RECOGNIZING DART-FREE PERFECT GRAPHS

V. Chvátal
Dept. of Computer Science, Rutgers University
New Brunswick, NJ 08903, U.S.A.
chvatal@cs.rutgers.edu

J. Fonlupt
Équipe Combinatoire, Université Pierre et Marie Curie
U.F.R. 921, 75252 Paris Cedex 05, France
Jean.Fonlupt@ecp6.jussieu.fr

L. Sun
(Deceased)

A. Zemirline
Dépt. d'Informatique, Université de Bretagne Occidentale
BP 809, 29285 Brest Cedex, France
zemirlin@univ-brest.fr

ABSTRACT

A graph G is called *Berge* if neither G nor its complement contains a chordless cycle whose length is odd and at least five; what we call a *dart* is the graph with vertices a, b, c, d, e and edges ab, ac, ad, ae, be, ce; a graph is called *dart-free* if it has no induced subgraph isomorphic to the dart. We present a polynomial-time algorithm to recognize dart-free Berge graphs.

1 INTRODUCTION

Claude Berge proposed to call a graph *perfect* if, for each of its induced subgraphs F, the chromatic number of F equals the largest number of pairwise adjacent vertices in F. This notion has received a considerable attention: see, for instance, Golumbic (1980) or Chvátal and Berge (1984).

It is not known whether perfect graphs can be recognized in polynomial time. However, polynomial-time algorithms for recognizing certain restricted classes of perfect graphs have been designed. One popular way of creating restricted classes of perfect graphs is to choose a graph F and to consider only F*-free* perfect graphs, meaning perfect graphs G such that no induced subgraph of G is isomorphic to F. In particular, let us consider the following three choices of F:

- the *claw* consists of vertices a, b, c, d, and edges ab, ac, ad,
- the *diamond* consists of vertices a, b, c, e and edges ab, ac, ae, be, ce,
- the *dart* consists of vertices a, b, c, d, e and edges ab, ac, ad, ae, be, ce.

Chvátal and Sbihi (1988) designed a polynomial-time algorithm to recognize claw-free perfect graphs; Fonlupt and Zemirline (1987) designed a polynomial-time algorithm to recognize diamond-free perfect graphs; we are going to outline a polynomial-time algorithm to recognize dart-free perfect graphs. Since both the claw and the diamond are induced subgraphs of the dart, our algorithm generalizes both the Chvátal-Sbihi algorithm and the Fonlupt-Zemirline algorithm; however, it uses the the Chvátal-Sbihi algorithm as a subroutine. It has developed in stages from the work of Fonlupt and Zemirline (1986, 1987), Zemirline (1987), Chvátal and Sun (1989), and Sun (1990); its detailed description and analysis are presented in Chvátal et al. (1999).

A *hole* is a chordless cycle with at least four vertices; an *antihole* is the complement of a hole; holes and antiholes are called *even* or *odd* according to the parity of their number of vertices. Berge conjectured that a graph is perfect if and only if it contains (as an induced subgraph) no odd hole and no odd antihole. This conjecture is known as the *Strong Perfect Graph Conjecture*; its "only if" part is trivial but its "if" part remains open after more than three decades. Following Chvátal and Sbihi (1988), we call a graph *Berge* if it contains no odd hole and no odd antihole. Sun(1991) has shown that dart-free Berge graphs are perfect; to put it differently, the class of dart-free perfect graphs equals the class of dart-free Berge graphs.

2 THEOREMS

Many theorems elucidate the structure of objects in some class $\mathcal{C}$ in terms of some proper subclass $\mathcal{C}_0$ of $\mathcal{C}$: they assert that, unless an object in $\mathcal{C}$ is primitive in the sense of belomging to $\mathcal{C}_0$, it must have a structural fault of a prescribed type. We follow this paradigm in three iterations.

First, let us call a graph *friendly* if, for each of its vertices x such that x is the center of a claw, the subgraph of G induced by the neighbors of x consists of vertex-disjoint cliques. Trivially, every friendly graph is dart-free; our first theorem specifies structural faults appearing in every dart-free graph that is not friendly. One of these faults is the presence of *adjacent twins,* meaning two adjacent vertices such that no vertex distinct from both is adjacent to precisely one of them.

THEOREM 2.1 *Unless a dart-free graph is friendly, it has at least one of the following properties:*

(i) it is disconnected,

(ii) its complement is disconnected,

(iii) it contains adjacent twins. □

Next, let us call a graph a *bat* if it consists of a chordless path $a_1a_2 \ldots a_m$ and an additional vertex z that is adjacent to a_1, a_i, a_{i+1}, a_m for some i with

$3 \leq i \leq m-3$ and to no other a_j. There are infinitely many bats; we call a graph *bat-free* if it contains none of them (as an induced subgraph). Our second theorem specifies structural faults appearing in every friendly Berge graph that is not bat-free. One of these faults is a clique-cutset; to describe the other, we need a few definitions. First, by a *z-edge* we mean any edge whose endpoints are both adjacent to a vertex z. Next, for any graph G and any vertex z of G, we let $G{*}z$ denote the graph obtained from $G-z$ by removing all the z-edges. Finally, we say that G has a *rosette* centered at z if $G{*}z$ is disconnected and the subgraph of G induced by all the neighbors of z consists of vertex-disjoint cliques.

THEOREM 2.2 *Unless a friendly Berge graph is bat-free, it has at least one of the following properties:*

(i) it has a clique-cutset that is a maximal clique,

(ii) it has a rosette. □

In terms of G, the two notions of a clique-cutset and a rosette may appear unrelated; in terms of a certain graph which we call *the clique graph of G,* they are nearly dual to each other. The clique graph of G is bipartite; its white vertices are the vertices of G and its red vertices are the maximal cliques in G (here, as usual, "maximal" is meant with respect to set-inclusion rather than size); a white vertex z is adjacent to a red vertex C if and only if $z \in C$. Trivially, the removal of a red vertex C and all its neighbors disconnects the clique graph of G if and only if $G-C$ is disconnected; trivially, the removal of a white vertex z and all its neighbors disconnects the clique graph of G if and only if $G{*}z$ is disconnected.

Incidentally, Theorem 2.2 is closely related to the following theorem of Conforti and Rao (1992):

> *If a bipartite graph H containing no cycle of length four and no chordless cycle of length congruent to 2 modulo 4 contains a cycle of length*

congruent to 2 modulo 4, then H has a vertex v such that the removal of v and all its neighbors disconnects H.

Let us sketch a derivation of the Conforti-Rao theorem from a weaker form of Theorem 2.2, where "friendly" is replaced by "diamond-free".

Given a bipartite graph H that satisfies the hypothesis of the Conforti-Rao theorem, color the vertices in one part of H red, color the vertices in the other part of H white, and consider the graph G defined as follows: the vertices of G are the white vertices of H and two vertices of G are adjacent if and only if they have a common neighbor in H. Since H contains no chordless cycle of length six, H is the clique graph of G; since H contains no cycle of length four, G is diamond-free; since H contains no chordless cycle of length congruent to 2 modulo 4, G contains no odd hole. The shortest cycle of length congruent to 2 modulo 4 in H has precisely one chord (cf. Lemma 3.1 of Conforti and Rao (1992)) and induces a bat in G. Now Theorem 2.2 (with "friendly" replaced by "diamond-free") guarantees that H has a vertex v such that the removal of v and all its neighbors disconnects H.

Finally, by a *separator* in a graph G we mean any cutset consisting of at most two vertices except cutsets S such that S consists of two nonadjacent vertices, $G - S$ has precisely two components, and one of these components consists of a single vertex.

THEOREM 2.3 *Unless a bat-free friendly Berge graph is bipartite or claw-free, it has a separator.* □

3 THE ALGORITHM

Let us begin with an algorithm TEST(G) that, given any bat-free friendly graph G, returns *true* if and only if G is Berge. More generally, given any friendly graph G, the algorithm

- returns *true* whenever G is Berge and bat-free and
- returns *false* whenever G is not Berge;

if G is Berge and contains an induced bat, then the algorithm returns either *true* or *false*. TEST(G) is recursive and evolves from Theorem 2.3.

It uses a procedure DECOMP1 (G;S) that, given any friendly graph G and a separator S in G, either finds an odd hole in G at once or else construct friendly graphs G_1 and G_2 such that

- if G is Berge and bat-free,
  then both of G_1 and G_2 are Berge and bat-free,
- if G is not Berge,
  then at least one of G_1 and G_2 is not Berge.

DECOMP1 (G;S) goes as follows. By definition, the set of vertices of $G-S$ splits into disjoint nonempty sets V_1 and V_2 such that no edge of G joins a vertex in V_1 to a vertex in V_2; if S consists of two nonadjacent vertices then each V_i includes at least two vertices. Let F_i denote the subgraph of G induced by $V_i \cup S$. If S is a clique then DECOMP1 ($G;S$) simply returns F_1 and F_2. If S consists of nonadjacent vertices u and v then we may assume that both $G-u$ and $G-v$ are connected (else a one-point separator could be used in place of S) and find a chordless path P_i from u to v in each F_i; now if both P_i have an odd number of edges then DECOMP1 ($G;S$) returns graphs G_1 and G_2 such that each G_i is F_i with one additional edge, uv; if both P_i have an even number of edges then DECOMP1 ($G;S$) returns graphs G_1 and G_2 such that each G_i is F_i with one additional vertex, w, and two additional edges, uw and vw; if one P_i has an odd number of edges and the other P_i has an even number of edges then DECOMP1 ($G;S$) returns the odd hole $P_1 \cup P_2$.

TEST(G):

if G is bipartite **then** return *true* **end**
if G is claw-free
then if G is Berge **then** return *true* **else** return *false* **end**
end
if G has no separator **then** return *false* **end**
S = separator in G;
if DECOMP1($G; S$) returns graphs G_1 and G_2
then return TEST(G_1)$\wedge$TEST(G_2);
else return *false*;
end

We prove that, given an input graph with n vertices, TEST calls itself recursively fewer than $4n$ times; it follows that TEST(G) runs in polynomial time.

Next, let us outline an algorithm BERGE(G) that, given any friendly graph G, returns *true* if and only if G is Berge. This algorithm is also recursive; it evolves from Theorem 2.2.

If G, the input of BERGE(G), contains a clique-cutset C then we call a procedure DECOMP2($G; C$) which goes as follows. Let $F_1, F_2, \ldots, F_k$ be the components of $G - C$; let G_i denote the subgraph of G induced by $F_i \cup C$; return $G_1, G_2, \ldots, G_k$.

If G contains a rosette centered at z (but G contains no clique-cutset) then we call a more complicated procedure DECOMP3($G; z$) which goes as follows: Let $F_1, F_2, \ldots, F_k$ be the components of $G * z$; let G_i denote the subgraph of G induced by $F_i \cup \{z\}$. Let N denote the subgraph of G induced by all the neighbors of z; let H be the graph obtained from N by adding pairwise non-adjacent vertices $w_1, w_2, \ldots, w_k$ along with all the edges xw_i such that $x \in N \cap F_i$. Return $G_1, G_2, \ldots, G_k$ and H.

Given a bat consisting of a chordless path $a_1 a_2 \ldots a_m$ and an additional vertex z adjacent to a_1, a_i, a_{i+1}, a_m, we refer to z as its *head* and to a_1, a_m as its *wing-tips*. A bat in a friendly graph G is called *fragile* if its wing-tips belong to distinct connected components of the graph arising from G by the removal of the unique maximal clique of G that contains the triangle of the bat. A straightforward polynomial-time procedure NOFB$(G; z)$, given any graph G along with a vertex z of G such that the neighborhood of z in G consists of vertex-disjoint cliques, returns *true* if and only if G contains no fragile bat with head z.

BERGE(G):

if G has a clique-cutset, C
then $G_1, G_2, \ldots, G_k =$ the output of DECOMP2$(G; C)$;
return BERGE$(G_1) \wedge$ BERGE$(G_2) \wedge \ldots \wedge$ BERGE(G_k);
end
if G has a rosette, centered at some vertex z
then $G_1, G_2, \ldots, G_k, H =$ the output of DECOMP3$(G; z)$;
return BERGE$(G_1) \wedge$ BERGE$(G_2) \wedge \ldots \wedge$ BERGE(G_k)
$\wedge$TEST(H)
$\wedge$NOFB$(G_1; z) \wedge$ NOFB$(G_2; z) \wedge \ldots$ NOFB$(G_k; z)$;
end
return TEST(G);

We prove that, as long as G is friendly, BERGE(G) returns *true* if and only if G is Berge. In addition, We prove that, given an input graph with n vertices, BERGE calls itself recursively fewer than n^2 times; it follows that BERGE(G) runs in polynomial time.

Finally, Theorem 2.1 reduces the task of recognizing dart-free Berge graphs in polynomial time to the task of recognizing friendly Berge graphs in polynomial time: given any dart-free graph H, we can find in polynomial time a family **F** of pairwise vertex-disjoint friendly induced subgraphs of H

such that H is Berge if and only if all the members of $\mathbf{F}$ are Berge. To initialize, we set $\mathbf{F} = \{H\}$. While some member G of $\mathbf{F}$ has one of the properties (i), (ii), (iii) of Theorem 2.1, we replace it in $\mathbf{F}$ by its connected components (in case G is disconnected) or by graphs $G_1, G_2, \ldots G_k$ such that $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_k$ are connected components of $\overline{G}$ (in case $\overline{G}$ is disconnected) or by $G - w$ such that w and some other vertex of G are adjacent twins.

References

[1] C. BERGE and V. CHVÁTAL (1984). *Topics on Perfect Graphs*, North-Holland Mathematics Studies 88 (Annals of Discrete Math. 21).

[2] V. CHVÁTAL, J. FONLUPT, L. SUN, and A. ZEMIRLINE (1999), "Recognizing dart-free perfect graphs", submitted to SIAM Journal on Computing.

[3] V. CHVÁTAL and N. SBIHI (1988), "Recognizing claw-free perfect graphs", J. Combinatorial Theory B44, 154-176.

[4] V. CHVÁTAL and L. SUN (1989), "Recognizing dart-free perfect graphs", Technical Report DCS-TR-259, Dept. Computer Science, Rutgers - The State University of New Jersey.

[5] M. CONFORTI and M.R. RAO (1992), "Structural properties and decomposition of linear balanced matrices", Math. Programming 55, 129-168.

[6] J. FONLUPT and A. ZEMIRLINE (1986), "Polynomial recognition algorithm of perfect 3-chromatic K_4-{e}-free graphs", Rapport technique RT-13, Artemis, IMAG, Grenoble.

[7] J. FONLUPT and A. ZEMIRLINE (1987), "A polynomial recognition algorithm for perfect K_4-{e}-free graphs", Rapport technique RT-16, Artemis, IMAG, Grenoble.

[8] M. C. GOLUMBIC (1980). *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York.

[9] L. SUN (1991), "Two classes of perfect graphs", J. Combinatorial Theory B53, 273-292.

[10] L. SUN (1990), "Two new classes of perfect graphs". Ph.D. thesis, Dept. Computer Science, Rutgers - The State University of New Jersey.

[11] S.H. WHITESIDES (1981), "An algorithm for finding clique cut-sets", Inform. Process. Lett. 12, 31-32.

[12] A. ZEMIRLINE (1987), Thèse d'État, Dépt de Mathématiques, USTHB, Alger, Algérie.