all the strong modules of G. An internal node is labeled by either P (for *parallel* module), S (for *series* module), or N (for *neighborhood* module). It has been shown that for every graph G the md-tree $T(G)$ is unique up to isomorphism and it can be constructed in linear time; the first linear-time algorithms for the construction of the md-tree are described in [5,15], while more recent and more practical ones can be found in [6,10]. Figure 1 depicts a P_4-sparse graph G on 13 vertices and its md-tree $T(G)$.

Let t be an internal node of the md-tree $T(G)$ of a graph G. We denote by $M(t)$ the module corresponding to t which consists of the set of vertices of G associated with the subtree of $T(G)$ rooted at node t. Let $u_1, u_2, \ldots, u_p$ be the children of the node t of $T(G)$. We denote by $G(t)$ the *representative graph* of the module $M(t)$ defined as follows: $V(G(t)) = \{u_1, u_2, \ldots, u_p\}$ and $u_iu_j \in E(G(t))$ if there exists edge $v_kv_\ell \in E(G)$ such that $v_k \in M(u_i)$ and $v_\ell \in M(u_j)$; by the definition of a module, if a vertex of $M(t_i)$ is adjacent to a vertex of $M(t_j)$ then every vertex of $M(t_i)$ is adjacent to every vertex of $M(t_j)$. Thus, $G(t)$ is isomorphic to the graph induced by a subset of $M(t)$ consisting of a single vertex from each maximal strong submodule of $M(t)$ in the modular decomposition of G. For the P-, S-, and N-nodes, the following lemma holds (see also [8]):

Lemma 2.1. *Let G be a graph, $T(G)$ its modular decomposition tree, and t an internal node of $T(G)$. Then, $G(t)$ is an edgeless graph if t is a P-node, $G(t)$ is a complete graph if t is an S-node, and $G(t)$ is a prime graph if t is an N-node.*

P_4-sparse Graphs

A graph G is called a *spider* if the vertex set $V(G)$ of the graph G admits a partition into sets S, K, and R such that:

(P1) $|S| = |K| \geq 2$, the set S is an independent set, and the set K is a clique;
(P2) all the vertices in R are adjacent to all the vertices in K and to no vertex in S;
(P3) there exists a bijection $f : S \longrightarrow K$ such that exactly one of the following statements holds:
  (i) for each vertex $v \in S$, $N(v) \cap K = \{f(v)\}$;
  (ii) for each vertex $v \in S$, $N(v) \cap K = K - \{f(v)\}$.

The triple (S, K, R) is called the *spider-partition*. A graph G is a *prime spider* if G is a spider with $|R| \leq 1$. If the condition of case P3(i) holds then the spider G is called a *thin spider*, whereas if the condition of case P3(ii) holds then G is a *thick spider*; note that the complement of a thin spider is a thick spider and vice versa.

Observation 2.1 (Observation 2.8 in [12]). *If a graph G is a spider, then exactly one of the following statements holds:*

(i) for every $v \in S$ and $u \in K$, degree$(v) = 1$ and degree$(u) = |V(G)| - |S|$;
(ii) for every $v \in S$ and $u \in K$, degree$(v) = |K| - 1$ and degree$(u) = |V(G)| - 2$.

Observation 2.2 (Observation 2.9 in [12]). *If a graph G is a spider and R is nonempty, then for every choice of v, u, and r in S, K, and R, respectively, degree$(v) <$ degree$(r) <$ degree(u).*

It is not difficult to see that a spider with $|K| = |S| = k$ contains exactly $\frac{k(k-1)}{2} + \ell$ P_4s, where ℓ is the number of P_4s in the subgraph $G[R]$. From the definition of the spider and Observations 2.1 and 2.2, it follows that if G is a spider, then S, K, and R are unique (see [12]). Finally, from the properties of a spider G, and also from the definition of the P_4-sparse graphs, it easily follows that G is P_4-sparse iff the graph $G[R]$ is P_4-sparse.

Let us now return to general P_4-sparse graphs. Then, the following result holds:

Lemma 2.2 (Theorem 1 in [12]). *For a graph G, the following conditions are equivalent:*

(i) G is a P_4-sparse graph;
(ii) for every induced subgraph H of G with at least two vertices, exactly one of the following statements is satisfied: (a) H is disconnected; (b) $\overline{H}$ is disconnected; (c) H is a spider.

Regarding the modular decomposition of P_4-sparse graphs, Giakoumakis and Vanherpe [8] showed the following result (recall that the graph $G(t)$ has vertices the children of the node t in $T(G)$):

Lemma 2.3. *Let G be a graph and let $T(G)$ be its modular decomposition tree. The graph G is P_4-sparse iff for every N-node t of $T(G)$, $G(t)$ is a prime spider with a spider-partition (S, K, R) and no vertex of $S \cup K$ is an internal node in $T(G)$.*

3 Finding a Max-size Subgraph That Has a Perfect Matching

In this section we give an optimal sequential algorithm for the problem of finding a maximum-size subgraph which has a perfect matching. It is important to note that such a subgraph is an induced subgraph; the condition that this subgraph be of maximum size requires that the subgraph be the subgraph induced by its vertex set. Below, we show that any maximum-size subgraph of a graph G, which has a perfect matching, is induced by the matched vertices in a maximum matching of G (Lemma 3.1); then, we prove the property that characterizes a maximum-size subgraph of G, among the subgraphs that are induced by the vertices of maximum matchings of G (Observation 3.1).

Lemma 3.1. *Let G be a graph and let G_1 be a maximum-size subgraph of G which has a perfect matching. Then, the subgraph G_1 is induced by the vertices participating in a maximum matching of G.*

Proof: Let M_{max} be a maximum matching of G, whose vertex set is V_{max}; if $n_{max} = |V_{max}|$ then any matching involving n_{max} matched vertices is a maximum matching of G. If $V_1 \subseteq V(G)$ is the vertex set of the subgraph G_1, then, because G_1 has a perfect matching, $|V_1| \leq n_{max}$. We will show that $|V_1| = n_{max}$, because then any perfect matching of G_1 is a maximum matching of G. It suffices to show that $|V_1| \geq n_{max}$. Suppose, for contradiction, that $|V_1| < n_{max}$. Then, we have that $V_1 \not\subseteq V_{max}$, since otherwise the vertices in V_{max} would induce a subgraph that has a perfect matching and is of size larger than that of G_1, in contradiction to the maximality of G_1. Yet, we will show that we can construct a matching of G whose vertex set has cardinality n_{max} and is a proper superset of V_1; this will yield a contradiction to the maximality of G_1.

Let M_1 be a perfect matching of G_1 (the vertices in V_1 are all matched in the matching M_1) and let us consider the graph H spanned by the non-common edges of the matchings M_1 and M_{max}. Then, the vertices in H have degree at most 2; in particular, the vertices in $V_1 - V_{max}$ have degree exactly 1 and are incident on an edge that participates in M_1. The fact that the degrees of the vertices of H do not exceed 2 implies that each connected component of H is either a path or a cycle. Additionally, edges of M_{max} and M_1 alternate on these paths and cycles.

Let us consider a vertex $x \in V_1 - V_{max}$ and let $\rho = v_0 v_1 \cdots v_t$, $t \geq 1$, be the path (connected component) of H to which x belongs; since x is an endpoint of ρ, we assume without loss of generality that $x = v_0$. Then, the length of ρ cannot be 1; otherwise, $v_1 \in V_1 - V_{max}$, which implies that the edges in M_{max} and the edge xv_1 define a matching of G larger than M_{max}, in contradiction to the optimality of M_{max}. In fact, the length of ρ cannot be odd: if $t = 2q + 1$, where $q \geq 1$, then the edges $v_{2i-1}v_{2i}$, $1 \leq i \leq q$, belong to M_{max} and the edges $v_{2i}v_{2i+1}$, $0 \leq i \leq q$, belong to M_1; thus, the vertices $v_1, v_2, \ldots, v_{t-1}$ belong to $V_1 \cap V_{max}$ and the vertices v_0, v_t belong to $V_1 - V_{max}$, which implies that we can replace all the edges $v_{2i-1}v_{2i}$, $1 \leq i \leq q$, by the remaining edges and obtain

a matching of G with vertex set $V_{max} \cup \{x, v_t\}$, again in contradiction to the maximality of M_{max}. Therefore, the path ρ has even length, say, $t = 2q$ where $q \geq 1$. Then, as above, the edges $v_{2i-1}v_{2i}$, $1 \leq i \leq q$, belong to M_{max}, the edges $v_{2i}v_{2i+1}$, $0 \leq i \leq q-1$, belong to M_1, the vertices $v_1, v_2, \ldots, v_{t-1}$ belong to $V_1 \cap V_{max}$ and $v_0 \in V_1 - V_{max}$ and $v_t \in V_{max} - V_1$. Thus, if we replace all the edges $v_{2i-1}v_{2i}$, $1 \leq i \leq q$, by the remaining edges, we obtain a matching of G with vertex set $\big(V_{max} \cup \{v_0\}\big) - \{v_t\}$; this matching has n_{max} matched vertices among which we find vertex x whereas the vertex $v_t \in V_{max} - V_1$ is left unmatched.

By doing the above process for each vertex in $V_1 - V_{max}$ and by taking into account that the paths in H to which the vertices in $V_1 - V_{max}$ belong are disjoint, we obtain a matching M of G with vertex set V_M such that $|V_M| = n_{max}$ and $V_1 \subset V_M$. But then, the subgraph $G[V_M]$ has a perfect matching and is of size larger than the size of $G_1 = G[V_1]$, a contradiction to the maximality of G_1. Therefore, the number of vertices of a maximum-size subgraph of G which has a perfect matching is n_{max} and thus is induced by the vertices of a maximum matching of G. ∎

Lemma 3.1 implies that each maximum-size subgraph of a graph G which has a perfect matching is a maximum-size subgraph of G induced by the vertices of a maximum matching of G. Therefore, in the following, we will be referring to the problem of finding a maximum-size subgraph that has a perfect matching while concentrating on maximum-size subgraphs induced by maximum matchings. Then, the following observation allows us to characterize the maximum matching that will yield a maximum-size induced subgraph of G which we seek.

Observation 3.1. *Among all maximum matchings of a graph G, any one whose vertices induce a maximum-size subgraph of G exhibits the minimum sum of degrees of unmatched vertices.*

Proof: Observe that any two vertices left unmatched during the computation of a maximum matching are not adjacent; otherwise, the edge connecting them would produce a larger matching. Hence, the number of edges of the subgraph of G induced by the vertices participating in a maximum matching is equal to the total number of edges of G minus the sum of the degrees of the unmatched vertices. Thus, in order to obtain a maximum-size subgraph of G induced by the vertices of a maximum matching of G, we need to find a maximum matching of G such that the unmatched vertices have the smallest sum of degrees in G. ∎

Based on Observation 3.1, one might think that, if we know the number k of vertices left unmatched in a maximum matching of G, the maximum-size subgraph of G that we seek can be obtained by removing the k vertices of G of smallest degrees. This is not however the case: consider for example the graph H on 14 vertices $\{v, x_1, x_2, x_3, y_1, \ldots, y_5, z_1, \ldots, z_5\}$ where v is adjacent to all the remaining vertices, $H[\{x_1, x_2, x_3\}]$ is a complete graph on 3 vertices, and $H[\{y_1, \ldots, y_5\}]$ and $H[\{z_1, \ldots, z_5\}]$ are complete graphs on 5 vertices each; H has a maximum matching involving 12 vertices, yet, the subgraph induced by

removing any two of the vertices $\{x_1, x_2, x_3\}$ (which exhibit the smallest degrees in H) does not have a perfect matching.

The following lemma establishes another useful property pertaining to a maximum matching.

Lemma 3.2. *Let G be a graph, M_{opt} be a maximum matching of G inducing a subgraph of G whose size is maximum (among all subgraphs induced by the vertices of a maximum matching), M be any other maximum matching of G, and U_{opt} (resp. U) be the set of vertices of G left unmatched by M_{opt} (resp. M). Then there is a bijection $f : U \to U_{opt}$ such that for each vertex $x \in U$, $degree(f(x)) \leq degree(x)$ in G, where by $degree(v)$ we denote the degree of vertex v in G.*

Our algorithm relies on the following lemma as well.

Lemma 3.3. *Let G be a spider and let (S, K, R) be its spider-partition. If a maximum-size subgraph of $G[R]$ induced by a maximum matching results after removal of the vertices in $U \subseteq R$, then*

(i) if $|U| \leq |S|$, a maximum-size subgraph of G induced by a maximum matching is $G - X$ where X is an arbitrary subset of S of cardinality $|U|$;
(ii) if $|U| > |S|$, a maximum-size subgraph of G induced by a maximum matching is $G - (S \cup Y)$ where Y is the set of the $|U| - |S|$ vertices of U of smallest degrees in G.

Our algorithm takes advantage of the modular decomposition tree $T(G)$ of the input graph G. To simplify the computations, we "binarize" the S-nodes and P-nodes of the tree $T(G)$. The algorithm processes the resulting tree $T'(G)$ as follows: in each node t of the tree, it computes a maximum matching for the subgraph $G[M(t)]$ of G corresponding to the subtree of $T'(G)$ rooted at t, while at the same time minimizing the sum of the degrees of the vertices of $G[M(t)]$ left unmatched.

Algorithm MaxSubgraph
Input: a P_4-sparse graph G.
Output: a maximum-size subgraph of G which has a perfect matching.

1. Compute the degrees of all the vertices in the graph G and store them in an array;
2. Construct the md-tree $T(G)$ of G;
   Make each S-node or P-node of $T(G)$ binary, obtaining the modified modular decomposition tree $T'(G)$;
3. Execute the subroutine *process*(*root*), where *root* is the root node of the modified md-tree $T'(G)$; the sought subgraph is the subgraph $G - U$, where U is the set of vertices returned by the subroutine.

where the description of the subroutine *process*( ) is as follows:

process(node t)

Input: node t of the modified md-tree $T'(G)$ of the input graph G.
Output: the set U_t of vertices whose removal leaves a maximum-size subgraph of $G[M(t)]$ which has a perfect matching.

1. `if` t is a leaf
   `then return`$(\{v\})$, where v is the vertex associated with the leaf t;
2. `if` t is an N-node
   `then` compute the spider-partition (S, K, R) of $G(t)$; $\{$*note:* $|R| \leq 1\}$
   `if` $R = \{t_R\}$
   `then` $U_R \leftarrow$ *process*(t_R);
   `else` $U_R \leftarrow \emptyset$; $\{$*note:* $R = \emptyset\}$
   `if` $|U_R| \leq |S|$
   `then` $X \leftarrow$ arbitrary subset of S of cardinality $|U_R|$;
   `else` $Y \leftarrow$ set of $|U_R| - |S|$ vertices of U_R of smallest degrees in G;
   $X \leftarrow S \cup Y$;
   `return`(X);
3. $\{$*node t is an S-node or a P-node that has a left and a right child with associated vertex subsets V_ℓ and V_r respectively*$\}$
   $U_\ell \leftarrow$ *process*(left child of t);
   $U_r \leftarrow$ *process*(right child of t);
   suppose without loss of generality that $|U_\ell| \geq |U_r|$, otherwise swap the two children of t and the corresponding sets;
4. `if` t is a P-node
   `then return`$(U_\ell \cup U_r)$;
5. `if` t is an S-node
   `then if` $|U_\ell| = |V_r|$
   `then return`$(\emptyset)$;
   `else if` $|U_\ell| < |V_r|$
   `then if` $|U_\ell| - |U_r|$ is even
   `then return`$(\emptyset)$;
   `else` $v \leftarrow$ vertex in $V_\ell \cup V_r$ of smallest degree in G;
   `return`$(\{v\})$;
   `else` $\{|U_\ell| > |V_r|\}$
   $X \leftarrow$ set of $|U_\ell| - |V_r|$ vertices of U_ℓ of smallest deg. in G;
   `return`(X);

For each node t of the tree $T'(G)$, subroutine process( ) computes a maximum matching for the subgraph $G[M(t)]$ implicitly; it can be easily modified to store and print such a matching. The correctness of Algorithm MaxSubgraph follows from Lemma 3.4.

Lemma 3.4. *When applied on a P_4-sparse graph G, Algorithm MaxSubgraph correctly computes a maximum-size subgraph of G which has a perfect matching.*

Proof: We need only prove the correctness of subroutine process( ). The proof proceeds inductively on the height of the (sub)tree of the modified md-tree $T'(G)$

of the input graph G rooted at the node t that is currently processed by the subroutine. If the (sub)tree is of height 0, then t is a leaf. Hence, it corresponds to a subgraph on 1 vertex; such a graph has no matching and its single vertex is left unmatched, which is what the subroutine returns (Step 1).

For the inductive hypothesis, we assume that the subroutine process( ) correctly handles any (sub)tree of height at most $h \geq 0$; we show that it correctly handles any (sub)tree of height $h+1$. Let t be the root of such a (sub)tree; then, t is an N-node, a binarized S-node, or a binarized P-node. If t is an N-node, then the graph $G(t)$ is a prime spider (Lemma 2.3); let (S, K, R) be its spider partition (recall that $R = \emptyset$ or $R = \{t_R\}$). Lemma 3.3 certifies the correctness of the computation in Step 2 taking into account that the ordering of the vertices of $G(t)$ by their degree in $G(t)$ is identical to the ordering based on their degree in G (note that $V(G(t))$ is a module and thus each vertex in $V(G(t))$ is adjacent to the same vertices in $V(G) - V(G(t))$), and that if $R = \{t_R\}$, by the inductive hypothesis, subroutine process( ) has correctly computed a maximum-size subgraph of $G[M(t_R)]$ which has a perfect matching. Suppose now that t is a P-node or an S-node; in either case, t has two children. If V_ℓ and V_r are the vertex subsets corresponding to the leaves of the left and right child of t respectively, and U_ℓ and U_r are the sets of unmatched vertices returned by subroutine process( ) respectively, then by the inductive hypothesis, the subgraphs $G[V_\ell - U_\ell]$ and $G[V_r - U_r]$ are maximum-size subgraphs of $G[V_\ell]$ and of $G[V_r]$ which have a perfect matching.

If t is a P-node, then there are no edges of G connecting a vertex in V_ℓ to a vertex in V_r; hence, the optimal solution for the subgraph of G corresponding to t, i.e., induced by $V_\ell \cup V_r$, is the union of the optimal solutions for the subgraphs $G[V_\ell]$ and $G[V_r]$, just as the algorithm does (Step 4).

Finally consider that t is an S-node, and assume without loss of generality (as the algorithm assumes) that $|U_\ell| \geq |U_r|$. Then, if $|U_\ell| = |V_r|$, the subgraph $G[V_\ell \cup V_r]$ has a perfect matching: extend the perfect matching of $G[V_\ell - U_\ell]$ with a matching resulting from an arbitrary bijection from U_ℓ to V_r; subroutine process( ) correctly reports that no vertex remains unmatched in this case. An optimal solution is also produced if $|U_\ell| < |V_r|$: a perfect matching is obtained if the total number of vertices is even, otherwise exactly the vertex of U_r with the smallest degree in G (which has the smallest degree in $G[M(t)]$ as well because $V_\ell \cup V_r$ is a module) is left unmatched (the optimality of this solution follows from Observation 3.1), and this can be easily shown to be feasible since $|U_r| \leq |U_\ell| < |V_r|$.

Consider now the case in which $|U_\ell| > |V_r|$. In this case, the algorithm constructs the set X containing the $|U_\ell| - |V_r|$ vertices in U_ℓ of minimum degrees in G, which it returns as the set of vertices left unmatched; this is indeed a feasible solution because a perfect matching in $G[(V_\ell - X) \cup V_r]$ can be constructed from a matching of $G[V_\ell - U_\ell]$ and a matching resulting from an arbitrary bijection from $U_\ell - X$ to V_r. Suppose for contradiction that this is not an optimal solution for the subgraph $G[M(t)]$ of G corresponding to the subtree of $T'(G)$ rooted at the S-node t; let U be the set of unmatched vertices in an optimal solution for

$G[M(t)]$. Then, $|U| = |X|$: the optimality of $G[M(t)] - U$ implies that $|U| \leq |X|$, while if $|U| < |X|$ the fact that the subgraph $G[M(t)] - U$ has a perfect matching would imply that a maximum-size subgraph of $G[V_\ell]$ which has a perfect matching would leave a number of unmatched vertices at most $|U|+|V_r| < |U_\ell|$, in contradiction to the optimality of the solution for $G[V_\ell]$ which leaves unmatched the vertices in U_ℓ. The optimality of $G[M(t)] - U$ and the choice of X also imply that

$$\sum_{v \in U} \text{degree}(v) \;<\; \sum_{v \in X} \text{degree}(v) \;\leq\; \sum_{v \in A} \text{degree}(v) \tag{1}$$

for any subset A of U_ℓ of cardinality $|U_\ell| - |V_r|$, where $degree(v)$ denotes the degree of vertex v in G.

Additionally, $U \subseteq V_\ell$; if not, then the fact that U cannot contain both a vertex in V_ℓ and V_r (for otherwise two such vertices would be matched resulting in a larger matching) would imply that $U \subseteq V_r$, which would in turn imply that $G[V_\ell]$ has a matching leaving unmatched at most $|V_r| - |U| \leq |V_r| < |U_\ell|$ vertices, a contradiction. We note however that it does not have to be the case that $U \subseteq U_\ell$; nevertheless, we will show that there exists a bijection $g : U \to W$, where $W \subseteq U_\ell$, such that for each vertex $x \in U$, $degree(g(x)) \leq degree(x)$.

Let us consider the restriction of the matching which yields the optimal solution for the graph $G[M(t)]$ (and leaves unmatched the vertices in U) to the vertex set V_ℓ; if Z is the set of vertices left unmatched by the resulting matching M', then clearly $U \subseteq Z$ and $|Z| \leq |U| + |V_r| = |U_\ell|$. Thus, since the optimal matching for this subgraph (by the inductive hypothesis) leaves unmatched the vertices in U_ℓ, $|Z| = |U_\ell|$ and the matching M' is maximum for the subgraph $G[V_\ell]$. Then, Lemma 3.2 applied on the two matchings for the graph $G[V_\ell]$, which leave unmatched the vertices in Z and U_ℓ respectively, yields that there exists a bijection $f : Z \to U_\ell$ such that for each $x \in Z$, $degree(f(x)) \leq degree(x)$. Then, the desired bijection g is the restriction of the bijection f to the domain U, and the set W is the set of images $f(x)$ of the vertices x in U. The properties of the bijection g imply that

$$\sum_{v \in W} \text{degree}(v) \;\leq\; \sum_{v \in U} \text{degree}(v),$$

which comes into contradiction with Inequality (1), since W is a subset of U_ℓ of cardinality $|U| = |U_\ell| - |V_r|$. Therefore, the solution produced by the algorithm for the case when $|U_\ell| > |V_r|$ is also optimal. ∎

Time and Space Complexity of Algorithm MaxSubgraph: Let n and m be the number of vertices and edges of the input P_4-sparse graph G. Then, the degrees of the vertices of G can be computed in $O(n+m)$ time and can be stored in $O(n)$ space, so that Step 1 of the algorithm takes $O(n+m)$ time and $O(n)$ space. The md-tree $T(G)$ can be constructed in $O(n + m)$ time and space [5,15,6,10] and has $O(n)$ size. Binarizing the S- and P-nodes takes $O(n)$ time and the resulting tree $T'(G)$ has $O(n)$ size. Thus, Step 2 takes $O(n + m)$ time and space as well.

Let us now bound the time and space needed by all the executions of subroutine process( ). The set of unmatched vertices returned by a call to process( ) is stored in an unordered linked list so that linking two such lists takes constant time. Observe that each node t of the tree $T'(G)$ is processed exactly once. If t is a leaf, then the processing of t takes constant time (Step 1). If t is an N-node, then t's processing takes $O(|S| + |U_R|)$ time (see Step 2): the set S is readily available from the md-tree; forming the set Y requires copying the vertices in U_R along with their degrees in an array, applying the linear-time selection algorithm [3] on this array to locate the value of the $(|U_R| - |S|)$-th smallest degree, and using this value to partition the vertices in U_R based on their degrees (special attention is needed for the vertices with degree equal to the partitioning value). In turn, the processing of a P-node takes constant time (thanks to the linked list representation of the set of unmatched vertices), while the processing of an S-node takes $O(|V_\ell| + |V_r|)$ time (again, forming the set X requires the application of the linear-time selection algorithm). We get a bound on the time taken for the processing of the entire tree $T'(G)$ by using the following crediting scheme: we credit each leaf and each (binary) P-node with 1 credit; we credit each N-node (corresponding to a spider with partition sets S, K, and R) with the number of edges of G connecting vertices in S and K, and additionally if $R = \{t_R\}$ with the number of edges connecting vertices in K and $M(t_R)$, which is at least equal to $|S|$ or $|S| + |K| \cdot |M(t_R)|$ respectively; we credit each (binary) S-node with 1 plus the number of edges of G connecting a vertex associated with a leaf in the left subtree of the S-node to a vertex associated with a leaf in the right subtree, that is, with $1 + |V_\ell| \cdot |V_r|$ credits. Since $|K| \geq 2$, $|V_\ell| + |V_r| \leq 1 + |V_\ell| \cdot |V_r|$, and $|U_R| \leq |M(t_R)|$ if $R = \{t_R\}$, the time taken is bounded by a constant multiple of the number of credits. Then, because each edge of G contributes at most one credit in our crediting scheme and because the size of $T'(G)$ is $O(n)$, the time required for the completion of Step 3 of Algorithm MaxSubgraph is $O(n + m)$. Clearly, the space needed by subroutine process( ) is $O(n)$.

The results of this section can be summarized in the following theorem.

Theorem 3.1. *Let G be a P_4-sparse graph on n vertices and m edges. The problem of finding a maximum-size subgraph of G admitting a perfect matching is solved in $O(n + m)$ time and space.*

4 Concluding Remarks

Motivated by this work, it would be interesting to consider the related problem where edges are added so that the resulting graph has a perfect matching while remaining in the same class of graphs, that is,

- Given a P_4-sparse graph (or cograph) G, find the minimum number of edges which need to be added to the edge set of G such that the resulting graph is a P_4-sparse graph (or cograph) and admits a perfect matching.

We expect that the structural results and the algorithmic approach used in this paper can help develop efficient algorithms for solving this problem as well.

References

1. A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.
2. A. Bretscher, D. Corneil, M. Habib, and C. Paul, A simple linear time LexBFS cograph recognition algorithm, *Proc. 29th Int'l Workshop on Graph Theoretic Concepts in Comput. Sci. (WG'03)*, LNCS 2880 (2003) 119–130.
3. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.
4. D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14** (1985) 926–934.
5. A. Cournier and M. Habib, A new linear algorithm for modular decomposition, *Proc. 19th Int'l Colloquium on Trees in Algebra and Programming (CAAP'94)*, LNCS **787** (1994) 68–84.
6. E. Dahlhaus, J. Gustedt, and R.M. McConnell, Efficient and practical algorithms for sequential modular decomposition, *J. Algorithms* **41** (2001) 360–387.
7. J.L. Fouquet, I. Parfenoff, and H. Thuillier, An O(n) time algorithm for maximum matchings in P_4-tidy graphs, *Inform. Process. Lett.* **62** (1999) 281–287.
8. V. Giakoumakis and J. Vanherpe, On extended P_4-reducible and extended P_4-sparse graphs, *Theoret. Comput. Sci.* **180** (1997) 269–286.
9. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc., 1980.
10. M. Habib, F. de Montgolfier, and C. Paul, A simple linear-time modular decomposition algorithm for graphs, using order extension, *Proc. 9th Scandinavian Workshop on Algorithm Theory (SWAT'04)*, LNCS 3111 (2004) 187–198.
11. C. Hoàng, Perfect graphs, *Ph.D. thesis*, McGill University, Montreal, Canada, 1985.
12. B. Jamison and S. Olariu, A tree representation for P_4-sparse graphs, *Discrete Appl. Math.* **35** (1992) 115–129.
13. B. Jamison and S. Olariu, Linear-time optimization algorithms for P_4-sparse graphs, *Discrete Appl. Math.* **61** (1995) 155–175.
14. H. Lerchs, On cliques and kernels, *Technical Report*, Department of Computer Science, University of Toronto, March 1971.
15. R.M. McConnell and J. Spinrad, Linear time modular decomposition and efficient transitive orientation of comparability graphs, *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms (SODA'94)* (1994), 536–545.
16. S. Micali and V.V. Vazirani, An $O(\sqrt{n}\,m)$ algorithm for finding maximum matching in general graphs, *Proc. 21st Annual IEEE Symposium on Foundation of Computer Science (FOCS'80)*, (1980) 17–27.
17. K. Nakano, S. Olariu, and A.Y. Zomaya, A time-optimal solution for the path cover problem on cographs, *Theoret. Comput. Sci.* **290** (2003) 1541–1556.
18. C-H. Yang and M-S. Yu, An O(n) time algorithm for maximum matchings in cographs, *Inform. Process. Lett.* **47** (1993) 89–93.