Polynomial time algorithms for perfect graphs

(A talk about two recent polynomial algorithms for recognizing perfect graphs due to Chudnovsky and Seymour, and Cornuéjols, Liu and Vuskovic)

Guillermo Durán[†]

[†]Dep. de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile

Basic definitions

- A clique in a graph is a complete subgraph maximal under inclusion. The cardinality of a maximum clique is denoted by $\omega(G)$.

- The chromatic number of a graph G is the smallest number of colours that can be assigned to the vertices of G in such a way that no two adjacent vertices receive the same colour, and it is denoted by $\chi(G)$.

- $\omega(G) \leq \chi(G)\ \forall G$

Perfect graphs (Berge, 1961)

- A graph G is perfect if $\omega(H) = \chi(H)$ for every induced subgraph H of G.

The Perfect Graph Theorem (Lóvasz, 1972)

A graph is perfect if and only if its complement is perfect.

The Strong Perfect Graph Theorem (Chudnovsky, Robertson, Seymour, Thomas, 2002)

A graph is perfect if and only if it contains neither induced odd cycle of length at least five nor its complement.

Polynomial (but no efficient!) recognition

Last year, two polynomial algorithms for recognizing perfect graphs were presented.

- Recognizing Berge Graphs, Chudnovsky and Seymour, 2002 (an $O(n^9)$ algorithm).
- A Polynomial Algorithm for Recognizing Perfect Graphs, Cornuéjols, Liu and Vuskovic, 2002 (an $O(n^{20})$ algorithm).

Both papers use the following joint work:

- Cleaning for Bergeness, Chudnovsky, Cornuéjols, Liu, Seymour and Vuskovic, 2002.

The strategy of cleaning was first introduced by Conforti and Rao (1992) to recognize linear balanced matrices. It was also used in recognition algorithms for balanced matrices (Conforti, Cornuéjols and Rao, 1999; Conforti, Cornuéjols, Kapoor and Vuskovic, 2001) and for even holes in graphs (Conforti, Cornuéjols, Kapoor and Vuskovic, 2002).

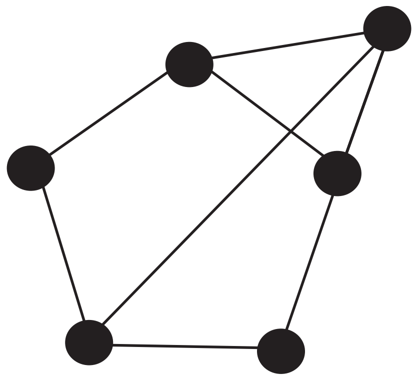
Recognizing Berge Graphs, Chudnovsky and Seymour, 2002 (an $O(n^9)$ algorithm).

Outline of the algorithm

- With input a graph G, we would like to decide either that G is not Berge, or that G contains no odd hole (and we run the algorithm on G and then again on the complement of G).

- If there is an odd hole in G, then there is a shortest one C, and C has nicer properties than a general odd hole.

- A vertex is major for C if its set of neighbours in C does not lie in a 2-edge path of C.



- The key observation is that, if there are no major vertices for C (that is, if C is clean), there is a polynomial algorithm that will discover an odd hole in G (if it exists).

- The second half of the algorithm consists of reducing the problem of testing Bergeness to the problem already solved.

- Let M be the set of all major vertices for a shortest odd hole C. If we knew the set M, we could delete it and run "the clean shortest odd hole detector" on what remains of G.

- But we do not know C, so we try to enumerate polynomially many sets $X \in V(G)$, so that if C exists then one of the sets X is guaranteed to include M and be disjoint from C (if so, the clean shortest odd hole detector to $G \setminus X$, for each set X, is applied).

There are 2 configurations we need to eliminate first: pyramids and jewels

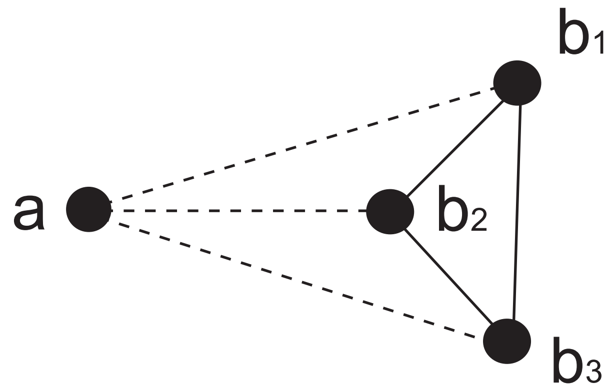
Testing for pyramids

- A pyramid in a graph G is an induced subgraph formed by the union of a triangle $\{b_1, b_2, b_3\}$, a fourth vertex a, and 3 induced paths P_1, P_2, P_3, satisfying:

  1. For $i = 1, 2, 3$, P_i is between a and b_i.

  2. For $1 \leq i < j \leq 3$, a is the only vertex in both P_i, P_j and (b_i, b_j) is the only edge of G between $V(P_i) - \{a\}$ and $V(P_j) - \{a\}$.

3. a is adjacent to at most one of b_1, b_2, b_3.



- So, every pyramid is a subdivision of K_4.

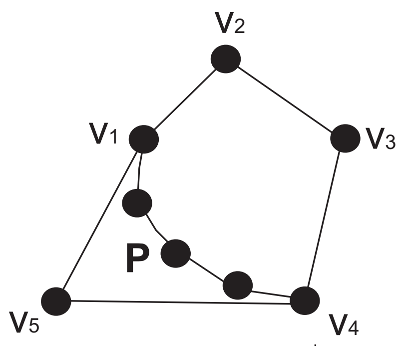
- Clearly, any graph containing a pyramid has an odd hole.

Algorithm for recognizing pyramids

- Input: A graph G
- Output: Either it finds a pyramid (and hence an odd hole) in G, or it determines that G contains no pyramid.
- Running Time: $O(n^9)$.
- The authors present some ideas trying to reduce the running time to $O(n^8)$, but none of them is enough.

Testing for jewels

- A sequence $v_1, \ldots, v_5, P$ is a jewel in G if $v_1, \ldots, v_5$ are distinct vertices; $(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_1))$ are edges; $(v_1, v_3), (v_2, v_4),$ are non edges, and P is an induced path of G between v_1 and v_4 such that v_2, v_3 and v_5 have no neighbors in the interior of P.



- Clearly, if there is a jewel in G, then there is an odd hole in G.

Algorithm for recognizing jewels

- Input: A graph G

- Output: Either it finds a jewel (and hence an odd hole) in G, or it determines that G contains no jewel.

- Running Time: $O(n^6)$.

Detecting clean odd holes

- Let C be a shortest odd hole in a graph G.
  - If x is a vertex in G, x is major for C if the set of its neighbors in C is not contained in a 2-edge path of C.
  - C is clean if no vertex is major for C.

The clean shortest odd hole detector

- Input: A graph G containing no pyramid nor jewel.

- Output: Determines one of the following:

  1. G contains an odd hole.

  2. There is no clean shortest odd hole in G.

- Running Time: $O(n^4)$ (there exists an easy implementation of $O(n^5)$).

Idea of the Algorithm

- For each u, v, two different vertices of G, find a shortest path $P(u, v)$ between them, if one exists.

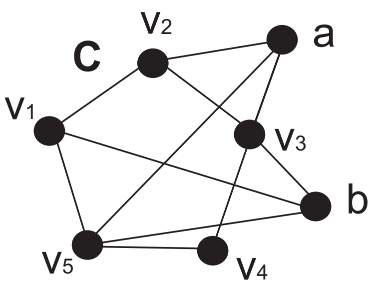
- For each u, v, w, three different vertices of G, test whether the three paths $P(u, v), P(v, w), P(u, w)$ exist, and if so whether their union is an odd hole.

- If we find such a hole, output: G contains an odd hole.

- If not, when all triples have been examined, output: There is no clean shortest odd hole in G.

A cleaner

- If C is an odd hole in G, a subset of vertices X is a cleaner for C, if X contains all vertices that are major for C, and $X \cap V(C)$ is a subset of a 2-edge path of C.



a, b: majors for C
$X = \{a, b, v_2, v_3\}$ is a cleaner

- In "Cleaning for Bergeness" (Cornuéjols, Chudnovsky, Liu, Seymour and Vuskovic, 2002), it is proved the following result:

  There is an algorithm with:

  - Input: A graph G.
  - Output: Either

  1. $O(n^5)$ subsets of vertices such that if C is a shortest odd hole in G, and every major vertex has at least 4 neighbors in C, then one of the subsets is a cleaner for C, or

  2. It determines that G is not Berge (because it finds an odd hole or an odd antihole).

- Running time: $O(n^6)$.

An algorithm for Bergeness

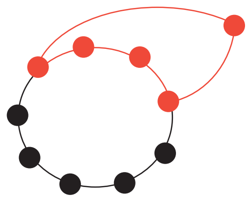
The final algorithm is the following:

- Input: A graph G.
- Output: Determines whether G is Berge.
- Running time: $O(n^9)$ (it is important to note that an easy implementation of this algorithm has a time complexity of $O(n^{12})$).

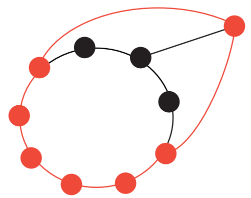
The algorithm:

1. Check there is no pyramid or jewel (if there is anyone, output: “G is not Berge”).

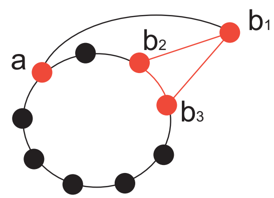
2. Run the “cleaner algorithm” and assume that $O(n^5)$ subsets of vertices X are obtained (otherwise, output: “G is not Berge”). Note that the hypothesis of that algorithm that every major vertex has at least 4 neighbors in C holds for every shortest odd hole C, since G contains no pyramid.



shorter odd hole

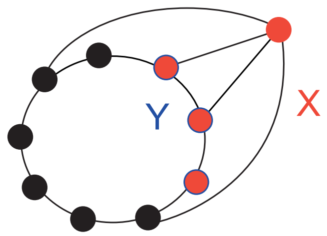


shorter odd hole



pyramid

3. For each subset X, enumerate all $Y \subseteq X$ with $|Y| \leq 3$ and apply the "clean shortest odd hole detector" to each graph $G \setminus (X \setminus Y)$.



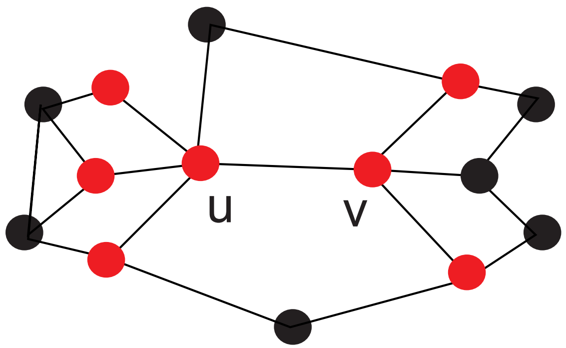
4. If G has an odd hole, then one of these subgraphs has a clean shortest odd hole, and it will be detected by the algorithm.

5. If this procedure does not find an odd hole, then run the whole thing over again on the complement.

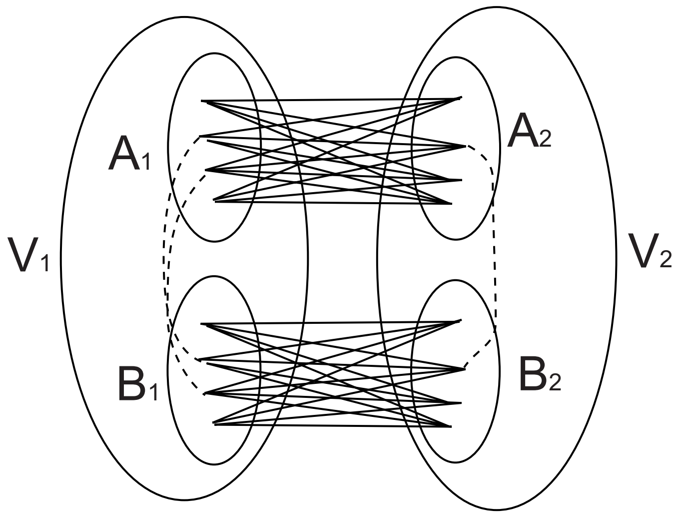
6. If an odd hole is detected in $\overline{G}$, output: "G is not Berge"; otherwise, output: "G is Berge".

A Polynomial Algorithm for Recognizing Perfect Graphs, Cornuéjols, Liu and Vuskovic, 2002 (an $O(n^{20})$ algorithm).

- A set S of vertices is a double star if S contains 2 adjacent vertices u and v such that $S \subseteq N(u) \cup N(v)$. The vertex set S is a cutset of a graph G if $G \setminus S$ contains more connected components than G.



- A graph G has a 2-join $V_1|V_2$ with special sets (A_1, B_1, A_2, B_2) if its vertices can be partitioned into sets V_1 and V_2 so that, for $i = 1, 2$, V_i contains disjoint, nonempty vertex sets A_i and B_i, such that every vertex of A_1 is adjacent to every vertex of A_2, every vertex of B_1 is adjacent to every vertex of B_2, and there are no other adjacencies between V_1 and V_2.



- A graph G is basic if it is one of the following graphs:
  - A bipartite graph or its complement.
  - A line graph of a bipartite graph or its complement.

Outline of the algorithm

- Decompose the input graph G into a polynomial number of simpler graphs $G_1, \ldots, G_m$ so that:

  1. G is odd hole free if and only if G_i is odd hole free, $\forall i = 1, \ldots, m$.

  2. It is easy to check directly whether G_i is odd hole free, $\forall i$.

- It is used the following decomposition theorem (Conforti, Cornuéjols and Vuskovic, 2001):

  If G is an odd hole free graph, then either G is basic, or G has a double star cutset or a 2-join.

- We can check in polynomial time if G is basic (easy), or G has a 2-join (Cornuéjols and Cunningham, 1985; Conforti, Cornuéjols, Kapoor and Vuskovic, 2002), or G has a double star cutset (easy).

- By the theorem, if G is not basic, and G has neither a 2-join nor a double star cutset, then it has an odd hole.

- Clearly, if G is basic, then G is perfect.

- The decomposition suggested in the work for a graph G which has a 2-join verifies (1) straightforward.

- Main difficulty: Decompose a graph G that has a double star cutset or a 2-join into blocks G_i that satisfy (1).

- In order to obtain a decomposition for a graph G which has a double star cutset verifying (1) the authors need to clean the graph.

- Running time of the whole algorithm: $O(n^{20})$.

Main open problems in perfect graphs:

- The complexity of determining if a graph G is odd hole free remains still open. These algorithms do not solve this problem, because in the cleaning process they can stop detecting an odd antihole (in this case, we do not know if the graph has an odd hole or not). Remember that it is NP-complete to recognize if a graph contains an odd hole passing through a specific vertex (Bienstock, 1991).
- Obtain an algorithm to recognize perfect graphs being polynomial and efficient.

- Obtain an algorithm to recognize perfect graphs using the decomposition theorem of the proof of The Strong Perfect Graph Theorem:

  For every Berge graph G, either G is basic, or one of G, $\overline{G}$ admits a proper 2-join, or G admits an M-join, or G admits a balanced skew partition.

- Obtain combinatorial algorithms for perfect graphs to compute in polynomial time the chromatic number.