A Polynomial Approximation Algorithm for the Minimum Fill-In Problem

Assaf Natanzon [*] Ron Shamir [*] [†] Roded Sharan [*]

Abstract

In the *minimum fill-in* problem, one wishes to find a set of edges of smallest size, whose addition to a given graph will make it chordal. The problem has important applications in numerical algebra and has been studied intensively since the 1970s. We give the first polynomial approximation algorithm for the problem. Our algorithm constructs a triangulation whose size is at most eight times the optimum size squared. The algorithm builds on the recent parameterized algorithm of Kaplan, Shamir and Tarjan for the same problem.

For bounded degree graphs we give a polynomial approximation algorithm with a polylogarithmic approximation ratio. We also improve the parameterized algorithm.

1 Introduction

A *chord* in a cycle is an edge between non-consecutive vertices on that cycle. A *chordless cycle* is a cycle of length greater than 3 that contains no chords. A graph is called *chordal* or *triangulated*, if it contains no chordless cycles. If $G = (V, E)$ is not chordal and F is a set of edges such that $(V, E \cup F)$ is chordal, then F is called a *fill-in* or a *triangulation* of G. If $|F| \leq k$ then F is called a k*-triangulation* of G. We use $\Phi(G)$ to denote the size of the smallest fill-in of G.

The *minimum fill-in* problem is to find a minimum triangulation (fill-in) of a given graph. The importance of the problem stems from its applications to numerical algebra. In many fields, including VLSI simulation, solution of linear programs, signal processing and others (cf. [7]), one has to perform a Gaussian elimination on a sparse symmetric positive-definite matrix. During the elimination process zero entries may become non-zeroes. Different elimination orders may introduce different sets of new non-zero elements into the matrix. The time of the computation and its storage needs are dependent on the sparseness of the matrix. It is therefore desirable to find an elimination order such that a minimum number of zero entries is filled-in with non-zeroes (even temporarily). Rose [20] proved that the problem of finding an elimination order for a symmetric positive-definite matrix M, such that fewest new non-zero elements are introduced, is equivalent to the minimum fill-in problem on a graph whose vertices correspond to the rows of M and in which (i, j) is an edge iff $M_{i,j} \neq 0$.

In 1979 Garey and Johnson [9] posed the complexity of the minimum fill-in problem as a major open problem. Yannakakis subsequently proved that the minimum fill-in problem is NP-complete [22]. Due to its importance the problem has been studied intensively [2, 11, 12, 21] and many heuristics have been developed for it [5, 13, 19, 20]. None of those gives a performance guarantee with respect to the size of the fill-in introduced. Note that in contrast, the *minimal* fill-in problem (finding a a triangulation of G which is minimal with respect to inclusion) is polynomial [18].

Approximation attempts succeeded only for the related *minimum triangulated super-graph* problem (MTS). In MTS the goal is to add edges to the input graph in order to obtain a chordal graph with minimum *total* number of edges. While as optimization problems MTS and minimum fill-in are equivalent, they differ drastically as approximation problems. For example, if we remove two properly chosen edges from an n-clique we obtain a graph with $\Omega(n^2)$ edges whose fill-in size equals 1. The approximation results regarding MTS use the nested dissection heuristic first proposed by George [10] (see [12] for details). Gilbert [14] showed that for a graph with maximum degree d there exists a balanced separator decomposition such that a nested dissection ordering based on that decomposition yields a chordal supergraph, in which the number of edges is within a factor of $O(d \log n)$ of optimal. (Throughout we use n and m to denote the number of vertices and edges, respectively, in a graph). The result was not constructive as one has yet to

[*]Department of Computer Science, Tel Aviv University, Tel Aviv, Israel. {natanzon,shamir,roded}@math.tau.ac.il.

[†]Current address: Department of Computer Science and Engineering, University of Washington, Seattle, Washington, USA.

find such a decomposition. Leighton and Rao [17] gave a polynomial approximation algorithm for finding a balanced separator in a graph of size within a factor of $O(\log n)$ of optimal. Agrawal, Klein and Ravi [1], using Gilbert's ideas and the result of [17], obtained a polynomial approximation algorithm with ratio $O(\sqrt{d}\log^4 n)$ for MTS on graphs with maximum degree d. They also gave a polynomial approximation algorithm for MTS on general graphs, which generates a chordal super-graph with total number of edges $O(|G^*|^{3/4}\sqrt{m}\log^{3.5} n)$, where $|G^*|$ denotes the size of an optimally triangulated super-graph.

In the *parametric* fill-in problem the input is a graph G and a parameter k. The goal is to find a k-triangulation of G, or to determine that none exists. Clearly this can be done in $n^{O(k)}$ time by enumeration. For fixed k and growing n, an algorithm with complexity $O(exp(k)poly(n))$ is superior. Parameterized complexity theory, initiated by Downey and Fellows (cf. [6]), studies the complexity of such problems. Parameterized problems that have algorithms of complexity $O(f(k)n^\alpha)$ (with α a constant) are called *fixed parameter tractable*. Kaplan, Shamir and Tarjan [16] (henceforth KST) and independently Cai [3] proved that the minimum fill-in problem is fixed parameter tractable, by giving an algorithm of complexity $O(exp(k)m)$ for the problem. KST also gave a more efficient $O(exp(k) + k^2nm)$-time algorithm (henceforth KST algorithm).

In this paper we give the first polynomial approximation algorithm for the minimum fill-in problem. Our algorithm builds on ideas from [16]. For an input graph G with minimum fill-in of size k, our algorithm produces a triangulation of size at most $8k^2$ - within a factor of $8k$ of optimal. The approximation is achieved by identifying in G a *kernel* set of vertices A of size at most $4k$, such that one can triangulate G by adding edges only between vertices of A. Our algorithm produces the triangulation without prior knowledge of k. Let $M(n)$ denote the number of operations needed to multiply two boolean matrices of order $n \times n$ (The current upper bound on $M(n)$ is $O(n^{2.376})$ [4]). The algorithm works in $O(knm + \min\{n^2M(k)/k, nM(n)\})$ time, which makes it potentially suitable for practical use.

Our algorithm is particularly attractive for small fill-in values. Note that if $k = \Omega(n)$ then our algorithm guarantees only the trivial bound of fill-in size - $O(n^2)$, but if for example the fill-in size is constant then the approximation guarantee is a constant. This type of approximation result is uncommon. It opens the question of obtaining polynomial approximation algorithms with performance guarantees depending on the optimal value for other important problems, even in the presence of hardness-of-approximation results.

We also obtain better approximation results for bounded degree graphs. For graphs with maximum degree d we give a polynomial algorithm which achieves an approximation ratio of $O(d^{2.5}\log^4(kd))$. Since $k = O(n^2)$ this approximation ratio is polylogarithmic in the input size.

In order to compare our results to the approximation results regarding MTS, we translate the latter to approximation ratios in terms of the fill-in obtained. We assume throughout that $m > n$. For general graphs the algorithm in [1] guarantees that the number of edges in the chordal super-graph obtained is $O((m+k)^{3/4}\sqrt{m}\log^{3.5} n)$. In terms of the fill-in, the approximation ratio achieved is $O(m^{1.25}\log^{3.5} n/k + \sqrt{m}\log^{3.5} n/k^{1/4})$. We obtain a better approximation ratio whenever $k = O(m^{5/8}\log^{1.75} n)$. For graphs with maximum degree d, the algorithm in [1] achieves an approximation ratio of $O(((nd+k)\sqrt{d}\log^4 n)/k)$. We provide a better ratio when $k = O(n/d)$. When any of these upper bounds on k is satisfied, our algorithm also achieves a better approximation ratio than [1] for the MTS problem.

Kaplan et al. posed in [16] an open problem of obtaining an algorithm for the parametric fill-in problem with time $O(exp(k) + km)$. The motivation is to match the performance of the $O(exp(k)m)$ algorithm for all k. We make some progress towards solving that problem by providing a faster $O(exp(k) + knm + \min\{n^2M(k)/k, nM(n)\})$-time implementation of their algorithm. We also give a variant of the algorithm which produces a smaller kernel. Finally, we apply our approximation algorithm to the *chain completion* problem, and obtain an approximation ratio of $8k$, where k denotes the size of an optimal solution.

The paper is organized as follows: Section 2 contains a description of KST algorithm and some background. Section 3 improves the complexity of KST algorithm and reduces the size of the kernel produced. Section 4 describes our approximation algorithm for general graphs. Section 5 gives an approximation algorithm for graphs with bounded degree. Section 6 gives further reduction of the kernel size, and Section 7 gives an approximation algorithm for the chain completion problem.

2 Preliminaries

Let $G = (V, E)$ be a graph. We denote its set of vertices also by $V(G)$ and its set of edges also by $E(G)$. For $U \subseteq V$ we denote by G_U the subgraph induced by the vertices in U. For a vertex $v \in V$ we denote by $N(v)$ the set containing all neighbors of v in G. We let $N[v] = N(v) \cup \{v\}$. A path with l edges is called an *l-path* and its *length* is l. A single vertex is considered a 0-path. We call a cycle with l edges an *l-cycle*.

Our polynomial approximation algorithm for the minimum fill-in problem builds on KST algorithm [16]. In the following we describe this algorithm.

Fact 2.1 *A minimal triangulation of a chordless l-cycle consists of $l-3$ edges.*

Lemma 2.2 *[16, Lemma 2.5] Let C be a chordless cycle and let p be an l-path on C, $1 \le l \le |C|-2$. If $l = |C|-2$ then in every minimal triangulation of C there are at least $l-1$ chords incident with vertices of p. If $l < |C|-2$ then in every minimal triangulation of C there are at least l chords incident with vertices of p.*

Let $< G = (V,E), k >$ be the input to the parametric fill-in problem. The algorithm has two main stages. In the first stage, which is polynomial in n, m, k, the algorithm produces a partition A, B of V and a set F of non-edges in G_A, such that $|A| = O(k^3)$ and no chordless cycle in $G' = (V, E \cup F)$ intersects B. We shall call this stage *the partition algorithm*. In the second stage, which is exponential in k, an exhaustive search is applied to find a minimum triangulation F' of G'_A. $(F \cup F')$ is then proved to be a minimum triangulation of G.

The partition algorithm applies sequentially the following three procedures. All three maintain a partition A, B of V and a lower bound cc on the minimum number of edges needed to triangulate G. Initially $A = \emptyset, B = V$ and $cc = 0$.

- Procedure $P_1(k)$: *Extracting independent chordless cycles*. Search repeatedly for chordless cycles in G_B and move their vertices from B to A. For each chordless l-cycle found, increment cc by $l - 3$. If at any time $cc > k$, stop and declare that the graph admits no k-triangulation.

- Procedure $P_2(k)$: *Extracting related chordless cycles with independent paths*. Search repeatedly for chordless cycles in G containing at least two consecutive vertices from B. Let C be such a cycle, $|C| = l$. If $l > k + 3$ stop with a negative answer. Otherwise, suppose that C contains $j \geq 1$ disjoint maximal sub-paths in G_B, each of length at least 1. Move the vertices of those sub-paths from B to A. Denote their lengths in decreasing order by $l_1, \ldots, l_j$. If $j = 1$ we increase cc either by $(l_1 - 1)$ if $l_1 = l - 2$, or by l_1 if $l_1 < l - 2$. Otherwise cc is increased by $\max\{\frac{1}{2}\sum_{i=1}^{j} l_i, l_1\}$. If at any time $cc > k$, stop and declare that the graph admits no k-triangulation.

Definition 2.3 *For every $x, y \in A$ such that $(x, y) \notin E$, denote by $A_{x,y}$ the set of all vertices $b \in B$ such that x, b, y occur consecutively on some chordless cycle in G. If $|A_{x,y}| > 2k$ then (x, y) is called a k-*essential edge.

- Procedure $P_3(k)$: *Adding k-essential edges in G_A*. For every $x, y \in A$ such that $(x, y) \notin E$ compute the set $A_{x,y}$. If (x, y) is a k-essential edge, then add it to G. Otherwise, move all vertices in $A_{x,y}$ from B to A.

Denote by A^i, B^i the partition obtained after procedure P_i is completed, for $i = 1, 2, 3$. We will omit the index i when it is clear from the context. The size of A^2 is at most $4k$, since in a triangulation of G there should be an edge incident on at least one of any two vertices occurring consecutively on a chordless cycle. The size of A^3 is $O(k^3)$ since there are $O(k^2)$ non-edges in G_{A^2} and the number of vertices moved to A due to any such non-edge is at most $2k$.

```
KST Partition Algorithm
    Execute procedure $P_1(k)$.
    Execute procedure $P_2(k)$.
    Execute procedure $P_3(k)$.
```

The partition algorithm is summarized above. Let G' denote the graph obtained after the execution of procedure P_3. KST prove that every k-essential edge must appear in any k-triangulation of G [16, Lemma 2.7], and that in G' no chordless cycle intersects B [16, Theorem 2.10]. Therefore, by the following theorem it suffices to search for a minimum triangulation of G'_A.

Theorem 2.4 *[16, Theorem 2.13] Let A, B be a partition of the vertex set of a graph G, such that the vertices of every chordless cycle in G are contained in A. A set of edges F is a minimal triangulation of G iff F is a minimal triangulation of G_A.*

The complexity of the partition algorithm is $O(k^2nm)$ [16]. The complexity of finding a minimum triangulation of a given graph is $O(\frac{4^k}{(k+1)^{3/2}}m)$ [3]. Since G'_A contains $O(k^6)$ edges, a minimum triangulation of G'_A can be found in $O(k^{4.5}4^k)$ time. Hence, the complexity of KST algorithm is $O(k^2nm + k^{4.5}4^k)$.

3 Improvements to the Partition Algorithm

In this section we show some improvements to the partition algorithm of KST. We assume throughout that the input is $< G = (V,E), k >$. We first show how to implement procedure P_3 in $O(nm + \min\{n^2M(k)/k, nM(n)\})$-time. We then prove that the size of A^3 is only $O(k^2)$. These results imply that KST algorithm can be implemented in $O(knm + \min\{n^2M(k)/k, nM(n)\} + k^{2.5}4^k)$-time.

Lemma 3.1 *Procedure P_3 can be implemented in $O(nm + \min\{n^2M(k)/k, nM(n)\})$-time.*

Proof: Let $S = \{(x, y) \notin E : x, y \in A^2\}$. The bottleneck in the complexity of P_3 is computing the sets $A_{x,y}$ for every $(x, y) \in S$. To do that, we find for every $b \in B$ all pairs $(x, y) \in S$ such that $b \in A_{x,y}$. We then construct the sets $A_{x,y}$. This is done as follows:

Fix $b \in B$. Compute the connected components of $G^b = G - N[b]$. This takes $O(m)$ time. Denote the connected components of G^b by $C_1^b, \ldots, C_l^b$. For each $x \in A \cap N(b)$ compute a binary vector $(v_1^x, \ldots, v_l^x)$ such that $v_j^x = 1$ iff C_j^b contains a neighbor of x, $1 \leq j \leq l$. Each vector can be computed in $O(n)$ time. Denote the vectors obtained by $\vec{v_1}, \ldots, \vec{v_{k'}}$. Define a $k' \times l$ boolean matrix M whose i-th row is the vector $\vec{v_i}$, $1 \leq i \leq k'$. Note that $k' = O(k)$ and $l \leq n$. Let $M^* = MM^T$. It can be seen that $M^*_{i,j} \geq 1$ iff $b \in A_{i,j}$ (for $i \neq j$). Since $k', l \leq n$ we can compute M^* in $O(M(n))$ time. If $k = o(n)$ then we can compute M^* in $O(nM(k)/k)$ time. Hence, the computation of M^* takes $O(\min\{nM(k)/k, M(n)\})$ time.

After the above calculations are performed for every $b \in B$, it remains to compute the sets $A_{x,y}$. We can do that in $O(\min\{k^2n, n^3\})$ time. The total time is therefore $O(nm + \min\{n^2M(k)/k, nM(n)\})$. ■

Observation 3.2 *Let $x, y \in A^2, (x,y) \notin E$. If $A_{x,y} \neq \emptyset$ then for any triangulation F of G, either $(x,y) \in F$, or for every $b \in A_{x,y}$, F contains an edge incident on b.*

Lemma 3.3 *Assume that G admits a k-triangulation, and that in procedure P_3 all sets $A_{x,y}$ moved into A are of size at most d. Then $|A^3 - A^2| \leq Mk$, where $M = \max\{d, 2\}$.*

Proof: Let the non-edges in G_{A^2} be $(x_1, y_1), \ldots, (x_l, y_l)$. We process the sets $A_{x_1,y_1}, \ldots, A_{x_l,y_l}$ in this order. Let $A^{(0)} = A^2$. Let $A^{(i)}$ be the set A right after A_{x_i,y_i} was processed, and let $\Delta_i = A_{x_i,y_i} - A^{(i-1)}$, for $1 \leq i \leq l$.

Let t be a lower bound on the minimum number of edges needed to triangulate G. Initially P_3 starts with $t = cc$. Let t_i be the value of t right after A_{x_i,y_i} was processed ($t_0 = cc$). If $\Delta_i \neq \emptyset$ then by Observation 3.2, t should increase by $\min\{1, |\Delta_i|/2\}$. We must maintain $t \leq k$. If $t_i - t_{i-1} = 0$ then $|\Delta_i| = 0$. If $t_i - t_{i-1} = 1/2$ then $|\Delta_i| = 1$. If $t_i - t_{i-1} \geq 1$ then $|\Delta_i| \leq d$. Therefore for all $1 \leq i \leq l$, $|\Delta_i| \leq M(t_i - t_{i-1})$. Now,

$$|A^3 - A^2| = |A^{(l)} - A^{(0)}| = \sum_{i=1}^{l} |A^{(i)} - A^{(i-1)}| =$$

$$\sum_{i=1}^{l} |\Delta_i| \leq M \sum_{i=1}^{l} (t_i - t_{i-1}) = M(t - t_0) \leq Mk.$$

■

Corollary 3.4 *If G has a k-triangulation, then the partition algorithm terminates with $|A| \leq 2k(k+2)$.*

Proof: Let us assume that all essential edges were added to G, and denote the new set of edges in G by E'. For all $x, y \in A^2, (x,y) \notin E'$ we know that $|A_{x,y}| \leq 2k$. By Lemma 3.3 $|A^3 - A^2| \leq 2k^2$. Since $|A^2| \leq 4k$ the corollary follows. ■

Theorem 3.5 *The algorithm of KST can be implemented in $O(knm + \min\{n^2 M(k)/k, nM(n)\} + k^{2.5}4^k)$-time.*

Proof: By the analysis in [16], P_1 takes $O(km)$ time, and P_2 takes $O(knm)$ time. By Lemma 3.1 the complexity of P_3 is $O(nm + \min\{n^2 M(k)/k, nM(n)\})$. By Corollary 3.4 if G admits a k-triangulation then the size of A^3 is $O(k^2)$. Hence, a minimum triangulation of G'_A can be found in $O(k^{2.5}4^k)$ time. The complexity follows. ■

4 The Approximation Algorithm

Let $G = (V, E)$ be the input graph. Let $k_{opt} = \Phi(G)$. The key idea in our approximation algorithm is to find a set of vertices $A \subseteq V$, such that $|A| = O(k_{opt})$ and, moreover, one can triangulate G by adding edges only between vertices of A. Since there are $O(k_{opt}^2)$ such edges, we achieve an approximation ratio of $O(k_{opt})$.

In order to find such a set A we use ideas from the partition algorithm. If we knew k_{opt} we could execute the partition algorithm and obtain a set A, with $|A| = O(k_{opt}^2)$ (by Corollary 3.4), such that G can be triangulated by adding edges only in G_A. This would already give an $O(k_{opt}^3)$ approximation ratio.

Before describing our algorithm we analyze the role of the parameter k given to the partition algorithm. If $k < k_{opt}$ then the algorithm might stop during P_1 or P_2 and declare that no k-triangulation exists. Moreover, k-essential edges are not necessarily k_{opt}-essential. If $k > k_{opt}$ then the size of A may be $\omega(k_{opt}^2)$. The algorithm is as follows:

```
Algorithm APPROX
    Procedure $P'_1$: Execute $P_1(\infty)$.
    Procedure $P'_2$: Execute $P_2(\infty)$.
    Procedure $P'_3$: Execute $P_3(0)$.
    Let $G'$ be the resulting graph.
    Procedure $P'_4$: Find a minimal triangulation of $G'_A$.
```

Procedures P'_1 and P'_2 execute P_1 and P_2 respectively, without bounding the size of the triangulation implied. Procedure P'_3 takes advantage of the fact that we no longer seek a minimum triangulation, but rather a minimal one. In order to obtain our approximation result we want to keep A as small as possible. Hence, instead of moving new vertices to A we add new 0-essential edges accommodating for those vertices. By the same arguments as in [16] and Section 2, the size of A after P'_2 is at most $4k_{opt}$. Since P'_3 does not add new vertices to A, its size remains at most $4k_{opt}$ throughout. The size of the triangulation found by the algorithm is therefore at most $8k_{opt}^2$. The correctness of algorithm APPROX is established in the sequel. We need the following lemma which is essentially proven in [16, Lemma 2.9]. The subsequent theorem is along the lines of [16, Theorem 2.10].

Lemma 4.1 *Let $G = (V, E)$ be a graph and let $v \in V$. Let F be a set of non-edges in $G - \{v\}$, such that each $e = (x, y) \in F$ is a chord in a chordless cycle $C_e = (x, z_e, y, \ldots, x)$ in G, where z_e is not an endpoint of any edge in F. Let $G' = (V, E \cup F)$. If there exists a chordless cycle C in G' with v_1, v, v_2 occurring consecutively on C, then either there exists a chordless cycle in G on which v_1, v, v_2 occur consecutively, or there exists a chordless cycle in G, on which v and z_e occur consecutively, for some $e \in F$.*

Theorem 4.2 *No chordless cycle in G' intersects B.*

Proof: Suppose to the contrary that C is a chordless cycle in G' with at least one vertex from B. Let $v \in C \cap B$. Denote by v_1 and v_2 the two neighbors of v on C. Let $F = E(G') - E(G)$. C must contain at least one edge $e = (x, y) \in F$, since otherwise C exists in G and either v would have been moved to A by P_1' or P_2', or (v_1, v_2) would have been added to G' by P_3'. By construction e is a chord in a chordless cycle C_e in G, such that if P_e^1 and P_e^2 are the two paths connecting x and y in C_e, with x and y removed from each path, then at least one of them consists of a single vertex $z_e \in B$. As $v, z_e \in B$, they are not incident on any edge in F. Applying Lemma 4.1 we find that there are two possible cases:

1. There exists a chordless cycle in G on which v_1, v, v_2 occur consecutively, a contradiction.

2. There exists a chordless cycle in G on which v and z_e occur consecutively. But then at least one of v and z_e should have been moved to A by P_1' or P_2', a contradiction.

■

Theorem 4.3 *Let G be a graph and let $k_{opt} = \Phi(G)$. The algorithm finds a triangulation of G of size at most $8k_{opt}^2$, in time $O(k_{opt}nm + \min\{n^2M(k_{opt})/k_{opt}, nM(n)\})$.*

Proof: **Correctness:** By Theorems 4.2 and 2.4 a minimal triangulation of G_A' is a minimal triangulation of G'. Therefore at the end of the algorithm G is triangulated. Throughout the algorithm the only edges added to G are between vertices of A. Since $|A| \leq 4k_{opt}$ the size of the triangulation is at most $8k_{opt}^2$.

Complexity: The complexity analysis of procedures P_1 and P_2 in [16] implies that P_1' and P_2' can be performed in $O(k_{opt}nm)$ time. By Lemma 3.1 the complexity of P_3' is $O(nm + \min\{n^2M(k_{opt})/k_{opt}, nM(n)\})$. Procedure P_4' requires finding a minimal triangulation of G_A'. Since $|A| = O(\min\{k_{opt}, n\})$ and $|E(G_A')| = O(\min\{k_{opt}^2, n^2\})$, this requires $O(\min\{k_{opt}^3, n^3\})$ time [18]. Thus, the algorithm takes $O(k_{opt}nm + \min\{n^2M(k_{opt})/k_{opt}, nM(n)\})$ time. ■

5 Bounded Degree Graphs

In order to improve the approximation ratio for bounded degree graphs, we improve P_4'. Instead of simply finding a minimal triangulation of G_A', we use the triangulation algorithm of Agrawal, Klein and Ravi [1]. This alone does not suffice to prove a better approximation ratio, since adding 0-essential edges (in P_3') might increase the minimum fill-in size. To overcome this difficulty we use KST partition algorithm with $k = \infty$ as its input parameter. The approximation algorithm is as follows:

- Execute KST partition algorithm with $k = \infty$.
- Find a minimal triangulation of G_A using the algorithm in [1].

Assume that the input graph G has maximum degree d, and let $k = \Phi(G)$. We will show that the algorithm achieves an approximation ratio of $O(d^{2.5}\log^4(kd))$. Since $k = O(n^2)$, this is in fact a polylogarithmic approximation ratio. It improves over the $O(k)$ approximation ratio obtained in the previous section, when $k/\log^4 k = \Omega(d^{2.5})$.

Theorem 5.1 *The algorithm finds a triangulation of G of size within a factor of $O(d^{2.5}\log^4(kd))$ of optimal.*

Proof: **Correctness:** By the correctness of KST partition algorithm, we obtain a partition A, B for which no chordless cycle in G intersects B (a parameter $k = \infty$ implies that no new edges will be added to G_A by P_3). By Theorem 2.4 a minimal triangulation of G_A is a minimal triangulation of G. Therefore, the algorithm correctly computes a minimal triangulation of G.

Approximation Ratio: When executing P_3 the size of each set $A_{x,y}$ is at most d. By Lemma 3.3 $|A^3 - A^2| = O(kd)$. Since $|A^2| = O(k)$, the size of A when the partition terminates is $O(kd)$. Setting the parameter value to ∞ in P_3 guarantees that no new edge is added to G_A, and therefore its maximum degree remains d and $|E(G_A)| = O(kd^2)$. Using the algorithm in [1] we can produce a chordal super-graph of G_A with $O((kd^2 + k)\sqrt{d}\log^4(kd))$ edges. The size of the fill-in obtained is therefore within a factor of $O(d^{2.5}\log^4(kd))$ of optimal. ■

6 Reducing the Kernel Size

We now return to the parametric fill-in problem. By modifying procedure P_3 in KST partition algorithm we will obtain a partition A, B of V for which no chordless cycle in G' intersects B and $|A| = O(k)$. In fact we will obtain at most 2^k such partitions and prove that if G has a k-triangulation, then at least for one of those partitions G_A' admits a $(k-\alpha)$-triangulation, where $\alpha = |E(G_A') - E(G_A)|$. Reducing the size of A results in improving the complexity of finding a minimum triangulation of G_A' to $O(\sqrt{k}4^k)$, although the total time of the algorithm increases, since we have to handle up to 2^k partitions. We include this result since it gives further insight of the problem.

As in the original algorithm we start by executing procedures $P_1(k)$ and $P_2(k)$. We store the value of cc. We also compute the sets $A_{x,y}$ for all $x, y \in A^2, (x, y) \notin E$. If (x, y) is k-essential, we add it to G. Otherwise we do nothing. Denote the new graph obtained by $G' = (V, E')$. Define $P = \{(x, y) \notin E' : x, y \in A^2\}$.

The algorithm now enumerates all subsets $S \subseteq P$. For a given S, every $(x, y) \in S$ is added as an edge in the triangulation, and for every $(x, y) \in P - S$, the vertices in $A_{x,y}$ are moved from B to A. The algorithm is implemented by the recursive procedure below, and is invoked by calling BRANCH$(cc, \emptyset, P, A^2)$.

```
Procedure BRANCH($cc, F, P, A$)
  If $cc > k$ return.
  While there exists $(x, y) \in P$ such that
  $|A_{x,y} - A| = 1$ do:
    $A := A \cup A_{x,y}$.
    $P := P - \{(x, y)\}$.
    $cc := cc + 1/2$.
  Delete from $P$ all $(x, y)$ such that $A_{x,y} \subseteq A$.
  If $P = \emptyset$ save the pair $(A, F)$ and return.
  Choose any $(x, y) \in P$.
  Call BRANCH($cc + 1, F \cup \{(x, y)\}$,
                $P - \{(x, y)\}, A$).
  Call BRANCH($cc + |A_{x,y} - A|/2, F$,
                $P - \{(x, y)\}, A \cup A_{x,y}$).
  Return.
```

Lemma 6.1 *The algorithm terminates after at most $2^{k+2}-1$ calls to procedure BRANCH. The number of pairs saved by the algorithm is at most 2^k.*

Proof: Denote by $T(i)$ the number of calls to procedure BRANCH, when invoked with $cc = i$ (including the first call). Clearly $T(i) \leq 1 + 2T(i + 1)$, where $T(j) = 1$ for all $j > k$. The solution of this recursion gives $T(0) \leq 2^{k+2} - 1$.

Denote by $P(i)$ the number of pairs saved by procedure BRANCH, when invoked with $cc = i$. Clearly $P(i) \leq 2P(i + 1)$, where $P(k) \leq 1$. The solution of this recursion gives $P(0) \leq 2^k$. ■

As usual, for a set $A \subseteq V$ saved by the algorithm, B denotes $V - A$.

Theorem 6.2 *For every pair (A, F) saved by the algorithm, $|A| \leq 6k$, and no chordless cycle in $G' = (V, E \cup F)$ intersects B.*

Proof: Whenever a partition is saved $cc \leq k$. Since $|A - A^2|/2 \leq cc$, at most $2k$ new vertices were added to A^2 in any partition obtained. Since $|A^2| \leq 4k$ we conclude that $|A| \leq 6k$.

Assume to the contrary that C is a chordless cycle in G' intersecting B. Let $v \in C \cap B$. Let v_1 and v_2 be the neighbors of v on C. v is not the endpoint of any edge in F since $v \in B$. Every edge $e = (x, y) \in F$ is a chord in a chordless cycle $C_e = (x, z_e, y, \ldots, x)$, where $z_e \in B$ (since the only edges we add in the algorithm are pairs (x, y) such that there exist a vertex $b \in B$, and a chordless cycle in G, on which x, b, y occur consecutively). Applying Lemma 4.1 we find that two cases are possible:

1. There exists a chordless cycle in G on which v_1, v, v_2 occur consecutively. But then either v would have been moved to A by P_1 or P_2 or BRANCH, or (v_1, v_2) would have been added as an edge by BRANCH, a contradiction.
2. There exists a chordless cycle in G on which v and z_e (for some $e \in F$) occur consecutively. But then at least one of v and z_e would have been moved to A by P_1 or P_2, a contradiction.

■

Definition 6.3 *A pair (A, F) saved by BRANCH is called* good, *if $\Phi(G) = \Phi(G') + |F|$, where $G' = (V, E \cup F)$.*

Lemma 6.4 *If $\Phi(G) \leq k$ then at least one output of the algorithm is good.*

Proof: Let $\hat{A}, \hat{B}$ be a partition obtained by executing KST partition algorithm with the same input. Let F^* denote a minimum triangulation of G. After executing P_2, our algorithm produces the same initial partition A^2, B^2 as the KST algorithm. For that partition, define $S = \{(x, y) \notin E : x, y \in A^2, A_{x,y} \neq \emptyset\}$. By observation 3.2,

$$|F^*| \geq |S \cap F^*| + \frac{1}{2}|\bigcup_{(x,y)\in S - F^*} A_{x,y}|$$

Since $|F^*| \leq k$, procedure BRANCH will save the pair $(\hat{A}, S \cap F^*)$. ■

Theorem 6.5 *The complexity of the new partition algorithm is $O(knm + \min\{n^2M(k)/k, nM(n)\} + k^3 2^k)$.*

Proof: By [16] P_1 and P_2 take $O(knm)$ time. By Lemma 3.1 the complexity of computing the sets $A_{x,y}$ for all $x, y \in A^2, (x, y) \notin E$ is $O(nm + \min\{n^2M(k)/k, nM(n)\})$. By Lemma 6.1 the number of calls to BRANCH is $O(2^k)$. Since $|P| = O(k^2)$, $|A| = O(k)$ and $cc \leq k$, each call can be carried out in $O(k^3)$ time. The total work done by BRANCH is therefore $O(k^3 2^k)$. ■

7 An Approximation Algorithm for the Chain Completion Problem

A bipartite graph $G = (P, Q, E)$ is called a *chain graph* if there exists an ordering π of P, $\pi : P \to \{1, \ldots, |P|\}$, such that $N(\pi^{-1}(1)) \subseteq N(\pi^{-1}(2)) \subseteq \ldots \subseteq N(\pi^{-1}(|P|))$. This class of graphs was introduced by Yannakakis [22], and independently by Golumbic (cf. [15, page 260]). The *chain completion* problem is defined as follows: Given a bipartite graph $G = (P, Q, E)$, find a minimum set of non-edges F such that $(P, Q, E \cup F)$ is a chain graph. We call $|F|$ the *chain fill-in*. Yannakakis proved that the chain completion problem is NP-complete and used this result to show that the minimum fill-in problem is NP-complete [22]. Chain graphs have also recently been investigated in [8], where a similar graph modification problem arises.

Theorem 7.1 *There exists a polynomial approximation algorithm for the chain completion problem, achieving an approximation ratio of $8k$, where k denotes the minimum chain fill-in. The complexity of the algorithm is $O(kn^3)$.*

Proof: Let $G = (U, V, E)$ be an input bipartite graph with chain fill-in k. We apply the reduction given by Yannakakis [22] from the chain completion problem to the minimum fill-in problem, as follows: Build a graph $G' = (U \cup V, E')$, where $E' = E \cup \{(u,v) : u,v \in U\} \cup \{(u,v) : u,v \in V\}$. Observe that G is a chain graph iff G' is chordal. Hence, a set of edges F triangulates G' iff $(U, V, E \cup F)$ is a chain graph.

Approximation Ratio: By the above argument k equals $\Phi(G')$. Using our approximation algorithm for the minimum fill-in, we can find a triangulation of G' of size at most $8k^2$. Adding these edges to G produces a chain graph. The number of new edges is within a factor of $8k$ of optimal.

Complexity: G' can be computed in $O(n^2)$ time. Due to the reduction $|E(G')| = \Theta(n^2)$. Therefore the complexity of the approximation algorithm is $O(kn^3)$. ■

Acknowledgments

We thank Itsik Pe'er for many useful remarks.

References

[1] A. Agrawal, P. Klein, and R. Ravi. Cutting down on fill using nested dissection: provably good elimination orderings. In A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, pages 31–55. Springer, 1993.

[2] J. R. Bunch and D. J. Rose, editors. *Sparse Matrix Computations*. Academic Press, 1976.

[3] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58:171–176, 1996.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, Mar. 1990.

[5] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. 24th Nat. Conf. ACM*, pages 157–172, 1969.

[6] R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1997. To appear.

[7] I. Duff, editor. *Sparse matrices and their uses*. Academic press, 1981.

[8] D. Fasulo, T. Jiang, R. M. Karp, and N. Sharma. Constructing maps using the span and inclusion relations. Technical report, University of Washington, Dept. of Computer Science and Engineering, 1997. To appear in Proc. RECOMB '98.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[10] A. George. Nested dissection of a regular finite element mesh. *SIAM J. on Numerical Analysis*, 10:345–367, 1973.

[11] A. George, J. R. Gilbert, and J. W. H. Liu, editors. *Graph Theory and Sparse Matrix Computation*. Springer, 1993.

[12] A. George and J. W. Liu. *Computer solution of large sparse positive definite systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[13] A. George and J. W. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[14] J. R. Gilbert. Some nested dissection order is near optimal. *Inf. Proc. Letts.*, 26:325–328, 1988.

[15] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[16] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping. In *Proceedings of the 35th Symposium on Foundations of Computer Science*, pages 780–791. IEEE Computer Science Press, Los Alamitos, California, 1994. To appear in *SIAM J. Computing*.

[17] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms. In *Proc. 29th FOCS*, pages 101–111. IEEE, 1988.

[18] T. Ohtsuki. A fast algorithm for finding an optimal ordering for vertex elimination on a graph. *SIAM J. Computing*, 5:133–145, 1976.

[19] T. Ohtsuki, L. K. Cheung, and T. Fujisawa. Minimal triangulation of a graph and optimal pivoting order in a sparse matrix. *Journal of Math. Anal. Appl.*, 54:622–633, 1976.

[20] J. D. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Reed, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, N.Y., 1972.

[21] U. Schendel. *Sparse matrices: numerical aspects with applications for scientists and engineers*. Ellis Horwood, 1989.

[22] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Alg. Disc. Meth.*, 2, 1981.