Optimal Spanners in Partial $k$–Trees

J.A. Makowsky*
Department of Computer Science
Technion–Israel Institute of Technology
Haifa, Israel
janos@cs.technion.ac.il

U. Rotics
IBM Scientific Center
Haifa, Israel
rotics@HAIFASC3.VNET.IBM.COM

October 1993

Abstract

Assume you have a problem $K$ which is NP–hard in general, but in P for a certain class of graphs $G$. Let $Patch_G$ be a class of graphs which are ‘patched’ together from graphs from $G$. Is membership in $K \cap Patch_G$ now in P? We show that for $K_t$, the problem of finding optimal $t$-spanners, $G$ certain families of cliques and $Patch_G$ the partial $k$–trees or the clique sequence graphs (here for 2–spanners only), this is the case. For the case of partial $k$–trees we use a method first proposed by Arnborg and Proskurowski. The clique sequence graphs are newly introduced in this paper. The choice of the problem of finding $t$–spanners was motivated by its application in distributed systems, communication networks, parallel architectures and motion planning, among others.

*Partially supported by a Grant of the French-Israeli Binational Foundation and by the Fund for Promotion of Research of the Technion–Israel Institute of Technology
1 Introduction

Given a graph $G = (V, E)$, a subgraph $G_1 = (V, E_1)$ is a $t$-spanner of $G$ if for every edge $(u, v) \in E$ the length of the shortest path between $u$ and $v$ in $G_1$ is at most $t$. In other words, a $t$-spanner of a graph $G$ is a spanning subgraph $G_1$ such that the distance in $G_1$ between every pair of vertices is at most $t$ times their distance in $G$.

In this paper we study the problem of finding an optimal $t$-spanner for a given graph $G$. By optimal we mean here a $t$-spanner with minimum number of edges. This problem has applications in distributed systems, communication networks, parallel machine architecture, motion planning and others. The problem has been studied extensively in recent years, see [PS89], [PU87], [KP92], [HPS92], [LS91], [ADDJ90], [CDNS92], [Cai92].

D. Peleg and A.A. Schäffer in [PS89] showed that the problem of determining, for a given graph $G$ and two integers $t, m$, whether $G$ has a $t$-spanner with at most $m$ edges is NP-complete. In fact, they showed that even for fixed $t$, the problem remains NP-complete. From this it follows that also our problem of finding optimal $t$-spanners is NP-complete.

We are interested in classes of graphs for which our problem has a solution in polynomial time. Trivially, the trees and the cliques form such a classes. S. Arnborg and A. Proskurowski in [AP89] have shown that many NP-hard problems, which have polynomial solution for trees, have also polynomial solutions for for partial $k$-trees. They have not shown a general theorem characterizing such problems, but have indicated a general scheme, along which many problems with polynomial solutions can be identified. The partial $k$-trees are therefore natural candidates for our situation.

Our main result, indeed, states that

THEOREM A. For fixed $k$ and $t$, the problem of finding an optimal $t$-spanner for partial $k$-trees can be solved in polynomial time.

Our proof follows the general scheme of [AP89]. However, the update procedure we use here is quite different from the update procedures they used for their problems.

It seems that the knowledge of the clique structures of the graph helps to find polynomial time algorithms for our problem. In the case of partial $k$-trees we know that the maximum clique size is at most $k+1$. Therefore we are looking for other classes of graphs, that although having a defined clique structures, do not limit the size of their cliques. As an example for such a class we introduce the clique sequence graphs, which are graphs that their cliques can be ordered in a sequence, such that only adjacent cliques can have a non-empty intersection. Indeed we can show that,

THEOREM B. The problem of finding an optimal 2-spanner for clique sequence graphs can be solved in linear time.
It happens that proving results such as Theorem B is not trivial. In our forthcoming papers we will be looking for extending our results to other types of graphs having a defined clique structures. We will try to find how far we can weaken the restrictions on the clique structures and still have polynomial time solutions.

2 Definitions

A graph is a \(k\)-tree if it satisfies either of the following conditions:

(i) It is a complete graph on \(k\) vertices, \(K_k\).

(ii) It has a vertex \(v\) with degree \(k\) with completely connected neighbors, and
the graph obtained by removing \(v\) and its incident edges is a \(k\)-tree.

A partial \(k\)-tree is a subgraph of a \(k\)-tree.

The recursive definition above defines a reduction process for \(k\)-trees. One can test a graph for being a \(k\)-tree by repeatedly removing a vertex of degree \(k\) with completely connected neighbors (a \(k\)-leaf) until no such vertex remains, then the graph is a \(k\)-tree if and only if what remains is a \(K_k\) [Ros70].

In the following we will consider a \(k\)-tree together with a reduction sequences, which is one of the many possible reduction sequences mentioned above.

A \(k\)-clique is a set of \(k\) pairwise adjacent vertices (and thus it is not a clique in the standard graph terminology, where it denotes a maximal completely connected subgraph).

If \(K\) is a \(k\)-clique, \(v \in K\) is a descendant of \(K\) in a given reduction sequence, if when \(v\) was being removed, each vertex to which it was adjacent was either a member of \(K\) or a descendant of \(K\). Note that the descendents of \(K\) change during the reduction process.

Example 2.1 Let \(G = (V, E)\), where \(V = \{1, 2, 3, 4\}\) and \(E = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4)\}\) be a partial 2-tree. Before removal of any vertex no vertex is considered as a descendant. After removal of 1, 1 is considered descendant of \(2, 4\) and no vertex is considered descendant of \(\{3, 4\}\) or \(\{2, 3\}\). After removal of 2, 1 and 2 are considered descendents of \(\{3, 4\}\).

The Connected components of the subgraph induced by the descendents of \(K\) are branches on \(K\), and \(B(K)\) denotes all the vertices in the branches of \(K\). In other words \(B(K)\) consists of all the descendents of \(K\).

The \(k\)-clique remaining after the reduction sequence completes is the root of the \(k\)-tree and will be denoted by \(R\).

Let \(G = (V, E)\) be a graph, and let \(X \subseteq V\) be any subset of \(V\). The subgraph of \(G\) induced by \(X\) is denoted \(G[X]\).

Lemma 2.2 Let \(T\) be a \(k\)-tree, and let \(\hat{T}\) be the \(k\)-tree resulted by applying several reductions from the reduction sequence associated with \(T\), then for every two different \(k\)-cliques \(K_1, K_2\) of \(\hat{T}\):
\( (i) \ B(K_1) \cap B(K_2) = \emptyset. \)

\( (ii) \) There is no edge in \( T \) between a member of \( B(K_1) \) to a member of \( B(K_2) \).

**Proof:** (i) Suppose \( B(K_1) \cap B(K_2) \neq \emptyset \), and let \( v \in B(K_1) \cap B(K_2) \). \( v \in B(K_1) \) implies that when \( v \) was removed it was adjacent just to members of \( K_1 \) or descendants of \( K_1 \). If it was adjacent to a descendant \( u \) of \( K_1 \) then \( v \in B(K_2) \) implies that \( u \) is a descendant of \( K_2 \) too. (Note that \( v \) can’t be a member of \( K_2 \), since if it is a member of \( K_2 \) it was not removed yet by the reduction sequence, and it is not a descendant yet of any \( k \)-clique, a contradiction). \( u \) was removed from the graph before \( v \), so we can continue the above argument on \( u \) until we reach a vertex \( x \in B(K_1) \cap B(K_2) \), such that when \( x \) was removed it was adjacent just to members of \( K_1 \). But \( x \in B(K_3) \) implies that when \( x \) was removed it was adjacent just to members of \( K_2 \). (Note that if it was adjacent to a descendant \( w \) of \( K_2 \) then \( w \) would not be a member of \( K_1 \), a contradiction). Thus when \( x \) was removed it was adjacent to \( k \) vertices all of them are members of \( K_1 \) and \( K_2 \). However since \( |K_1| = |K_2| = k \) it follows that \( K_1 = K_2 \), a contradiction.

(ii) Suppose that there is an edge in \( T \) between a vertex \( v \in B(K_1) \) to a vertex \( u \in B(K_2) \). Suppose, without loss of generality that \( v \) is removed by the reduction process before \( u \). When \( u \) is removed it is adjacent just to members of \( K_2 \) or to descendents of \( B(K_2) \). Therefore \( v \) is either a descendant of \( K_2 \) or a member of \( K_2 \). If \( v \) is a descendant of \( K_2 \) then \( v \in B(K_2) \), but \( v \in B(K_1) \), a contradiction to (i). If \( v \) is a member of \( K_2 \) then \( v \) can’t be a descendant of any \( k \)-clique, since it was not removed yet. This contradicts \( v \in B(K_1) \).

\( \square \)

3 Algorithm for the 2-spanner problem on partial \( k \)-trees

3.1 Description of the Algorithm

The 2-spanner problem is defined as follows: Given a graph \( G = (V, E) \) find an optimal 2-spanner for \( G \). i.e. find a subgraph \( \tilde{G} = (V, \tilde{E}) \) of \( G \), \( \tilde{E} \subseteq E \), such that the distance between any pair of vertices in \( \tilde{G} \) is at most twice their distance in \( G \), and \( \tilde{G} \) has a minimum number of edges.

Since we want to solve the above problem for partial \( k \)-trees, we assume that \( G = (V, E) \) is a partial \( k \)-tree embedded in a \( k \)-tree \( T = (V, E') \) with \( E \subseteq E' \).

The steps of the algorithm are the steps of the reduction sequence associated with the \( k \)-tree \( T \).
Let $T$ be the $k$–tree resulted after applying several reductions from the reduction sequence associated with $T$. For every $k$–clique $K$ of $T$ and every set of pairs $H$ of elements (i.e. vertices) of $K$ we define a class of solutions to the problem (i.e. the $2$–spanner problem) on the subgraph of $G$ induced by $K \cup B(K)$. This class, denoted by $C(H, K)$ consists of all the sets of pairs $H'$ of elements from $K \cup B(K)$ which satisfies the following two conditions:

(i) The intersection of $H'$ with $K$ equals $H$, i.e. the set of all pairs $(u, v)$ of $H'$ such that both $u$ and $v$ are members of $K$ equals $H$.

(ii) $H'$ defines a 2-spanner for the subgraph of $G$ induced by $K \cup B(K)$, i.e. for every edge $(x, y)$ of $G[K \cup B(K)]$ either the pair $(x, y) \in H'$ or there is a vertex $z$ of $K \cup B(K)$ such that both $(x, z) \in H'$ and $(y, z) \in H'$, and all the pairs of $H'$ corresponds to edges of $G$. (i.e. if $(x, y)$ is a pair of $H'$ then $x$ and $y$ are adjacent in $G$).

An optimal solution in the class $C(H, K)$ is any solution in the class with minimum number of edges, (i.e. minimum number of pairs).

At the beginning of the algorithm for every $k$–clique $K$ of $T$ and every set of pairs $H$ of elements of $K$, the only possible solution in the class $C(H, K)$ is $H$ itself, since $B(K) = \emptyset$. However $H$ is a solution only if it satisfies the 2-spanner condition, see condition ii above. So, either that class $C(H, K)$ is empty, or it has one solution $H$.

The major concern of the algorithm is to keep an optimal solution for each class $C(H, K)$, where $K$ is a $k$–clique of $T$, and $H$ is a set of pairs of elements of $K$, and to update these optimal solutions during the reduction process. If these updates could be done then we could derive an optimal solution (i.e. an optimal 2-spanner in $G$) as follows:

First, by the end of the algorithm the $k$–tree $T$ is reduced to the root $k$–clique $R$ and the subgraph of $G$ induced by $R \cup B(R)$ is $G$ itself. Second, by the end of the algorithm we have for each set of pairs $H$ of elements of $R$ an optimal solution for the class $C(H, R)$. Third, each solution to the problem in $G$ belongs to some class $C(H, R)$, for some set of pairs $H$ of elements of $R$. Therefore the optimal solution to the problem in $G$ is the minimum of the optimal solutions for the classes $C(H, R)$, where $H$ is any set of pairs of elements of $R$. Note that there are many sets of pairs $H$ of elements of $R$ such that the class $C(H, R)$ has no solution at all and therefore no optimal solution. This occurs for example for every set of pairs $H$ containing a pair $(a, b)$ such that $a$ and $b$ are not adjacent in $G$.

Thus, to complete the description of the algorithm we need only to show how the optimal solutions are updated during the reduction process.

Suppose, that the algorithm has reached a point, after several reductions of the reduction sequence, such that the $k$–tree $T$ is reduced to $T$, and for every $k$–clique $K$ of $T$ and every set of pairs $H$ of elements of $K$ an optimal solution in the class $C(H, K)$ has already been computed. Suppose that the next step of
the algorithm is to remove vertex \( v \) which is a \( k \)-leaf of \( \hat{T} \) and is adjacent to the \( k \)-dique of \( \hat{T} \) denoted by \( K(v) \). All the \( k \)-diques in which \( v \) participates are of the form \( K(v) - \{u\} \cup \{v\} \) for some \( u \in K(v) \). Removing \( v \) will remove all these \( k \)-diques from the \( k \)-tree \( \hat{T} \), and will cause \( v \) to become a descendent of \( K(v) \), and all the descendants of these \( k \)-diques to become descendants of \( K(v) \).

We will denote by \( \hat{T} \) the \( k \)-tree resulted after removing \( v \) from \( \hat{T} \), and we will denote by \( \overline{B(K(v))} \) the set of all the descendants of \( K(v) \) in \( \hat{T} \). From the previous paragraph:

\[
\overline{B(K(v))} = B(K(v)) \cup \{v\} \bigcup_{u \in K(v)} \overline{B(K(v)) - \{u\} \cup \{v\}} \tag{1}
\]

Note that \( B(K(v)) \) and \( \overline{B(K(v)) - \{u\} \cup \{v\}} \) in the above formula denote the descendants of \( K(v) \) and \( K(v) - \{u\} \cup \{v\} \) in the \( k \)-tree \( \hat{T} \) respectively.

For each set of pairs \( H \) of elements of \( K(v) \) we define a class of solutions to the problem on the subgraph of \( G \) induced by \( K(v) \cup \overline{B(K(v))} \). This class denoted by \( \tilde{C}(H, K(v)) \) consists of all the sets of pairs \( H' \) of elements from \( K(v) \cup \overline{B(K(v))} \) which satisfies the following two conditions:

(i) The intersection of \( H' \) with \( K(v) \) equals \( H \).

(ii) \( H' \) defines a 2-spanner for the subgraph of \( G \) induced by \( K(v) \cup \overline{B(K(v))} \).

For each set of pairs \( H \) of elements of \( K(v) \), our purpose is to find an optimal solution in the class \( \tilde{C}(H, K(v)) \). To do this we further partition this class to subclasses of the form \( \tilde{C}(H', H, K(v)) \) as defined below.

Let \( H' \) be a (possibly empty) set of pairs of elements of \( K(v) \) such that \( v \) is in every pair, i.e. \( H' \) is a subset of \( \{(v, w) : w \in K(v)\} \), we define a subclass of class \( \tilde{C}(H, K(v)) \). This subclass denoted by \( \tilde{C}(H', H, K(v)) \) consists of all the solutions \( H'' \) in the class (of solutions) \( \tilde{C}(H, K(v)) \) such that the intersection of \( H'' \) and \( K(v) \cup \{v\} \) equals \( H \cup H' \).

If for every set of pairs \( H' \) subset of \( \{(v, w) : w \in K(v)\} \) we could find an optimal solution in the subclass \( \tilde{C}(H', H, K(v)) \), then the optimal solution in the class \( \tilde{C}(H, K(v)) \) will be the minimum of all these solutions. Therefore we complete the description of the algorithm by lemma 3.1 below, which shows how an optimal solution in the class \( \tilde{C}(H', H, K(v)) \) can be found.

**Lemma 3.1** Let \( H \) be any set of pairs of elements of \( K(v) \), let \( H' \) be any set of pairs subset of \( \{(v, w) : w \in K(v)\} \), let \( H_1 \) be an optimal solution to the problem in the class \( \tilde{C}(H, K(v)) \), and for every \( u \in K(v) \) let \( H_u \) be an optimal solution to the problem in the class \( \tilde{C}(H \cup H' - \{u\}, K(v) - \{u\} \cup \{v\}) \), where by \( H \cup H' - \{u\} \) we denote all the pairs of \( H \cup H' \) excluding the pairs containing \( u \). Then \( H_1 \cup \bigcup_{u \in K(v)} H_u \) is an optimal solution in the class \( \tilde{C}(H', H, K(v)) \).

(Note that by the assumptions about the state of the algorithm before removing \( v \), the optimal solutions \( H_1 \) and \( \{H_u : u \in K(v)\} \) are all known.)
**Proof:** Claim 1: \( H_1 \in K(v) \cup \overline{B(K(v))} \), and for every \( u \in K(v) \), \( H_u \in K(v) \cup B(K(v)) \). **Proof:** This follows immediately from the definition of \( B(K(v)) \).

Claim 2: The intersection of \( H_1 \) with \( K(v) \cup \{v\} \) equals \( H \). **Proof:** This follows from the fact that \( H_1 \) is in the class \( C(H, K(v)) \).

Claim 3: For every \( u \in K(v) \), the intersection of \( H_u \) with \( K(v) \cup \{v\} \) equals \( H \cup H' - \{u\} \). **Proof:** From the definition of \( H_u \), the intersection of \( H_u \) with \( K(v) - \{u\} \cup \{v\} \) equals \( H \cup H' - \{u\} \), and \( u \) is not contained in any pair of \( H_u \). This implies Claim 3.

Claim 4: The intersection of \( H_1 \cup \bigcup_{u \in K(v}) H_u \) with \( K(v) \cup \{v\} \) equals \( H \cup H' \). **Proof:** This follows from Claims 2, 3 above.

Claim 5: \( H_1 \cup \bigcup_{u \in K(v}} H_u \) is a 2-spanner for the subgraph of \( G \) induced by \( K(v) \cup \overline{B(K(v))} \). (Note that we interpret the pairs of \( H_1 \cup \bigcup_{u \in K(v}} H_u \) as edges of the graph \( G \).) **Proof:** Let \( x, y \) be two vertices of \( K(v) \cup \overline{B(K(v))} \).

Case (i): Suppose \( x \in K(v) \) and \( y \in K(v) \). Then since \( H_1 \) is a 2-spanner for the subgraph of \( G \) induced by \( K(v) \cup \overline{B(K(v))} \), it follows that either \( (x, y) \in H_1 \) or there exist \( z \) such that \( (x, z) \in H_1 \) and \( (y, z) \in H_1 \).

Case (ii): Suppose \( x \in \overline{B(K(v))} \) and \( y \in K(v) \). As before \( x \) and \( y \) can be either \( v \), members of \( B(K(v)) \), or members of \( B(K(v) - \{u\} \cup \{v\}) \) for some \( u \in K(v) \). We will assume that \( y \) is a member of \( B(K(v) - \{u\} \cup \{v\}) \) for some \( u \in K(v) \), and that \( y \neq x \). (The other cases can be handled similarly). Then since \( H_u \) is a 2-spanner for the subgraph of \( G \) induced by \( K(v) - \{u\} \cup \{v\} \cup \overline{B(K(v))} \), it follows that either \( (x, y) \in H_u \) or there exist \( z \) such that \( (x, z) \in H_u \) and \( (y, z) \in H_u \).

Case (iii): Suppose \( x \in B(K(v)) \) and \( y \in B(K(v)) \). We will assume that \( x \) is a member of \( B(K(v) - \{u_1\} \cup \{v\}) \) for some \( u_1 \in K(v) \), and \( y \) is a member of \( B(K(v) - \{u_2\} \cup \{v\}) \) for some \( u_2 \in K(v) \), and \( u_1 \neq u_2 \). (The other cases can be handled similarly). Then from lemma 2.2 it follows that \( x \) and \( y \) are not adjacent in \( G \). Thus for showing that \( H_1 \cup \bigcup_{u \in K(v}} H_u \) is a 2-spanner there is no need to consider the pair \( (x, y) \).

**Completion of proof of lemma 3.1:** From claims 4, 5 above it follows that \( H_1 \cup \bigcup_{u \in K(v}} H_u \) is a solution in the class \( C(H', H, K(v)) \). Suppose that this solution is not optimal. Then there exist another solution \( \hat{H} \) with less edges (i.e. less pairs) than \( H \). Let \( H_u \) denote the intersection of \( \hat{H} \) with \( K(v) - \{u\} \cup \{v\} \cup B(K(v) - \{u\} \cup \{v\}) \). Then from the definition of \( \hat{H} \) and lemma 2.2 \( H_u \) must be a solution in the class \( C(H \cup H' - \{u\}, K(v) - \{u\} \cup \{v\}) \). Also since \( \hat{H} \) has less edges than \( H_1 \cup \bigcup_{u \in K(v}} H_u \), it follows that either the intersection of \( \hat{H} \) with \( K(v) \cup B(K(v)) \) has less edges than \( H_1 \), or there exist \( u \in K(v) \) such that \( H_u \) has less edges than \( H_u \). But this contradicts the optimality of \( H_1 \) and \( \{H_u : u \in K(v)\} \) in the classes \( C(H, K(v)) \) and \( C(H \cup H' - \{u\}, K(v) - \{u\} \cup \{v\}) \) respectively. \( \square \)
3.2 Algorithm’s Complexity

The computations done in each step of the algorithm are functions of $k$ only (although exponential function in $k$). These computations are considered as constants since it is assumed that $k$ is fixed. The number of steps done in the algorithm is $n - k$, where $n$ denoted the number of vertices in $G$. Therefore the complexity of the algorithm is linear in $n$. However if we take as an input to the algorithm a partial $k$-tree $G = (V, E)$, and apply as a first step the algorithm of [ACP87] to get the $k$-tree $T = (V, E')$ embedding of $G$, $E \subseteq E'$, the complexity of the algorithm will be the complexity of this step which according to [ACP87] is $O(n^{k+2})$.

3.3 Extending the result to the $t$-spanner problem

The $t$-spanner problem is defined as follows: Given a graph $G = (V, E)$ find an optimal $t$-spanner for $G$, i.e. find a subgraph $\tilde{G} = (V, \tilde{E})$ of $G$, $\tilde{E} \subseteq E$, such that the distance between any pair of vertices in $\tilde{G}$ is at most $t$-times their distance in $G$, and $\tilde{G}$ has a minimum number of edges.

We assume that $G = (V, E)$ is a partial $k$-tree embedded in a $k$-tree $T = (V, E')$ with $E \subseteq E'$.

The algorithm described in the previous section can be extended to solve the $t$-spanner problem by replacing the sets of pairs $H$ of vertices of $G$, by sets of $t$-tuples of vertices of $G$. In the algorithm described above a pair $(x, y)$ in $H$ corresponds to an edge in $G$ between $x$ and $y$. In the extension to this algorithm a $t$-tuple $X_1, \ldots, X_t$ corresponds to the path between $X_1$ and $X_t$ in $G$ which goes through vertices $X_2, \ldots, X_{t-1}$. The rest of the details of the extended algorithm goes along the same lines of the algorithm described above and will not be presented here. We conclude this section by the following theorem:

THEOREM A. For fixed $k$ and $t$, the problem of finding an optimal $t$-spanner for partial $k$-trees can be solved in polynomial time.

4 Solving the 2-spanner problem on a clique sequence graph

A Clique $C$ of a graph $G$ denote a maximal completely connected subgraph of $G$. (This is in accordance with standard graph terminology). In the following when we say that two cliques $C_1$ and $C_2$ intersects we mean that their sets of vertices intersects.

A graph $G = (V, E)$ is a clique sequence graph if its set of cliques can be ordered in a sequence $C_1, \ldots, C_m$ such that for $1 < i < m$, $C_i$ intersects just $C_{i+1}$ and $C_{i-1}$, i.e. for $1 < i < m$, $C_i \cap C_{i-1} \neq \emptyset$ and $C_i \cap C_{i+1} \neq \emptyset$, and for every $1 < j < m$, such that $j \neq i - 1$ and $j \neq i + 1$ $C_i \cap C_j = \emptyset$.
Example 4.1 Let $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{(1, 2), (2, 3), (1, 3), (3, 4), (3, 5), (4, 5), (5, 6)\}$. Then $G$ is a clique sequence graph, since its maximal cliques are:

- $C_1 = \{V_1, E_1\}$, where $V_1 = \{1, 2, 3\}$ and $E_1 = \{(1, 2), (2, 3), (1, 3)\}$,
- $C_2 = \{V_2, E_2\}$, where $V_2 = \{3, 4, 5\}$ and $E_2 = \{(3, 4), (3, 5), (4, 5)\}$,
- $C_3 = \{V_3, E_3\}$, where $V_3 = \{5, 6\}$ and $E_3 = \{(5, 6)\}$,

and $C_1 \cap C_3 = \emptyset$, $C_1 \cap C_2 = \{3\}$ and $C_2 \cap C_3 = \{5\}$.

THEOREM B. The problem of finding an optimal 2-spanner for clique sequence graphs can be solved in linear time.

Proof: Let $G = (V, E)$ be a clique sequence graph, and let its clique sequence be $C_1, \ldots, C_m$. We assume that $m > 2$, since if $m \leq 2$ the result of the above theorem is trivial. Let $C_j, C_{j+1}$ be the two cliques such that their intersection has maximum size of all the intersections $C_i \cap C_{i+1}$ for $1 \leq i \leq m-1$. Construct the set of edges $E'$ of the subgraph $G' = (V, E')$ of $G$ as follows:

(i) Start with $E'$ empty.

(ii) For $1 \leq i \leq m-1$, such that $i \neq j$ choose any vertex $u$ from $C_i \cap C_{i+1}$, and add to $E'$ the edges that connects $u$ to all vertices of $C_i \cup C_{i+1}$, i.e.

\[ E' = E' \cup \{(u, v) : v \in C_i \cup C_{i+1} \} \]

(iii) If $j \neq 1$ and $j \neq m-1$ stop.

(iv) If $j = 1$, choose any vertex $x$ of $C_1$, and add to $E'$ the edges that connects $x$ to all vertices of $C_1$.

(v) If $j = m-1$, choose any vertex $x$ of $C_m$, and add to $E'$ the edges that connects $x$ to all vertices of $C_m$.

It is easy to see that $G'$ is a 2-spanner for $G$. For let $x, y$ be two vertices adjacent in $G$. Then either both $x$ and $y$ are in clique $C_i$, for some $i$, $1 \leq i \leq m$, or $x$ and $y$ are in adjacent cliques $C_i, C_{i+1}$ for some $i$, $1 \leq i \leq m-1$. Suppose that $x$ and $y$ are in adjacent cliques $C_i, C_{i+1}$ (the other cases can be handled similarly), and let $u$ be the vertex of $C_i \cap C_{i+1}$ chosen in step (ii) of the above construction. Then the edges $(u, x)$ and $(u, y)$ of $E'$ defines a path of length two between $x$ and $y$ in $G'$.

Also it is easy to see that the complexity of the construction of $G'$ is linear in $n$, (where $n$ is the number of vertices in $G$). For the complexity of the construction is given by the number of edges in $E'$. The number of edges of $E'$ can be counted as $m-1$ edges of the form $(u, v)$ such that both $u$ and $v$ are chosen at step (ii) above, and additional edges of the form $(u, x)$ where $u$ is chosen at step (ii) above and $x$ is not chosen at step (ii) above. These additional
edges are bounded by \(2n\), since the degree of each vertex \(x\) not chosen at step (ii) above is at most 2. Thus the number of edges in \(E'\) is bounded by \(2n + m - 1\) which is bounded by \(3n\).

However verifying that \(G'\) is an optimal 2-spanner is difficult and will not be presented here.

\[\Box\]

**Example 4.2** Let \(G = (V, E)\) be the sequence clique graph of example 4.1. One of the possible optimal 2-spanners constructed by the algorithm for graph \(G\) is the graph \(G' = (V, E')\) where \(E' = \{(5, 6), (5, 4), (5, 3), (1, 2), (1, 3)\}\).

## 5 Conclusions and further research

Assume you have a problem \(K\) which is \(\text{NP}\)-hard in general, but in \(\text{P}\) for a certain class of graphs \(G\). Let \(\text{Patch}_G\) be a class of graphs which are ‘patched’ together from graphs from \(G\). Is membership in \(K \cap \text{Patch}_G\) now in \(\text{P}\)?

The clique sequence graphs are a simple example of such a patching, and in subsequent papers we study how far such a construction can be generalized such that Theorem B remains true. The \(k\)-trees (not partial) can be viewed as a more complicated patching of \(k + 1\) cliques. The partial \(k\)-trees are the obtained as subsets of edges of a patching. Again, the question arises on how to generalize Theorem A. A good candidate for general notion of patchings are those simplicial decomposition of graphs (cf. [Die90]), which fit the framework of generalized sums studied in [Gur79].

For \(K\) we restricted our investigation here to finding optimal \(t\)-spanners. In [AP89] theorem A was also proved for \(K\) being maximum independent set size, minimum dominating set size, chromatic number, hamiltonian circuit and network reliability. A further problem is to find structural properties of \(K\), which allow generalizations of Theorem A. Those could be found in the structure of the second order logic sentence describing \(K\). In this paper we have shown, that the question, as posed above, may have interesting non-trivial answers.

## References


