On the recognition of probe graphs of some self–complementary classes of perfect graph

Maw-Shang Chang[1], Ton Kloks*, Dieter Kratsch[2], Jiping Liu**[3], and Sheng-Lung Peng[4]

[1] Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan 621, R.O.C.
mschang@cs.ccu.edu.tw
[2] Université de Metz
LITA, 57045 Metz Cedex 01
France
kratsch@sciences.univ-metz.fr
[3] Department of Mathematics and Computer Science
The university of Lethbridge
Alberta, T1K 3M4, Canada
liu@cs.uleth.ca
[4] Department of Computer Science and Information Engineering
National Dong Hwa University
Hualien 974, Taiwan, R.O.C.
lung@csie.ndhu.edu.tw

Abstract. In this paper we consider the recognition of some probe graph classes. Given a class of graphs $\mathcal{G}$, a graph G is a probe graph of $\mathcal{G}$ if its vertices can be partitioned into a set $\mathbb{P}$ of *probes* and an independent set $\mathbb{N}$ of *non–probes*, such that G can be extended to a graph of $\mathcal{G}$ by adding edges between certain non-probes. We consider the recognition problem of probe cographs, probe P_4–reducible graphs, probe P_4–sparse graphs, and probe split graphs.

Introduction

Probe interval graphs were introduced in [22, 23] to model certain problems in physical mapping of DNA when only partial data is available on the overlap of clones. Chapter 4 of [11] is dedicated to this class of graphs. In the biological application the partition of the vertex set into two sets, probes and non-probes,

* Supported by the National Science Council under grant NSC 93–2811–M–002–004. I thank the following Departments and Universities for their hospitality and support where part of this research took place. These are the LITA department of the Université de Metz, the Department of Mathematics and Computer Science of the University of Lethbridge, and the Department of Mathematics of the National Taiwan University.

** Corresponding author.

is part of the input and the problem is to add edges between some non-probes to complete the graph into an interval graph. This problem was solved successfully in [19] in $O(n^2)$ time. An alternative $O(n + m \log n)$ time algorithm appeared in [21]. The problem of recognizing probe interval graphs in polynomial time when the partition of the vertex set is *not* part of the input was recently solved in [4]. Earlier attempts were made to gain a better understanding of the structure of these graphs by analyzing probe *chordal* graphs [10]. In this paper polynomial time recognition algorithms were given for the family of probe chordal graphs, both in the case when a partition is a part of the input and for the situation where no partition is given.

In this paper we proliferate the research into probe graph classes by investigating probe cographs, probe P_4–reducible graphs, probe P_4–sparse graphs, and probe split graphs. The general pursuit is to investigate the partitioned case first. Ensuing, the unpartitioned case is dealt with by exhibiting a polynomial number of feasible partitions.

1 Preliminaries

A graph G is a pair $G = (V, E)$, where V is a finite set the elements of which are called the *vertices* of G and where E is a family of two-element subsets of V, called the edges. We write $n = |V|$ for the number of vertices and $m = |E|$ for the number of edges.
For a graph $G = (V, E)$ and a subset $S \subseteq V$ of vertices, we write $G[S]$ for the subgraph of G *induced* by S. For a subset $W \subseteq V$ of vertices of a graph $G = (V, E)$ we write $G - W$ for the graph $G[V - W]$, i.e., the subgraph induced by $V - W$. For a vertex x we write $G - x$ rather than $G - \{x\}$. For other conventions on graph–related notations we refer to any standard textbook. For graph classes not defined here we refer to [9, 15].

Definition 1. *Let $\mathcal{G}$ be a class of graphs. A graph G is a* probe graph of $\mathcal{G}$ *if its vertices can be partitioned into a set $\mathbb{P}$ and an independent set $\mathbb{N}$ such that G can be embedded into a graph $G' \in \mathcal{G}$ by adding edges between certain vertices of $\mathbb{N}$. The vertices of $\mathbb{P}$ are called* probes *and those of $\mathbb{N}$ are called* non–probes.

If the partition of the vertices into probes and non–probes is part of the input, then we call the graph a *partitioned probe graph*. We call a graph $G' \in \mathcal{G}$ obtained from G by adding some edges between vertices of $\mathbb{N}$ an *embedding* of G.

Theorem 1 (Probe Sandwich Theorem). *Let $\mathcal{G}$ be a* self–complementary *class of graphs, i.e., $H \in \mathcal{G} \Leftrightarrow \overline{H} \in \mathcal{G}$. Let $G = (\mathbb{P} + \mathbb{N}, E)$ be a graph with a partition of its vertices into $\mathbb{P}$ and an independent set $\mathbb{N}$. Let G^* be obtained from $\overline{G}$ by removing all edges between vertices of $\mathbb{N}$. Then G is a partitioned probe graph of $\mathcal{G}$ if and only if G^* is in the same category.*

To clarify our motivation for investigating these problems, we digress shortly from the actual topic of this paper in the remaining part of this section.

Let $\mathcal{G}$ be the class of *perfect graphs*.[1] Lovász proved in 1972 that the class of perfect graphs is self–complementary [20]. If $G = (\mathbb{P} + \mathbb{N}, E)$ is a probe perfect graph, then the complement of every odd cycle of length at least 5 contains exactly two non–probes, and these are connected by an edge in any embedding of G. This observation, and the Probe Sandwich Theorem 1 mentioned above, led us to conjecture the following:

Conjecture 1 (Probe Perfect Graph Conjecture). There exists a polynomial time algorithm to test if a partitioned graph $G = (\mathbb{P} + \mathbb{N}, E)$ is probe perfect.

Conjecture 2 (Strong Probe Perfect Graph Conjecture). There exists a polynomial time algorithm to test if a graph is probe perfect.

Remark 1 (Motivation). Our reason for investigating these conjectures and probe graphs of special classes of perfect graphs are partly based upon the fact that there exists a nice decomposition tree for a perfect graph [8]. When a suitable embedding of a graph into a perfect graph is given, this decomposition tree could be of use to solve NP–complete problems for the graph. Furthermore, if $G = (\mathbb{P} + \mathbb{N}, E)$ is a partitioned probe perfect graph then the CLIQUE and INDEPENDENT SET problem can both be solved in polynomial time. If $\omega(G) = \omega(G[P]) + 1$, then $\chi(G) = \omega(G)$, since we can use one additional color for $\mathbb{N}$. In the case where $\omega(G) = \omega(G[P])$ the complexity of the CHROMATIC NUMBER problem remains open; it is either $\omega(G[P])$ or $\omega(G[P]) + 1$.

2 Probe cographs

A cograph is a graph without induced P_4, i.e., an induced path with 4 vertices. By now there are many characterizations known and in the literature different characterizations of the class are used to define the class. There is a wide variety of linear time cograph recognition algorithms. To mention just a few, see, e.g., [3, 7]. For our purpose, the following portrayal of the class will make the grade [6].

Theorem 2. *1. A graph consisting of a single vertex is a cograph.*
2. Given two cographs G_1 *and* G_2*, then the* join*, obtained by making every vertex of* G_1 *adjacent to every vertex of* G_2 *is again a cograph.*

[1] A graph G is called *perfect* if for every induced subgraph the chromatic number equals its clique number. For a nice appetizer on perfect graphs we refer to [12]. The *strong perfect graph theorem* (formulated by Berge in 1960, formalizing a problem first mentioned by Shannon), states that a graph is perfect if and only if it does not contain an induced *odd* cycle of length at least 5 or the complement of such a cycle. Recently this was proved to be correct [5]. Furthermore, a polynomial time recognition algorithm for perfect graphs was announced [8]. One of the merits of perfect graphs is that some of the 'basic' NP–complete problems such as CLIQUE, INDEPENDENT SET, CHROMATIC NUMBER, and CLIQUE COVER become solvable in polynomial time when restricted to perfect graphs [1].

3. *Given two cographs* G_1 *and* G_2*, then the* union *of* G_1 *and* G_2 *is again a cograph.*
4. *There are no other cographs.*

To struck up an acquaintance with the class of probe cographs we abide the mention of some bagatelles. The proofs of these observations are included in the full version of this paper.

Remark 2. Let G be a probe cograph. Then:

a. Every induced $2K_2$ in a probe cograph remains an induced $2K_2$ in every embedding.
b. If G contains an induced P_5, say $P = [u, v, w, x, y]$, then vertices u, w, and y must be non–probes.
c. G has no induced P_k for any $k \geq 6$. It is easy to check (for example by exhaustive search) that other forbidden induced subgraphs include the C_k for any $k \geq 5$, parachute, domino, co-rising sun, the parapluie, and plenty of others. We refer to [2] for depicting these graphs. We confide that until now we haven't come up with a complete list of forbidden induced subgraphs.
d. If G is a probe cograph then its *clique-width* is at most 4. This follows from the fact that cographs have cliquewidth 2. In case a cliquewidth expression for a graph with bounded cliquewidth is available, many NP–complete problems become tractable in polynomial time. Another gap in our investigations so–far is that we still have not established a polynomial time algorithm to compute a 4–expression for probe cographs.

Recall that a graph is called *weakly chordal* if it has no induced cycle of length at least 5 or the complement of such a cycle [13].

Proposition 1. *A probe cograph is weakly chordal, hence it is perfect [13].*

In the remainder of this section we concern ourselves with the recognition problem of probe cographs.
We start with some observations that enable us to reduce the recognition problem in case the graph or its complement is disconnected.

Theorem 3. *1. Assume* $G = (V, E)$ *is disconnected. Then* G *is a probe cograph if and only if the subgraph induced by every connected component is a probe cograph.*
2. Let $G = (V, E)$ *be a graph such that the complement* $\overline{G}$ *is disconnected. Let* $C_1, C_2, \ldots, C_k$ *be the vertex sets of the components of* $\overline{G}$*. Then* G *is a probe cograph if and only if all induced subgraphs* $G[C_1], G[C_2], \ldots, G[C_k]$ *are cographs except possibly one which is a probe cograph.*

We take the partitioned case as an hors d'oeuvre, i.e., we first assume that the partition of the vertex set into $\mathbb{P}$ (probes) and $\mathbb{N}$ (non-probes) is given.

Theorem 4. *There exists a polynomial time algorithm to test if a partitioned graph* $G = (\mathbb{P} + \mathbb{N}, E)$ *is a probe cograph. This algorithm can be implemented such that it runs in* $O(n^3)$ *time.*

Proof. Let $\mathsf{G} = (\mathbb{P}+\mathbb{N}, \mathsf{E})$ be a partitioned graph. Recall that, using this notation, we assume that $\mathbb{N}$ is an independent set. We can use either linear time cograph recognition algorithm, e.g., [3, 7] to test in $O(n+m)$ time if G is a cograph. If this is the case, we are done, namely G is a cograph embedding of itself.
Assume G is disconnected and let $C_1, \ldots, C_k$ be the components of G, for some $k \geq 2$. By Theorem 3 on the facing page G is a probe cograph if and only if each $G[C_i]$ is a probe cograph (with the induced partition into probes and non–probes). By induction, we can test each component C_i in time $O(|C_i|^3)$ time. Obviously we may assume $n \geq 4$, and so $|C_1|^3 + \cdots + |C_k|^3 \leq n^3 - 2n^2$. Thus we obtain an overall $O(n^3)$ time bound.
Assume $\overline{G}$ is disconnected with components $C_1, \ldots, C_k$. By Theorem 3 on the preceding page G is a probe cograph if and only if each $G[C_i]$ is a probe cograph except possibly one, which is a probe cograph. We can use the algorithm [3, 7] to test if each component induces a cograph or not. Using an adjacency matrix for the graph G, we can find a representation for $\overline{G}$ in $O(n^2)$ time. If each component is a cograph, then we are done since, by Theorem 2 on page 3, G is a cograph. To test each component whether it is a cograph or not we use either algorithm of, e.g., [3, 7] again which gives us an overall time bound of $O(n^2)$. Also, when there are more than one component which are *not* cographs, we conclude by Theorem 3 on the preceding page, that G is not a probe cograph. Otherwise, we reduce to the unique component C_i which is not a cograph and find an embedding of $G[C_i]$ as a cograph. This will give us an embedding of G as a cograph. By induction, the test if $G[C_i]$ is a probe cograph takes $O(|C_i|^3)$ time. As above, we may conclude that the overall time is $O(n^3)$.
Finally assume that G and $\overline{G}$ are disconnected. Assume that G is a probe cograph and let H be a valid embedding of G into a cograph. Since G is connected, and since H is obtained from G by adding some edges to it, also H will be connected. By Theorem 2 on page 3 H will be the *join* of two cographs H_1 and H_2. Obviously, $\overline{H}$ is disconnected and the union of $\overline{H_1}$ and $\overline{H_2}$.
Now recall the Probe Sandwich Theorem 1 on page 2, i.e., consider the graph G^* obtained from $\overline{G}$ by removing *all* edges between vertices of $\mathbb{N}$. Using the adjacency matrix we find a representation of G^* in $O(n^2)$ time. By the Probe Sandwich Theorem, G is a probe cograph if and only if G^* is a probe cograph. Since $\overline{H}$ is disconnected, and G^* is obtained from $\overline{H}$ by deleting edges between vertices of $\mathbb{N}$ in $\overline{H}$, also G^* will be disconnected. Let $C_1, \ldots, C_k$ be the components of G^* and check if each is a probe cograph. By induction each component can be tested in $O(|C_i|^3)$ time, so reasoning as before, we conclude that the overall time will be $O(n^3)$. □

Remark 3. Notice that the proof shows that each probe cograph has a decomposition tree where the leaves are the vertices of the graph and the internal nodes are of *three* types. Either a join node, a union node, or a "sandwich join" node. This decomposition tree can be built in $O(n^3)$ time.

Remark 4. Unfortunately, we have been unable so–far to reduce the time bound of the algorithm to a linear time algorithm. Our gut feeling tells us that such an algorithm exists.

In the rest of this section we show that there is a polynomial time algorithm that finds a suitable non-probe set $\mathbb{N}$ for an unpartitioned probe cograph.

Theorem 5. *Let* G *and* $\overline{G}$ *be connected and assume that* G *is not a cograph. Then* G *is a probe cograph if and only if there are two non–adjacent vertices* x *and* y *in* G *such that* G *is a probe cograph with probe set* $\mathbb{P} = N(x) + N(y)$ *and non-probe set* $\mathbb{N} = V - \mathbb{P}$.

Proof. Assume G is a probe cograph with an embedding G'. We may assume that G' is the *join* of two graphs G'_1 and G'_2 since G' is obtained from G by adding some edges in $\mathbb{N}$ and G is connected.
Let $\mathbb{N}_i$ and $\mathbb{P}_i$ be the probes and non-probes in G'_i, $i = 1, 2$. We may assume $\mathbb{N}_1 \neq \varnothing$ and $\mathbb{N}_2 \neq \varnothing$ since $\overline{G}$ is connected. Take $x \in \mathbb{N}_1$ and $y \in \mathbb{N}_2$. □

Theorem 6. *There is an* $O(n^5)$ *time algorithm to test if* G *is a probe cograph.*

Proof. Using Theorem 3 we can reduce the problem to the case where both G and $\overline{G}$ are connected. By Theorem 5 there exists a collection of $O(n^2)$ feasible partitions. Each can be tested using the algorithm of Theorem 4 on page 4 in $O(n^3)$ time. □

3 Probe P_4–reducible graphs

Definition 2. *A graph* G *is* P_4–reducible *if every vertex belongs to at most one induced* P_4 *of* G.

These graphs were defined in [16] and in this paper appears also the following characterization which led to a linear time recognition algorithm.

Theorem 7. *A graph* G *is* P_4*–reducible if and only if for every induced subgraph* H *of* G*, exactly one of the following conditions is satisfied:*

1. *either* H *or* $\overline{H}$ *is disconnected, or*
2. *there is a unique* P_4*,* $P = [a, b, c, d]$ *in* H *such that every vertex of* $H - P$ *is adjacent to* b *and* c *and non-adjacent to* a *and* d.

Remark 5. Notice that the class of P_4–reducible graphs is self–complementary. Probe P_4–reducible graphs are weakly chordal, hence perfect. This follows since a P_4–reducible graph has no induced house.

We first consider the partitioned case of a graph $G = (\mathbb{P} + \mathbb{N}, E)$, where some partition of V into probes $\mathbb{P}$ and non-probes $\mathbb{N}$ is given.

Definition 3. *An* ornament *in a graph* G *is an induced* P_4*,* $P = [a, b, c, d]$ *such that every vertex of* $G - P$ *is adjacent to* b *and to* c *and non-adjacent to* a *and to* d.

Lemma 1. *Let* $G = (\mathbb{P} + \mathbb{N}, E)$ *be a partitioned graph. Assume* P *is a set of 4 vertices in* G *that can be made into an ornament of* G *by adding some edges between vertices of* $\mathbb{N}$.[2] *Then* G *is a probe* P_4*–reducible if and only if* $G - P$ *is probe* P_4*–reducible.*

Proof. Assume G is probe P_4–reducible. Since $G - P$ is an induced subgraph, also $G - P$ is probe P_4–reducible.
If $G - P$ is probe P_4–reducible, then add edges incident with vertices of $P \cap \mathbb{N}$ to make P an ornament. Also add edges between vertices of $\mathbb{N} - P$ to make $G - P$ P_4–reducible. It's easy to check that every vertex is then in at most one P_4. □

The next lemma deals with the case that G can be embedded into a P_4–reducible graph G' such that $\overline{G'}$ is disconnected.

Lemma 2. *Let* $G = (\mathbb{P} + \mathbb{N}, E)$ *be connected partitioned graph. Let* G^* *be the graph obtained from* $\overline{G}$ *by deleting all edges between vertices of* $\mathbb{N}$. *Let* $C_1, \ldots, C_k$ *be the components of* G^**, and let*

$$\mathbb{N}_i = C_i \cap \mathbb{N} \ \wedge \ \mathbb{P}_i = C_i \cap \mathbb{P}$$

Then G *is probe* P_4*–reducible if and only if each* $G[\mathbb{P}_i + \mathbb{N}_i]$ *is probe* P_4*–reducible with a vertex partition into probes* $\mathbb{P}_i$ *and non-probes* $\mathbb{N}_i$*.*

Theorem 8. *There exists a polynomial time algorithm that checks if a graph* $G = (\mathbb{P} + \mathbb{N}, E)$ *is a partitioned probe* P_4*–reducible graph.*

Proof. If G is disconnected, then G is probe P_4–reducible if and only if every component is P_4–reducible.
If $G = (\mathbb{P} + \mathbb{N}, E)$ is a connected probe P_4–reducible graph, then either Lemma 1 or Lemma 2 with $k \geq 2$ applies. This can be tested in polynomial time. If neither lemma applies then G is not a partitioned probe P_4–reducible graph. □

We now describe the recognition algorithm for the unpartitioned case. We propose a polynomial number of feasible non-probe sets.

i. Assume that G' is a *disconnected* embedding. Then G is also disconnected, and G is probe P_4–reducible if and only if every component is P_4–reducible. Henceforth assume that G is connected.
ii. Assume that G' is an embedding such that $\overline{G'}$ is disconnected. Identical to the proof of Theorem 5 it can be shown that in this case there must exist non-adjacent vertices x and y in G such that choosing $\mathbb{P} = N(x) + N(y)$ and $\mathbb{N} = V - \mathbb{P}$ provides a valid partition.
iii. Assume that G' is an embedding with an ornament $P = [a, b, c, d]$. Notice that, if G *itself* has some ornament P', then G is probe P_4–reducible if and only if $G - P'$ is P_4–reducible. Hence, we may assume that P is not an ornament of G. Since G is connected, we may assume that $a, d \in \mathbb{P}$ in some valid partition if this exists. We consider three possibilities:

[2] It may be necessary to add edges between vertices of $P \cap \mathbb{N}$ and between vertices of $P \cap \mathbb{N}$ and $(G - P) \cap \mathbb{N}$.

a. There is a valid partition with $b \in \mathbb{P}$ and $c \in \mathbb{N}$. Hence b is adjacent in G to all vertices of $V - P$ and $\mathbb{N} = (V - P - N(c)) + c$.
b. There is a valid partition with $b, c \in \mathbb{N}$. Hence b and c are not adjacent in G. Then $N(b) - P = N(c) - P = \mathbb{P} - \{a, d\}$.
c. If $b, c \in \mathbb{P}$, then P is an ornament in G which contradicts our assumption.

By the discussion above we obtain:

Theorem 9. *There exists a polynomial time algorithm to test if a graph G is a probe P_4–reducible graph and produce a valid embedding if this is the case.*

4 Probe P_4–sparse graphs

Hoàng introduced P_4–sparse graphs [14].

Definition 4. *A graph G is* P_4–sparse *if no set of 5 vertices induces more than one P_4.*

In [17] Jamison and Olariu characterized P_4–sparse graphs using spiders.

Definition 5. *A graph G is a* spider *if there is a partition of the vertices into three sets S, K, and R, satisfying:*

1. *S is a stable set and K is a clique and $|S| = |K| \geq 2$,*
2. *Every vertex of R is adjacent to every vertex of K and to no vertex of S,*
3. *there is a bijection f between S and K such that* either $\forall_{x \in S} N(x) = \{f(x)\}$, or $\forall_{x \in S} N(x) = K - f(x)$. *In the first case, G is called a* thin *spider and in the second case G is a* thick *spider.*

The set R is called the head *of the spider.*

Theorem 10 ([17]). *A graph is P_4–sparse if and only if for every induced subgraph H exactly one of the following conditions is satisfied:*

1. *either H or $\overline{H}$ is disconnected, or*
2. *H is isomorphic to a spider.*

Notice that the class of P_4–sparse graphs is self–complementary and properly contains the P_4–reducible graph. For a forbidden induced subgraph characterization of P_4–sparse graphs we refer to [17]. A decomposition tree for P_4–sparse graphs can be obtained in linear time [18].
Notice that Theorems 3, 5, and Lemma 2 can be "translated" into similar statements for P_4–sparse graphs. Due to space limitations we have to omit the precise description.

In the following lemma we show how to check if a partitioned graph G can be embedded into a *thin spider* by adding some edges in $\mathbb{N}$. The proof is included in the full version.

Lemma 3 (Thin spider embedding). *Assume* $G = (\mathbb{P} + \mathbb{N}, E)$ *is connected and partitioned.*

1. *Assume there exists a vertex* $x \in \mathbb{P}$ *and a* pendant *vertex* $y \in N(x)$ *such that* V *partitions into a set* $S = y + (V - N[x])$ *of pendants,* $K = N(S)$*, and* $R = V - K - S$*, and assume that this partition can be completed into a thin spider by adding edges to* $\mathbb{N}$*. Then* G *is a probe thin spider if and only if* $G[R]$ *is probe* P_4*–sparse with the induced partition into probes and non–probes.*
2. *Let* S *be the set of pendant vertices that are probes. Let* $K = N(S)$ *and* $R = V - K - S$*. Assume* G *can be embedded into a thin spider with this partition. Then* G *is a probe thin spider if and only if* $G[R]$ *is probe* P_4*–sparse with the induced partition.*
3. *Otherwise,* G *is not a probe thin spider.*

Lemma 4 (Thick spider embedding). *Let* $G = (\mathbb{P} + \mathbb{N}, E)$ *be a partitioned graph. Let* G^* *be the graph obtained from* $\overline{G}$ *be removing all edges between vertices of* $\mathbb{N}$*. Then* G *is a probe thick spider if and only if* G^* *is a probe thin spider.*

Proof. The class of P_4–sparse graphs is self–complementary. The complement of a thin spider is a thick spider. The claim now follows immediately from the Probe Sandwich Theorem 1. □

Theorem 11. *There exists a polynomial time algorithm that checks if a partitioned graph* $G = (\mathbb{P} + \mathbb{N}, E)$ *is probe* P_4*–sparse.*

A proof is included in the full version of this paper. Thus we've arrived at the unpartitioned recognition problem.

Theorem 12. *There exists a polynomial time algorithm to test if a graph* G *is a probe* P_4*–sparse graph and produce a valid embedding if this is the case.*

Proof. The cases where G or $\overline{G}$ is disconnected are easily dealt with. Likewise, the seeking after an embedding G' such that $\overline{G'}$ is disconnected can also efficiently be dealt with.
The proofs of Lemmas 3 and 4 show that only a polynomial number of partitions into a probe–clique K, –stable set S, and –head R need to be checked. □

5 Probe split graphs

Recall that a graph $G = (V, E)$ is a *split graph* if its vertices can be partitioned into a clique C and an independent set S. We use $G = (C, S, E)$ to denote a split graph.

Lemma 5. *Probe split graphs are perfect.*

We notate a partitioned probe split graph as $G = (C, S, \mathbb{N}, E)$ where $\mathbb{N}$ is the set of non–probes. Let $\mathbb{N}_C$ (respectively, $\mathbb{N}_S$) denote the non–probe sets in C and S respectively.

Observe that if $G = (C, S, \mathbb{N}, E)$ is a partitioned probe split graph, then there exists an embedding without edges between $\mathbb{N}_C$ and $\mathbb{N}_S$. Thus, to recognize a probe split graph, it is sufficient to hit upon $\mathbb{N}_C$ and add the necessary edges to turn this into a clique.
Let $G = (C, S, \mathbb{N}, E)$ be a partitioned probe split graph. Obviously, the graph $G[\mathbb{N}_C + S]$ is bipartite with color classes $\mathbb{N}_C$ and S.

Theorem 13. *A graph* G *is a probe split graph if and only if* G *is a split graph or if there exists a clique* K *and one vertex* $\kappa \in K$ *such that* $G - K$ *is bipartite and there is a bipartition of* $G - K$ *such that every vertex of one color class is adjacent to every vertex of* $K - \kappa$ *and not adjacent to* κ.

The recognition algorithm for probe split graphs G is now lucid. We assume that G is not a splitgraph, then $\mathbb{N}_C \neq \varnothing$. If G is bipartite (i.e., if $\omega(G) = \chi(G) \leq 2$) then we complete it into a split graph by making a clique of one of the two color classes.
If $\omega(G) \geq 3$, then consider a maximal clique K of size at least 3. Then K consists of $\mathbb{P}_C$ and either one vertex $\kappa \in \mathbb{N}_C$, or one vertex $s \in S$, or both. We may try all possible additions and/or deletions of vertices κ and s. It's easy to see that the desired partition of the bipartite remaining part as requested in Theorem 13 can be found in polynomial time, if it exists. This proves the following theorem.

Theorem 14. *There exists a polynomial time algorithm to recognize probe split graphs.*

Conclusion

We break off burning the midnight oil with the following concluding remarks. A write–up is in progress for probe graphs of some other self–complementary classes of graphs, such as the classes of permutation graphs and weakly chordal graphs. Serious investigations of the Probe Perfect Graph Conjecture are under way.

Acknowledgments

We thank Ross McConnell for preliminary discussions concerning some of the topics of this paper during our visit at the Université de Metz. We thank Gérard Chang for fruitful discussions concerning several parts of this paper. Finally we are grateful to Alexander Schrijver for ongoing discussions concerning the probe perfect graph conjectures.

References

1. Berge, C. and C. Chvatal, *Topics on perfect graphs*, Annals of Discrete Mathematics **21**, 1984.

2. Brändstadt, A., V. B. Le, and J. P. Spinrad, *Graph classes: A survey* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999. See also: `http://wwwteo.informatik.uni-rostock.de/isgci/`
3. Capelle, Christian, Alain Cournier, and Michel Habib, Cograph recognition algorithm revisited and online induced P_4–search. Rapport technique 94073, LIRMM, (1994).
4. Chang, G. J., A. J. J. Kloks, J. Liu, and S. L. Peng, The PIGs full monty–a floor show of minimal separators, to appear in STACS'2005, LNCS 3404.
5. Chudnovsky, M., P. Seymour, N. Robertson, and R. Thomas, The strong perfect graph theorem. Manuscript 2002.
6. Corneil, D. G., H. Lerchs, and L. Stewart–Burlingham, Complement reducible graphs, *Discrete Applied Mathematics* **3**, (1981), pp. 163–174.
7. Corneil, D. G., Y. Perl, and L. K. Stewart, A linear recognition algorithm for cographs, *SIAM Journal on Computing* **14**, (1985), pp. 926–934.
8. Cornuéjols, Gérard, Xinming Liu, and Kristina Vušković, A polynomial algorithm for recognizing perfect graphs, *Proceedings of the 44^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, 2003.
9. Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
10. Golumbic, M. C. and M. Lipshsteyn, Chordal probe graphs (extended abstract), *Proceedings WG'2003*, LNCS 2880, (2003), pp. 249–260.
11. Golumbic, M. C. and Ann N. Trenk, *Tolerance Graphs*, Cambridge studies in advanced mathematics 89, 2004.
12. Grötschel, Martin, Characterizations of perfect graphs, *Mathematical Programming Society Newsletter* **62**, (1999).
13. Hayward, R. B., Weakly triangulated graphs, *Journal of Combinatorial Theory, series B* **39**, (1985), pp. 200–208.
14. Hoàng, C. T., *Perfect graphs*, PhD thesis, School of Computer Science, McGill University, Montreal 1985.
15. Hougardy, Stefan, Inclusions between classes of perfect graphs. Preprint, Institut für Informatik, Humboldt–Universität zu Berlin, (1998). See: `http://www.informatik.hu-berlin.de/ hougardy/paper/classes.html.`
16. Jamison, B. and S. Olariu, P_4–reducible graphs–a class of uniquely tree representable graphs, *Studies in Appl. Math.* **81**, (1989), pp. 79–87.
17. Jamison, B. and S. Olariu, A unique tree representation for P_4–sparse graphs, *Discrete Applied Mathematics* **35**, (1992), pp. 115–129.
18. Jamison, B. and S. Olariu, Recognizing P_4–sparse graphs in linear time, *SIAM Journal on Computing* **21**, (1992), pp. 381–406.
19. Johnson, J. L. and J. Spinrad, A polynomial time recognition algorithm for probe interval graphs, *Proceedings* 12^{th} *ACM–SIAM Symposium on Discrete Algorithms* (2001), pp. 477–486.
20. Lovász, L., A characterization of perfect graphs, *Journal of Combinatorial Theory Series B* **13**, (1972), pp. 95–98.
21. McConnell, R. M. and J. Spinrad, Construction of probe interval graphs, *Proceedings* 13^{th} *ACM–SIAM Symposium on Discrete Algorithms* (2002), pp. 866–875.
22. McMorris, F.R., Chi Wang, and P. Zhang, On probe interval graphs, *Discrete Applied Mathematics* **88**, (1998), pp. 315–324.
23. Zhang, P, E. A. Schon, S. G. Fisher, E. Cayanis, J. Weiss, S. Kistler, and P. E. Bourne, An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA, *CABIOS* **10**, (1994), pp. 309–317.