An Algorithm To Find The Core Of A CV-Graph

A. Antonysamy
Department of Mathematics
St Joseph's (Autonomous) College
Tiruchirapalli - 620 002
India

G. Arumugam
Department of Computer Science
Madurai - Kamaraj University
Madurai - 625 021
India

C. Xavier
Department of Computer Science
St. Xavier's (Autonomous) College
Palayamkottai - 627 002
India

ABSTRACT. The core of a graph was defined by Morgan and Slater [MS80] as a path in the graph minimizing the sum of the distance of all vertices of the graph from the path. A linear algorithm to find the core of a tree has been given in [MS80]. For the general graph the problem can be shown to be NP hard using a reduction from the Hamiltonian path problem.

A graph with no chordless cycle of length exceeding three is called a chordal graph. Every chordal graph is the intersection graph of a family of subtrees of a tree. The intersection graph of a family of undirected paths of a tree is called a UV graph. The intersection graph of an edge disjoint family of paths of a tree is called a CV graph [AAPX91]. We have characterised that the CV graphs are nothing but block graphs. CV graphs form a proper subclass of UV graphs which inturn form a proper subclass of chordal graphs. In this paper an $O(ne)$ algorithm to find the core of a CV graph is given where n is the number of vertices and e is the number of edges.

ARS COMBINATORIA 46(1997), pp. 153-160

1 Introduction and definitions

The core of a graph was defined by Morgan and Slater [MS80], [SL80] as a path in the graph minimizing the sum of the distances of all vertices of the graph from the path. A linear algorithm to find the core of a tree is given in [MS80]. This algorithm traverses the edges of the tree in double scanning, first bottom-up and then top-down. One of the applications mentioned in [MS80] is routing a highway through a road network. Becker and Perl [BP85] generalise this concept and define 2-core of a graph as a set of 2 paths minimizing the sum of the distances of all vertices of the graph from the two paths. They have considered both the cases of disjoint paths and the intersecting paths which they call 2-core and I2-core respectively. The core is called the 1-core. The 1-core and the 2-core problems can be shown to be NP-hard for the general graphs from the reduction of the Hamiltonian path problem.

The following definitions and notations of [BP85] are used here.

A graph is called a block graph iff every block is complete. Let $G = (V, E)$ be a graph and P a path in G. Let $d(v_1, v_2)$ denote the number of edges in the shortest path joining vertices v_1 and v_2 in G.

Let $d(v, P) = \text{Min}(v, v') : v' \in P\}$.

Let B be a subgraph of G. We denote

$$d_B(P) = \sum_{v \in V(B)} d(v, P)$$

If $B = G$ then we simply write $d(P)$ in the place of $d_B(P)$. A 1-core of a graph G is a path P minimizing $d(P)$.

We shall have the intersection characterisation for a block graph. A graph with no chordless cycle of length exceeding three is called chordal graph. Chordal graphs are precisely the intersection graphs of subtrees of a tree [Ga74]. Given a chordal graph G, the tree can be constructed with the vertex set as the set of maximal cliques of G [MW86]. The intersection graph of paths of an undirected tree is called an undirected vertex path graph or UV graph [MW86].

A family of paths in a tree T is said to be *compact* if every edge of T is in one and only one path of the family. The intersection graph of a *compact family* of paths in a tree is called a Compact Vertex path graph or CV graph. In this paper cliques refer to only maximal cliques and all the graphs considered are connected. The CV graphs have been introduced and studied in [AAPX91].

The following results of [AAPX91] are used directly or indirectly.

Proposition 1.1.

a) *A complete graph is a CV graph.*

b) *A chordal graph with exactly two cliques is a CV graph if and only if there is exactly one point common to the cliques.*

Proposition 1.2. *A graph G is CV if and only if intersection of any two cliques is at the most a singleton and collection of cliques satisfies Helly's property.*

Proposition 1.3. *A graph G is CV graph if and only if G is block graph.*

A vertex v of G is said to be a cut vertex if the removal of v and the edges incident on v makes G disconnected. We also observe that a vertex v of a CV graph G is cut vertex if and only if v lies in more than one clique.

2 Labelling the edges

Let v and v' be two adjacent vertices of a CV graph where v' is a cut vertex. Let C denote the clique containing both v and v'. Let $C_1, C_2, \ldots, C_s$ be the cliques which contain v' but not v. Without loss of generality assume that $C_1C_2 \ldots C_q$ $(1 \leq q \leq s)$ are the cliques which contain no cut vertex other than v'. For each C_i $(q < i \leq s)$ assume that v_{ij} $(1 \leq j \leq k_i)$ be the cut vertices of C_i other than v' and v'_{ij} $(1 \leq j \leq t_i)$ be the non-cut vertices of C_i.

For $q < i \leq s$, $t_i + k_i + 1 = |C_i|$

$$p_1 = \sum_{q<i\leq s} k_i \text{ is the number of cut vertices adjacent to } v'$$

and

$$p_2 = \sum_{1\leq i\leq q} (|C_i| - 1) + \sum_{q<i\leq s} t_i \text{ is the number of non-cut vertices adjacent to } v'.$$

By deleting the edges of C (only edges, not the vertices) the CV graph G is decomposed into a number of components. These components are called the *branches* separated by C. We define two labels for each edge vv', one corresponding to v and the other to v'. Let $B_{v'}$ denote the branch containing v' separated by C. Let $L^{vv'} = (1^{vv'}, m^{vv'}, n^{vv'})$ denote the label of vv' correponding to the vertex v. We define the labels such that $n^{vv'}$ is the number of vertices in B_v and $1^{vv'} = d_{B_{v'}}(\{v\})$. $m^{vv'}$ is the maximum one can save on $d_{B_{v'}}(\{v\})$ by putting a path P through v into $B_{v'}$ and

replacing $d_{B_{v'}}(\{v\})$ by $d_{B_v}(P)$. We denote

$$L^{v'v_{ij}} = L_{ij},$$
$$1^{v'v_{ij}} = 1_{ij},$$
$$m^{v'v_{ij}} = m_{ij},$$

and

$$n^{v'v_{ij}} = n_{ij}$$

Let $L_{ij} = (1_{ij}, m_{ij}, n_{ij})$ be the label of edge $v'v_{ij}$ corresponding to v'. $(1 \leq j \leq k_i, q < i \leq s)$.

Let S be the set of all cut vertices in the clique C and

$$B = \bigcup_{u \in V(C)-\{v\}} B_u$$

where B_u is the branch containing u separated by C.

Let B_{ij} denote the branch containing v_{ij} separated by C_i. Let there be a path P up to v'.

Let there be a path P up to v'. Let m_{ij} be the distance saved on $d_B(P)$ by extending P from v' to v_{ij} after visiting all the vertice of C_i and then extending into B_{ij}.

Let

$$m_1 = \text{Max}_{1 \leq i \leq q}\{|C_i| - 1\}$$
$$m_2 = \text{Max}_{\substack{q \leq i \leq s \\ 1 \leq j \leq k_i}} \{m_{ij}\}$$

Lemma 2.1. *The number of vertices in $B_{v'}$ is given by*

$$n^{vv'} = [\sum_{q<i\leq s} \sum_{i \leq j \leq k_i} n_{ij}] + p_2 + 1 \tag{2.1}$$

Proof: If all the cliques $C_1, C_2, \ldots, C_s$ have no other cut vertex then $n^{vv'} = p_2 + 1$, which is trivially the number of elements in the branch $B_{v'}$.

By induction let us assume that n_{ij} has been already calculated which is the number of vertices in the branch with v_{ij} separated by the clique C_i $(q < i \leq s, 1 \leq j \leq k_i)$.

The number of non-cut vertices adjacent to v' is p_2. At every cut vertex v_{ij} adjacent to v' there is a branch with n_{ij} vertices; v' is also in the branch $B_{v'}$. So, the total number of vertices in $B_{v'}$ is given by (2.1).

Lemma 2.2.

$$d_{B_{v'}}(\{v\}) = \sum_{q<i\leq s} \sum_{i \leq j \leq k_i} (1_{ij} + n_{ij}) + 2p_2 + 1 \tag{2.2}$$

Proof: Let v' be the cut vertex adjacent to v. If all the cliques $C_1, C_2, \ldots, C_s$ have no cut vertex other than v then every vertex except v' is at a distance 2 from v. So, the sum of the distances of all the vertices except v' is $2p_2$. Hence the total distance $d_{B_{v'}}(\{v\}) = 2p_2+1$. Now consider the cut vertices v_{ij} adjacent to v' $(q < i \leq s, 1 \leq j \leq k_i)$. By induction hypothesis assume that 1_{ij} has already been evaluated and $1_{ij} = d_{B_{ij}}(\{v'\})$ where B_{ij} is the branch with v_{ij} separated by the clique C_i. Since v is adjacent to v'.

$$d_{B_{ij}}(\{v\}) = d_{B_{ij}}(\{v'\}) + n_{ij} = 1_{ij} + n_{ij}$$

So, the sum of the distance coming through the cut vertices is

$$\sum_{q<i\leq s} \sum_{i\leq j\leq k_i} (1_{ij} + n_{ij})$$

So, including the distances from the non-cut vertices and the distance of v', $d_{B_{v'}}(\{v\})$ is given by (2.2).

Lemma 2.3. *Let P be a path which terminates at v such that $V(P) \cap V(C) = \{v\}$. By extending v to v' after visiting all the vertices of C one can save*

$$|C| - 1 + \sum_{u\in S-\{v\}} (n^{vu} - 1) \tag{2.3}$$

on $d(P)$.

Proof: Since C is a separating clique a path which extends from v to v' gets a saving of distance one for each vertex of C other than v. For every cut vertex u of C, the branch B_u has n^{vu} vertices. Of them u has been already counted and the other vertices also get a saving of 1 and hence (2.3).

Lemma 2.4. *Let P be a path which terminates at v' and $P \cap C_i = \{v'\}$ $(1 \leq i \leq s)$. By extending P from v' and visiting all the vertices of C_i $(1 \leq i \leq q)$ one can save*

$$|C_i| - 1 \tag{2.4}$$

on $d_B(P)$.

Proof: All the vertices of C_i except v' were at a distance 1 from P and because of the extension they lie on the path P. No other vertices of B are affected.

Let us denote

$$m^{vv'} = |C| - 1 + \sum_{u\in S-\{v\}} (n^{uv} - 1) + \text{Max } (m_1, m_2)$$

Theorem 2.5. *Let P be a path which terminates at v and $V(P) \cap V(C) = \{v\}$. By extending P from v to v' after visiting all the vertices of C and then extending into B, one can save at the most $m^{vv'}$ on $d(P)$.*

Proof: The extension of the path P is done in two phases. In the first phase the path is extended from v to v' after visiting all the vertices of C and this yields a saving of

$$|C| - 1 + \sum_{\substack{u \in V(C) - \{v\} \\ \text{and } u \text{ is a cut vertex}}} (n^{vu} - 1) \text{ by Lemma (2.3)}$$

In the second phase the path is extended further. If the path is extended to clique C_i $(1 \leq i \leq q)$, one can save at the most m_1. If extended to some v_{ij} $(i \geq q)$ then one can save at the most m_2. This establishes Theorem 2.5.

3 Algorithm

The edges of the CV-graph G are given two labels one for each vertex using the following procedure

Procedure 3.1.

Input: A CV graph G

Output: For each edge vv' of G, evaluate two labels, one for v and the other for v' The label of vv' corresponding to v is denoted by $L^{vv'} = (1^{vv}, m^{vv}, n^{vv'})$.

1. For every edge vv',while v' is a non-cut vertex do

   1.a) $n^{vv'} \longleftarrow 1$

   1.b) $1^{vv'} \longleftarrow 1$

   1.c) $m^{vv'} \longleftarrow |C| - 1$ where C is the clique containing vv'.

2. Check if there is an edge vv' such that for every edge $v'u$ $(u \neq v)$ the labels $n^{v'u}$ and $1^{v'u}$ have already been evaluated but $n^{vv'}$ and $1^{vv'}$ have not.

   2.a) If the test in step-2 fails, then the labels $1^{vv'}$ and $n^{vv'}$ have been evaluated for all the edges;
   go to step-3.

   2.b) If the test in step-2 succeeds,
   $n^{vv'} \longleftarrow [\sum_{q<i\leq s} \sum_{i\leq j\leq k_i} n_{ij}] + p_2 + 1;$
   $1^{vv'} \longleftarrow \sum_{q<i\leq s} \sum_{1\leq j\leq k_i} (1_{ij} + n_{ij}) + 2p_2 + 1;$
   go to step-2.

3. Check if there is an edge vv' such that for every edge $v'u$ $(u \neq v)$ the label $m^{v'u}$ has already been evaluated but $m^{vv'}$ have not.

   3.a) If the test in step-3 fails, then the labelling process is over.

   3.b) If the test in step-3 succeeds

   3.b.1) $m_1 \longleftarrow \text{Max}_{1 \leq i \leq q}\{|C_i| - 1\}$

   3.b.2) $m_2 \longleftarrow \text{Max}_{\substack{q<i\leq s \\ 1 \leq j \leq k_i}} \{m_{ij}\}$

   3.b.3) $m^{vv'} \longleftarrow |C| - 1 + \sum_{u \in S-\{v\}} n^{vu} + \text{Max}\{m_1, m_2\}$

   3.b.4) go to step-3.

Algorithm 3.2.

Input: A CV graph G

Output: A core of G.

1. Label the edges as per procedure 3.1.
2. Choose the edge vv' which has a maximum label $m^{vv'}$
3. The path $P = \{v\}$
4. Extend P from v and reach v' after visiting all the vertices of the clique C containing v and v'.
5. In the evaluation of $m^{vv'}$

   If $m_1 = \text{Max}\{m_1, m_2\}$ and $m_1 = |C_i| - 1$ for a particular $i \leq q$ then go to step (5.a), otherwise go to step (5.b).

   5.a) Extend the path from v' into C_i, visit all the vertices of C_i and stop. The core of the CV graph has been constructed.

   5.b) If $m_2 = m_{ij}$ for some particular values i and j $(i > q$ and $i \leq j \leq k_i)$ Treating $v'v_{ij}$ as vv' go to step 4.

4 Correctness and computing time

Theorem 4.1. *Algorithm 3.2 constructs a core of a CV-graph.*

Proof: In step-2, vv' has been selected in such a way that the label $m^{vv'}$ is the maximum. As per the definition of $m^{vv'}$ this minimizes $d(P)$. The same procedure ls repeated in step-5 and hence $d(P)$ is the mlnimum.

Lemma 4.2. *The time taken by procedure 3.1 ls bounded by $O(ne)$ where e is the number of edges and n is the number of vertices.*

Proof: A vertex is a cut vertex if and only if it is in more than one clique. Hence a vertex can be checked for cut vertex in $O(p)$ time where p is the number of cliques. Hence step 1 takes at the most $O(ep)$ time. Test-2 can be performed in constant time and this is repeated $2e$ times. Step-3 takes no more time than step-2. Hence the total time taken is $O(ep)$. But p is bounded by n since a CV-graph is chordal. Hence the time taken by the procedure 3.1 is bounded by $O(ne)$.

Theorem 4.3. *The computing time of algorithm 3.2 is* $O(ne)$.

Proof: Follows from Lemma 4.2.

References

[AAPX91] A. AntonySamy, G. Arumugam, M. Paul Devasahayam, C. Xavier, A recognition Algorithm for the Intersectlon graphs of internally dlsjoint paths in trees, *Proceedings of National Seminar on Theoretical Computer Science*, I MSc Madras Report 115 (1991), 169–178.

[BP85] Ronald I. Becker and Yehoshua Perl, Finding the two-core of a tree, (personal communication).

[Ga74] F. Gavril, The intersection graphs of sub trees of a tree are exactly the chordal graphs, *J. Comb. Theory Ser. B* **16** (1974), 47–56.

[MS80] C.A. Morgan and P.J. Slater, A linear algorithm for the core of a tree, *J. Algotrithms* **1** (1980), 247–258.

[MW86] C.L. Monma and V.K. Wei, Intersection graphs of Paths in a tree, *Jour. of Comb. Theory (B)* **41**(2) (1986), 141–181.

[SL80] P.J. Slater, Centrality of paths and vertices in a graph: Cores and pits in: G. Chartrand et al. eds. 4th International Conf. Theory and Appl. Graphs, Western Michigan University, Wiley, New York, (1980), 529–542.