SIAM J. COMPUT.
Vol. 30, No. 4, pp. 1067–1079

A POLYNOMIAL APPROXIMATION ALGORITHM FOR THE MINIMUM FILL-IN PROBLEM*

ASSAF NATANZON†, RON SHAMIR†, AND 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.

Key words. approximation algorithms, parameterized algorithms, graph algorithms, minimum fill-in, chordal graphs, chain graphs, chordal completion, chain completion

AMS subject classifications. 68W25, 68W99, 05C85, 05C99

PII. S0097539798336073

1. Introduction.

A *chord* in a cycle is an edge between nonconsecutive vertices on the cycle. A *chordless cycle* is a cycle of length greater than 3 that contains no chord. A graph is called *chordal* or *triangulated* if it contains no chordless cycle. 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 denote by $\Phi(G)$ 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 nonzeros. Different elimination orders may introduce different sets of new nonzero 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 nonzeros (even temporarily). Rose [21] proved that the problem of finding an elimination order for a symmetric positive-definite matrix M, such that fewest new nonzero 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 if and only if $M_{i,j} \neq 0$.

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 [23]. Due to its importance the problem has been studied intensively [2, 11, 13, 22], and many heuristics have been developed for it [5, 12, 20, 21]. None of those gives a performance guarantee with respect to the size

*Received by the editors March 23, 1998; accepted for publication (in revised form) March 22, 2000; published electronically August 29, 2000. Portions of this paper appeared in the Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998 [18].

http://www.siam.org/journals/sicomp/30-4/33607.html

†Department of Computer Science, Tel Aviv University, Tel Aviv, Israel (natanzon@math.tau.ac.il, shamir@math.tau.ac.il, roded@math.tau.ac.il). The research of the second author was supported by a grant from the Ministry of Science, Israel. The research of the third author was supported by an Eshkol Fellowship from the Ministry of Science, Israel.

of the fill-in introduced. Note that in contrast, the *minimal* fill-in problem (finding a triangulation of G which is minimal with respect to inclusion) is polynomial [19].

Approximation attempts succeeded only for the related *minimum triangulated supergraph* 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 may differ drastically as approximation problems. For example, if the input graph has $\Omega(n^2)$ edges and fill-in of size $o(n^2)$, then one can trivially achieve a constant approximation ratio for MTS by making the graph an n-clique (a complete graph), while no such approximation guarantee exists for the minimum fill-in problem. (Throughout we use n and m to denote the number of vertices and edges, respectively, in a graph.) The approximation results regarding MTS use the nested dissection heuristic first proposed by George [10] (see [13] 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. The result was not constructive as one has yet to 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 for an input graph G a chordal supergraph with total number of edges $O((m + \Phi(G))^{3/4} \sqrt{m} \log^{3.5} n)$.

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 $2^{O(k)} n^{O(1)}$ 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] and later independently Cai [3] proved that the minimum fill-in problem is fixed parameter tractable, by giving an algorithm of complexity $2^{O(k)}m$ for the problem. Both used the same algorithm, with the time bound in [3] being slightly tighter. Kaplan, Shamir, and Tarjan also gave a more efficient $2^{O(k)}+O(k^2nm)$-time algorithm (henceforth, the KST algorithm) for the problem.

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 $8k^2$, i.e., 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 integer matrices of order $n \times n$. (The current upper bound on $M(n)$ is $O(n^{2.376})$ [4].) The algorithm works in time $O(knm + \min\{n^2 M(k)/k, nM(n)\})$, 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 supergraph obtained is $O((m + k)^{3/4}\sqrt{m}\log^{3.5} n)$. In terms of the fill-in obtained, 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, Shamir, and Tarjan posed in [16] an open problem of obtaining an algorithm for the parametric fill-in problem with time $2^{O(k)} + O(km)$. The motivation is to match the performance of the $2^{O(k)}m$ algorithm for all k. We make some progress towards solving that problem by providing a faster $2^{O(k)} + O(knm + \min\{n^2M(k)/k, nM(n)\})$-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 optimum solution.

The paper is organized as follows. Section 2 contains a description of the KST algorithm and some background. Section 3 improves the complexity of the 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 V of vertices also by $V(G)$ and its set E 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 the KST algorithm [16]. In the following we describe this algorithm. Our presentation generalizes that in [16] in order to allow succinct description of the approximation algorithm in section 4.

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

LEMMA 2.2 (see [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 $\langle G = (V, E), k\rangle$ 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, and k, the

algorithm produces a partition A, B of V and a set F of nonedges 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 search procedure can be viewed as traversing part of a search tree T, which is defined as follows. Each tree node v corresponds to a supergraph $G(v)$ of G. For the root r, $G(r) = G$. Each leaf of T corresponds to a chordal supergraph of G. At an internal node v, a chordless cycle C in $G(v)$ is identified. For each minimal triangulation F_C of C, a node u is added as a child of v, and its corresponding graph $G(u)$ is obtained by adding F_C to $G(v)$. The algorithm visits only nodes v of T for which $|E(G(v))\backslash E| \leq k$. If such a node is a leaf, then the search terminates successfully. Otherwise, no k-triangulation exists for 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 $cc=0$.

(i) *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.

(ii) *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 subpaths in G_B, each of length at least 1. Move the vertices of those subpaths from B to A. Denote their lengths in nonincreasing 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.

(iii) *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 k-essential, 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 shall omit the index i when it is clear from the context. Denote by cc_i the value of cc after procedure P_i is completed for $i = 1, 2$. The size of A^2 is at most $4k$ since $k \geq cc_2 = cc_1 + (cc_2 - cc_1) \geq \frac{1}{4}|A^1| + \frac{1}{2}|A^2 \setminus A^1| \geq \frac{1}{4}|A^2|$. The size of A^3 is $O(k^3)$ since there are $O(k^2)$ nonedges in G_{A^2} and the number of vertices moved to A due to any such nonedge is at most $2k$.

The partition algorithm is summarized in Figure 2.1. Let G' denote the graph obtained after the execution of procedure P_3. Kaplan, Shamir, and Tarjan prove

Execute procedure $P_1(k)$.
Execute procedure $P_2(k)$.
Execute procedure $P_3(k)$.

FIG. 2.1. *The KST partition algorithm.*

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 (see [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 if and only if 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 the 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 KST partition algorithm. We assume throughout that the input is $\langle G = (V, E), k\rangle$. 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 the KST algorithm can be implemented in $O(knm + \min\{n^2M(k)/k, nM(n)\} + k^{2.5}4^k)$-time.

LEMMA 3.1. *There is an $O(nm+\min\{n^2M(k)/k, nM(n)\})$-time implementation of procedure P_3.*

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 this end, 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 $G^b = G \setminus N[b]$. This takes $O(m)$ time. Denote the connected components of G^b by $C^b_1, \ldots, C^b_l$. For each $x \in A^2 \cap N(b)$ compute a binary vector $\vec{v}_x = (v^x_1, \ldots, v^x_l)$ such that $v^x_j = 1$ if and only if C^b_j contains a neighbor of x, $1 \le j \le l$. Each vector can be computed in $O(n)$-time. Let $k' = |A^2 \cap N(b)|$, and number the vertices in $A^2 \cap N(b)$ arbitrarily according to some 1-1 mapping $\sigma : \{1, \ldots, k'\} \to A^2 \cap N(b)$. Define a $k' \times l$ boolean matrix M whose ith row is the vector $\vec{v}_{\sigma(i)}$, $1 \le i \le k'$. Note that $k' = O(k)$ and $l \le n$. Let $M^* = MM^T$. It can be seen that for every pair (i, j) such that $1 \le i < j \le k'$ and $(\sigma(i), \sigma(j)) \in S$, $M^*_{i,j} \ge 1$ if and only if $b \in A_{\sigma(i),\sigma(j)}$. Since $k', l \le n$ we can compute M^* in $O(M(n))$-time. If $k = o(n)$, then we can compute M^* in $O(nM(k)/k)$-time by partitioning M and M^T into $\lceil n/k'\rceil$ submatrices of order at most $k' \times k'$, multiplying corresponding pairs of submatrices, and summing the results. 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 $A_{x,y} \ne \emptyset$, 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 $|A^3 \setminus A^2| \le Mk$, where $M = \max\{d, 2\}$.*

Proof. Let the nonedges 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 $\Delta_i = A_{x_i,y_i} \setminus A^{(i-1)}$ for $1 \le i \le l$.

Let t be a lower bound on the minimum number of edges needed to triangulate G. Initially P_3 starts with $t = 0$. Let t_i be the value of t right after A_{x_i,y_i} was processed ($t_0 = 0$). 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,

$$|A^3 \setminus A^2| = |A^{(l)} \setminus A^{(0)}| = \sum_{i=1}^{l} |A^{(i)} \setminus A^{(i-1)}|$$
$$= \sum_{i=1}^{l} |\Delta_i| \leq M \sum_{i=1}^{l} (t_i - t_{i-1}) = M(t - t_0) \leq Mk\,. \qquad \square$$

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 k-essential edges were added to G, and denote the new set of edges of 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, $|A^3 \setminus A^2| \leq 2k^2$. Since $|A^2| \leq 4k$, the corollary follows. $\square$

THEOREM 3.5. *There is an $O(knm + \min\{n^2 M(k)/k, nM(n)\} + k^{2.5}4^k)$-time implementation of the KST algorithm.*

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 [3]. The complexity follows. $\square$

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 $O(k_{opt}^2)$ nonedges in G_A, 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 shown in Figure 4.1.

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

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.

FIG. 4.1. *The approximation algorithm.*

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 the execution of 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 what follows. We need the following lemma which is implied by the proof of [16, Lemma 2.9]. The subsequent theorem is a generalization 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 nonedges in $G \setminus \{v\}$, 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 for some $v_1, v_2 \in N(v)$, 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. *Let $G = (V, E)$ be a graph. Let A, B be a partition of V such that no chordless cycle in G contains two consecutive vertices from B. Let $S = \{(x, y) \notin E : x, y \in A, A_{x,y} \neq \emptyset\}$. Then for any choice of $F \subseteq S$ no chordless cycle in $G' = (V, E \cup F)$ intersects $B' = B \setminus (\bigcup_{(x,y)\in S\setminus F} A_{x,y})$.*

Proof. Suppose 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. Since $v \in B'$, it is not an endpoint of any edge in F. Every edge $e = (x, y) \in F$ is a chord in a chordless cycle $C_e = (x, z_e, y, \ldots, x)$ of G, where $z_e \in B$. 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. If $v_1 \in B$ or $v_2 \in B$, we arrive at a contradiction. Hence, $v_1, v_2 \in A$ and $v \in A_{v_1,v_2}$. We conclude that either $(v_1, v_2) \in F$ or $v \notin B'$, a contradiction.

2. There exists a chordless cycle in G on which v and z_e occur consecutively (for some $e \in F$), 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 $8k_{opt}^2$ and can be implemented to run 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 [19]. Hence, the complexity of the approximation algorithm is $O(k_{opt}nm + \min\{n^2M(k_{opt})/k_{opt}, nM(n)\})$. ∎

Note that, although our analysis uses an upper bound of $\binom{t}{2}$ for the triangulation size of a t-vertex graph, replacing G'_A by the complete graph is not guaranteed to produce a triangulation of G.

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 not be optimal. In other words, if we denote by F the set of 0-essential edges added to G by P_3', then it might be that $|F| + \Phi(G') > \Phi(G)$. To overcome this difficulty we use the KST partition algorithm with $k = \infty$ as its input parameter, which implies that no new edge will be added to G_A by P_3. The approximation algorithm is as follows:

(i) Execute the KST partition algorithm with parameter $k = \infty$.
(ii) 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 the KST partition algorithm, we obtain a partition A, B of $V(G)$ for which no chordless cycle in G intersects B. 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, $|A^3 \setminus A^2| = O(kd)$. Since $|A^2| = O(k)$, the size of A when the partition algorithm 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 is at most d and $|E(G_A)| = O(kd^2)$. Using the algorithm in [1] we can produce a chordal supergraph 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. Let $\langle G = (V, E), k\rangle$ be the input instance. By modifying procedure P_3 in the KST partition algorithm we shall obtain a partition A, B of V and a set of nonedges F, such that no chordless cycle in $G' = (V, E \cup F)$ intersects B and $|A| = O(k)$. In fact we shall obtain at most 2^k such pairs (A, F) and prove that if G has a k-triangulation, then at least for one of those pairs G'_A admits a $(k - |F|)$-triangulation. 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 pairs. We include this result, since it gives further insight on the problem and presents ideas that may help resolve the open problem posed in [16].

As in the original algorithm we start by executing procedures $P_1(k)$ and $P_2(k)$. 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. Let $\hat{E}$ be the set of k-essential edges, and let $e = |\hat{E}|$. Define $P := \{(x, y) \notin E \cup \hat{E} : x, y \in A^2, A_{x,y} \neq \emptyset\}$, and let $p = |P|$.

The algorithm now enumerates subsets $F \subseteq P$. For a given set F, every $(x, y) \in F$ is added as an edge in the triangulation, and for every $(x, y) \in P \setminus F$, the vertices in $A_{x,y}$ are moved from B to A (which was initialized to A^2). Instead of directly enumerating each set F, we branch and bound. We construct these sets incrementally and stop when a lower bound for the size of the triangulation implied by F exceeds k.

Specifically, the algorithm considers pairs in P one at a time in an arbitrary order $(x_0, y_0), \ldots, (x_{p-1}, y_{p-1})$. For the current pair (x_i, y_i) it distinguishes between three cases as follows. Let $t = |A_{x_i,y_i} \setminus A|$ with respect to the current A. Let cc denote a lower bound for the size of the triangulation implied by the set F constructed so

far (cc is initialized to e). If $t = 0$, then the algorithm does nothing. If $t = 1$, it updates A to $A \cup A_{x_i,y_i}$ and increases cc by $1/2$. Finally, if $t \geq 2$, then the algorithm branches into two cases. In the first case, (x_i, y_i) is added to the triangulation and cc is increased by 1. In the second case, the vertices in A_{x_i,y_i} are moved from B to A, and cc is increased by $t/2$. The algorithm is implemented by the recursive procedure shown in Figure 6.1 and is invoked by calling BRANCH$(e, \emptyset, 0, A^2)$.

```
Procedure BRANCH(cc, F, r, A)
  If cc > k then return.
  If r = p then save the pair (A, F ∪ Ê) and return.
  Let t = |A_{x_r,y_r} \ A|.
  If t = 0 then
    Call BRANCH(cc, F, r + 1, A).
  Else if t = 1 then
    Call BRANCH(cc + 1/2, F, r + 1, A ∪ A_{x_r,y_r}).
  Else /* t ≥ 2 */
    Call BRANCH(cc + 1, F ∪ {(x_r, y_r)}, r + 1, A).
    Call BRANCH(cc + t/2, F, r + 1, A ∪ A_{x_r,y_r}).
  Return.
```

FIG. 6.1. *Algorithm BRANCH.*

LEMMA 6.1. *The algorithm terminates after at most* $p2^{k+1} + 1$ *calls to procedure BRANCH.*

Proof. Denote by $T(i, j)$ the number of recursive calls invoked by BRANCH when called with parameters $cc = i, r = j$ (including this first call). Since always $i \geq 0$ and $0 \leq j \leq p$ in the following, we consider these ranges only. Clearly, $T(i, j) \leq 1 + \max\{T(i, j + 1), T(i + 1/2, j + 1), 2T(i + 1, j + 1)\}$ for all $j < p, i$. Also, $T(i, j) = 1$ for all $i > k, j$, and $T(i, p) = 1$ for all i. It follows that $T(0, 0) \geq T(i, j)$ for all i, j. Hence, it suffices to compute an upper bound for $T(0, 0)$.

We prove that $T(i, j) \leq (p - j)2^{k+1-i} + 1$ by induction on i, j. For $i > k$ or $j = p$ the claim is true. Suppose the claim holds for all i, where $i' \leq i \leq k + 1$, and for all j, where $j' < j \leq p$. Then for $i = i'$ and $j = j'$ we have

$$\begin{aligned} T(i, j) &\leq 1 + \max\{T(i, j + 1), T(i + 1/2, j + 1), 2T(i + 1, j + 1)\} \\ &\leq 2 + \max\{(p - j - 1)2^{k+1-i}, (p - j - 1)2^{k+\frac{1}{2}-i}, (p - j - 1)2^{k+1-i} + 1\} \\ &\leq 3 + (p - j - 1)2^{k+1-i} \leq (p - j)2^{k+1-i} + 1 \, . \end{aligned}$$

It follows that $T(0, 0) \leq p2^{k+1} + 1$. ∎

LEMMA 6.2. *The number of pairs saved by the algorithm is at most* 2^k.

Proof. The proof is analogous to that of Lemma 6.1. Denote by $N(i, j)$ the number of pairs saved by procedure BRANCH, when invoked with parameters $cc = i, r = j$. Since always $i \geq 0$ and $0 \leq j \leq p$, in the following we consider these ranges only. Clearly, $N(i, j) \leq \max\{N(i, j + 1), N(i + 1/2, j + 1), 2N(i + 1, j + 1)\}$ for all $j < p, i$. Also, $N(i, j) = 0$ for all $i > k, j$, $N(k, j) \leq 1$ for all j, and $N(i, p) \leq 1$ for all i. It follows that $N(0, 0) \geq N(i, j)$ for all i, j. Thus, it suffices to compute an upper bound for $N(0, 0)$.

We prove that $N(i,j) \leq 2^{k-i}$ by induction on i,j. If $i \geq k$ or $j = p$, then the claim holds. Suppose the claim holds for all i, where $i' \leq i \leq k$, and for all j, where $j' < j \leq p$. Then for $i = i'$ and $j = j'$ we have

$$\begin{aligned} N(i,j) &\leq \max\{N(i,j+1), N(i+1/2,j+1), 2N(i+1,j+1)\} \\ &\leq \max\{2^{k-i}, 2^{k-\frac{1}{2}-i}, 2^{k-i}\} \\ &= 2^{k-i} . \end{aligned}$$

It follows that $N(0,0) \leq 2^k$. □

As usual, for a set $A \subseteq V$ saved by the algorithm, B denotes $V \setminus A$. The following two claims establish the correctness of our partition algorithm.

LEMMA 6.3. *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, $cc \leq k$. By definition of BRANCH, $\frac{1}{2}|A \setminus A^2| \leq cc$. Hence, at most $2k$ new vertices were added to A^2 in any partition obtained. Since $|A^2| \leq 4k$, we conclude that $|A| \leq 6k$. By Theorem 4.2 no chordless cycle in G' intersects B. □

DEFINITION 6.4. *A pair (A,F) saved by BRANCH is called* good *if $\Phi(G) = \Phi(G') + |F|$, where $G' = (V, E \cup F)$.*

PROPOSITION 6.5. *If $\Phi(G) \leq k$, then at least one pair saved by the algorithm is good.*

Proof. Let T be the tree of recursive calls of BRANCH. The nodes of T correspond to invocations of BRANCH. The root of T corresponds to the first invocation of BRANCH. The leaves of T correspond to invocations of BRANCH in which either a pair was saved, or cc was found to exceed k. In nodes at level i of T, $0 \leq i < p$, the pair $(x_i, y_i) \in P$ is processed. Let cc_v, F_v, r_v, and A_v denote the parameters of the invocation of BRANCH which correspond to node v of T.

Let F^* denote a minimum triangulation of G. The proof will identify a root-leaf path in T which corresponds to F^*, and trace the changes to cc, A, and F along that path. We use the following notation:

$$\begin{aligned} P_v &:= \{(x_0,y_0), \ldots, (x_{r_v-1}, y_{r_v-1})\} , \\ F_v^* &:= P_v \cap F^* , \\ A_v^* &:= A^2 \cup \bigcup_{(x,y) \in P_v \setminus F_v^*} A_{x,y} , \\ cc_v^* &:= e + |F_v^*| + \frac{1}{2}|A_v^* \setminus A^2| . \end{aligned}$$

LEMMA 6.6. *For every node v of T, $cc_v^* \leq k$.*

Proof. Let v be any node of T. Let $cc^* = e + |P \cap F^*| + \frac{1}{2}|\bigcup_{(x,y) \in P \setminus F^*} A_{x,y}|$. Since $P_v \subseteq P$, it follows that $cc_v^* \leq cc^*$. By Observation 3.2, for every pair $(x,y) \in P$, either $(x,y) \in F^*$, or for every $b \in A_{x,y}$, F^* contains an edge incident on b. Hence, $cc^* \leq |F^*| \leq k$, where the last inequality follows from the fact that F^* is a k-triangulation. □

We now return to the proof of Proposition 6.5. We shall prove that T has a leaf in which a good pair is saved. To this end, we show that for every $0 \leq i \leq p$, T contains some vertex v at level i for which $F_v \subseteq F_v^*$ and $cc_v \leq cc_v^*$. In particular, this claim implies that T has a leaf z at level p for which $F_z \subseteq F_z^*$ and $cc_z \leq cc_z^*$. By Lemma 6.6, $cc_z \leq cc_z^* \leq k$. Hence, the pair $(A_z, F_z \cup \hat{E})$ is saved at z. By [16, Lemma

2.7], $\hat{E} \subseteq F^*$. In addition, $F_z \subseteq F_z^* \subseteq F^*$. Therefore $(A_z, F_z \cup \hat{E})$ is a good pair, since $F_z \cup \hat{E} \subseteq F^*$ and, by definition, $F^* \setminus (F_z \cup \hat{E})$ triangulates $G' = (V, E \cup F_z \cup \hat{E})$.

We prove the claim by induction on i. The base of the induction is obvious, and as for the root r at level 0, $F_r = \emptyset$ and $cc_r = e$. We assume that the claim is true for level $i-1$ $(i > 0)$ and prove its correctness for level i. By the induction hypothesis T contains a node v at level $i-1 < p$ for which $F_v \subseteq F_v^*$ and $cc_v \leq cc_v^*$. By Lemma 6.6 $cc_v \leq cc_v^* \leq k$, and therefore v is not a leaf. Thus, v has either one or two children in T. There are two cases to examine.

1. Suppose that $(x_i, y_i) \in F^*$. Then for any child w of v, $cc_w^* = cc_v^* + 1 \geq cc_v + 1$. If v has a single child w, then $F_w = F_v \subseteq F_v^* \subset F_w^*$ and $cc_w \leq cc_v + 1/2 < cc_w^*$. Otherwise, let w be the child of v for which $(x_i, y_i) \in F_w$. Then clearly $F_w \subseteq F_w^*$ and $cc_w = cc_v + 1 \leq cc_w^*$.

2. Suppose that $(x_i, y_i) \notin F^*$. Since $F_v \subseteq F_v^*$ and $A_v = A^2 \cup \bigcup_{(x,y) \in P_v \setminus F_v} A_{x,y}$, it follows that $A_v^* \subseteq A_v$. Let w be the child of v for which $(x_i, y_i) \notin F_w$. Then $F_w = F_v \subseteq F_v^* = F_w^*$ and

$$cc_w = cc_v + \frac{1}{2}|A_{x_i,y_i} \setminus A_v| \leq cc_v^* + \frac{1}{2}|A_{x_i,y_i} \setminus A_v^*| = cc_w^* . \qquad \square$$

THEOREM 6.7. *If $\Phi(G) \leq k$, then the new partition algorithm produces at least one pair (A, F) for which $|A| \leq 6k$ and $\Phi(G) = \Phi(G'_A) + |F|$, where $G' = (V, E \cup F)$. The complexity of the algorithm is $O(knm + \min\{n^2 M(k)/k, nM(n)\} + k^3 2^k)$.*

Proof. Correctness. By Lemma 6.3 for each pair (A, F) saved by the algorithm, $|A| \leq 6k$ and no chordless cycle in G' intersects B. Therefore, by Theorem 2.4 for each such pair $\Phi(G') = \Phi(G'_A)$. Since $\Phi(G) \leq k$, by Proposition 6.5 the algorithm saves some pair (A, F) for which $\Phi(G) = \Phi(G') + |F|$. Correctness follows.

Complexity. By [16] P_1 and P_2 take $O(knm)$ time. By Lemma 3.1, computing the sets $A_{x,y}$ for all $x, y \in A^2, (x, y) \notin E$ takes $O(nm + \min\{n^2 M(k)/k, nM(n)\})$ time. By Lemma 6.1 and the fact that $|P| = O(k^2)$, the number of calls to BRANCH is $O(k^2 2^k)$. By Lemma 6.3 and since $\Phi(G) \leq k$, the parameters A and F to each invocation of BRANCH satisfy $|A| = O(k)$ and $|F| \leq k$. Also, for all $(x, y) \in P$, $|A_{x,y}| \leq 2k$. Thus, each call can be carried out in $O(k)$ time. The total work done by BRANCH is therefore $O(k^3 2^k)$. $\square$

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 $N(\pi^{-1}(1)) \subseteq N(\pi^{-1}(2)) \subseteq \cdots \subseteq N(\pi^{-1}(|P|))$. This class of graphs was introduced by Yannakakis [23], 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 nonedges 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 [23]. Chain graphs have been also 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 following reduction given by Yannakakis [23] from the chain completion problem to the minimum fill-in problem. 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

if and only if G' is chordal. Hence, a set of edges F triangulates G' if and only if $(U, V, E \cup F)$ is a chain graph.

Approximation ratio. By the above argument, k equals $\Phi(G')$. Using our approximation algorithm for the minimum fill-in problem, 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. We also thank an anonymous referee for many helpful suggestions.

REFERENCES

[1] A. Agrawal, P. Klein, and R. Ravi, *Cutting down on fill using nested dissection: Provably good elimination orderings*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, New York, 1993, pp. 31–55.

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

[3] L. Cai, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inform. Proc. Lett., 58 (1996), pp. 171–176.

[4] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput., 9 (1990), pp. 251–280.

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

[6] R. Downey and M. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[7] I. Duff, ed., *Sparse Matrices and Their Uses*, Academic Press, New York, 1981.

[8] D. P. Fasulo, T. Jiang, R. M. Karp, and N. Sharma, *Constructing maps using the span and inclusion relations*, in Proceedings of the 2nd International Conference on Computational Molecular Biology, ACM, New York, 1998, pp. 64–73.

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

[10] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[11] A. George, J. R. Gilbert, and J. W. H. Liu, eds., *Graph Theory and Sparse Matrix Computation*, Springer-Verlag, New York, 1993.

[12] A. George and J. Liu, *The evolution of the minimum degree ordering algorithm*, SIAM Rev., 31 (1989), pp. 1–19.

[13] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[14] J. R. Gilbert, *Some nested dissection order is near optimal*, Inform. Process Lett., 26 (1988), pp. 325–328.

[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, strongly chordal, and proper interval graphs*, SIAM J. Comput., 28 (1999), pp. 1906–1922.

[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 Proceedings of the 29th Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, CA, 1988, pp. 101–111.

[18] A. Natanzon, R. Shamir, and R. Sharan, *A polynomial approximation algorithm for the minimum fill-in problem*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, pp. 41–47.

[19] T. Ohtsuki, *A fast algorithm for finding an optimal ordering for vertex elimination on a graph*, SIAM J. Comput., 5 (1976), pp. 133–145.

[20] T. Ohtsuki, L. K. Cheung, and T. Fujisawa, *Minimal triangulation of a graph and optimal pivoting order in a sparse matrix*, J. Math. Anal. Appl., 54 (1976), pp. 622–633.

[21] J. D. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Reed, ed., Academic Press, New York, 1972, pp. 183–217.

[22] U. Schendel, *Sparse Matrices: Numerical Aspects with Applications for Scientists and Engineers*, Ellis Horwood, Chichester, UK, 1989.

[23] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Discrete Methods, 2 (1981), pp. 77–79.