MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

A Benders Approach to the Minimum Chordal Completion Problem

Bergman, D.; Raghunathan, A.U.

TR2015-056 May 2015

Abstract

This paper introduces an integer programming approach to the minimum chordal completion problem. This combinatorial optimization problem, although simple to pose, presents considerable computational difficulties and has been tackled mostly by heuristics. In this paper, an integer programming approach based on Benders decomposition is presented. Computational results show that the improvement in solution times over a simple branch-and-bound algorithm is substantial. The results also indicate that the value of the solutions obtained by a state-of-the-art heuristic can be in some cases significantly far away from the previously unknown optimal solutions obtained via the Benders approach.

2015 International Conference on Integration of Artificial Intelligence and Operations Research (CPAIOR)

A Benders Approach to the Minimum Chordal Completion Problem

David Bergman[1] and Arvind U. Raghunathan[2]

[1] University of Connecticut, 1 University Place, Stamford, CT
david.bergman@business.uconn.edu
[2] Mitsubishi Electric Research Labs, 201 Broadway, Cambridge, MA
raghunathan@merl.com

Abstract. This paper introduces an integer programming approach to the minimum chordal completion problem. This combinatorial optimization problem, although simple to pose, presents considerable computational difficulties and has been tackled mostly by heuristics. In this paper, an integer programming approach based on Benders decomposition is presented. Computational results show that the improvement in solution times over a simple branch-and-bound algorithm is substantial. The results also indicate that the value of the solutions obtained by a state-of-the-art heuristic can be in some cases significantly far away from the previously unknown optimal solutions obtained via the Benders approach.

1 Introduction

Given graph G, the *minimum chordal completion problem* (MCCP) asks for a minimum cardinality set of edges whose addition to G results in a *chordal graph* (a graph for which every cycle consisting of four or more vertices contains a *chord* - an edge connected vertices that do not appear consecutively in the cycle). The problem is also known as the *minimum fill-in problem* and the *minimum triangulation problem*.

Chordal completions find applications in a variety of fields. These include sparse matrix computation and semidefinite programming [31,18,23,27], database management [1,34], computer vision [12], many others (the interested reader may refer to a survey on the topic [19]). In addition to these applications, chordal completions are related to the *tree-width problem* [6], the *minimum interval completion problem* [25], and are a special case of graph sandwich problems [17].

Although the problem has applications in a variety of domains, computational approaches in the literature have been very limited, with the focus being on developing heuristics. The MCCP (in its decision version which ask whether or not a chordal completion containing fewer than k edges exists) was listed as one of the open problems in Garey and Johnson's classical book on computational complexity [15] and later proven NP-complete [36].

Surprisingly, the optimization community has largely overlooked computational approaches to the problem. Some algorithms are published which solve

the MCCP exactly on particular classes of graphs [12,9,24,11,33,7]. For general graphs algorithms exists as well, although computational results have not been reported. The first fixed parameter tractable algorithm [22] was proven to have time complexity $O\left(2^{O(k)} + k^2 nm\right)$ (where n is the number of vertices in the graph and m is the number of edges in the graph). Algorithms have since investigated [8] and recently the running time has been reduced to sub-exponential parameterized time complexity [13].

As opposed to exact algorithms, the literature is vast on algorithms designed to find *minimal* chordal completions (the interested reader can again refer to a survey written on the topic [19]). The objective of these algorithms are twofold. First, they seek to create chordal completion in the least possible computational time. Second, they search for chordal completions using as few edges as possible. A surprisingly simple algorithm based on ordering the vertices of the graph by their degree and running a *vertex elimination game* [16] runs in polynomial time [20] ($O(n^2 m)$) and produces very good solutions in terms of the number of edges added, compared to other heuristics [26,3,4,5,29].

This paper presents an integer programming approach to the MCCP based on Benders decomposition [2], and in particular on logic-based Benders decomposition [21]. The problem is modeled using only a quadratic (with respect to the number of vertices in the graph) number of variables (one per edge not in the graph), and cuts are added iteratively when they become violated. The problem is decomposed into a master problem and a subproblem. In the master problem, a solution is found which adds some subset of the complement edge set to the graph subject to Benders cuts that have been previously identified. This graph then defines a subproblem which returns either that the graph is chordal (and hence a minimum chordal completion) or finds chordal cuts that must be satisfied by any chordal completion of the graph. In the latter case, the cuts are added to the master problem, and any solution found by the master problem in subsequent iterations will never violate these cuts.

The remainder of the paper is organized as follows. In Section 2 graph notation is introduced which will be used throughout the paper. Section 3 formally introduces the MCCP. Section 4 introduces the Benders decomposition approach to the MCCP with Section 5 explaining in detail the solution to the subproblem and Section 6 describing the Bender' cuts and proving that they are valid. Section 7 details the exact algorithm and heuristic algorithms used for comparison in the computational experiments presented in Section 8. The paper concludes in Section 9.

2 Graph Notation

Let S be any set. $\binom{S}{2}$ denotes the family of two-element subsets of S.

Let $G = (V, E)$ be a graph. For the remainder of the paper it is assumed that G is connected, undirected, has no self-loops or multi-edges.

Each edge $e \in E \subseteq \binom{V}{2}$ is a two-element subset of V. We denote by E^c the *complement edge set* of G: $E^c = \binom{V}{2} \backslash E$.

The *neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices which share an edge with (are *adjacent* to) v: $N(v) = \{v' : (v, v') \in E\}$. The *closed neighborhood* $N[v]$ is the neighborhood of v together with v: $N[v] = N(v) \cup \{v\}$.

Let $V' \subseteq V$. The graph *induced* by $V', G[V'] = (V', E(V'))$ is the graph on vertex set V' with edge set $E(V') = E \cap \binom{V'}{2}$. Given $F \subseteq E^c$, a *completion* of G, denoted by $G + F$, is the addition of the sets in F to E: $G + F = (V, E \cup F)$.

A *cycle* C in $G = (V, E)$ is an ordered list of distinct vertices of G, $C = (v_1, \ldots, v_k)$, for which $\bigcup_{i=1,\ldots,k-1}\{v_i, v_{i+1}\} \cup \{v_k, v_1\} \subseteq E$. We denote by $V(C)$ the set of vertices that appear in the cycle, and $E(C)$ the edges connected to consecutive vertices in the cycle ($E(C) := \bigcup_{i=1,\ldots,k-1}\{v_i, v_{i+1}\} \cup \{v_k, v_1\}$). The *interior* of C, $\text{int}(C)$, is the family of two-element subsets of the vertices in the cycle that do not coincide with the edges of the cycle: i.e., $\text{int}(C) = \binom{V(C)}{2} \backslash E(C)$. A cycle containing k vertices is called a *k-cycle*.

A cycle C for which the graph $G[V(C)]$ contains only those edges in the cycle is a *chordless* cycle. A graph is said to be *chordal* (or *chordless*) if the maximum size of any chordless cycle is three. A chordless cycle with k vertices is called a *k-chordless cycle*.

3 Problem Description

Let $G = (V, E)$ be a graph. A *chordal completion* of G is any subset of edges $F \subseteq E^c$ for which $G + F$ is chordal. A *minimal chordal completion* is a chordal completion F for which F' is not a chordal completion for any proper subset $F' \subset F$. A minimum chordal completion is a chordal completion of minimum cardinality. The minimum chordal completion problem (MCCP) is the problem of identifying such a subset of the complement edge set.

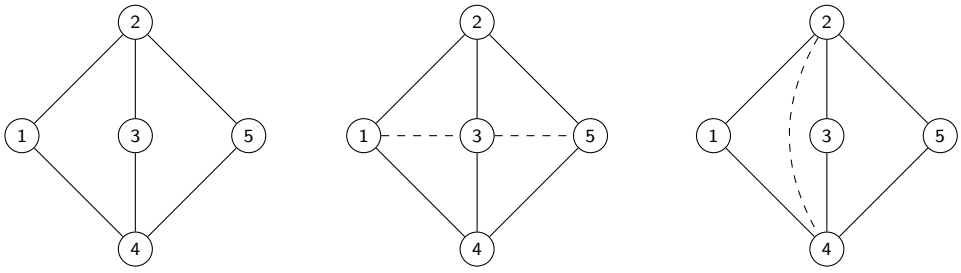
We refer to E^c as both the complement edge set and the set of candidate *fill edges*, since we think of filling G with edges in order to create chordless graphs. We will use both terms interchangeably.

Example 1. Consider the graph in Figure 1 (a). This graph has two chordless cycles, $C_1 = (1, 2, 3, 4)$ and $C_2 = (2, 3, 4, 5)$. Figure 1 (b) shows a minimal chordal completion, which adds edges $\{1, 3\}$ and $\{3, 5\}$. Removing either of these edges will result in a graph that is not chordal. Figure 1 (c) shows a smaller, minimum chordal completion consisting only of edge $\{2, 4\}$.

4 Integer programming approach

Benders decomposition is a general scheme proven to be useful for a variety of problems. Benders decomposition calls for the communication of *Benders cuts* between two models in order to communicate inferences.

In the scheme proposed in this paper, the decomposition is broken into an integer programming (IP) phase which identifies completions of G (the *master*



(a) (b) (c)

Fig. 1: (a) A graph. (b) A minimal chordal completion of the graph. (c) A minimum chordal completion of the graph.

problem) and a combinatorial optimization problem (the *subproblem*) of identifying cutting planes that will restrict the completions found by the subsequent IPs. Each time a completion is assigned and found to be inconsistent (i.e., leading to a graph which is not chordal), an inequality is produced at the end of the subproblem phase which will prohibit the IP from ever leading to this solution again. This passing of information is done iteratively until the subproblem no longer finds a Benders cut and certifies that the completion found by the master problem is not only feasible (a chordal completion), but a minimum such completion.

The master problem is the following:

$$\begin{aligned} \min \quad & \sum_{f \in E^c} z_f \\ \text{subject to} \quad & \text{[accumulated Benders cuts]} \\ & z_f \in \{0,1\}, \qquad\qquad f \in E^c \end{aligned} \tag{MP}$$

leading initially to the solution z^0 for which, $\forall f \in E^c, z^0_f = 0$, Benders cuts are yet to be generated.

For a solution z' with elements in $\{0,1\}$ for each $f \in E^c$, let $F(z') = \{f : z'_f = 1\}$. Each time a master problem is solved, the solution z^k encodes a graph $G(z^k) = (V, E \cup F(z^k))$. The subproblem has the goal of either determining that $G(z^k)$ is chordal, or producing some certificate of infeasibility which is translated into a linear inequality, restricting the master problem from ever producing this solution again (called *Benders cuts*). In the case of the MCCP, this certificate will be a k-cycle C with $k \geq 4$ (or a set of such cycles).

In a general iteration, the master problem will contain all of the elements in (MP), in addition to Benders cuts which are accumulated from earlier iterations.

The pseudocode for the algorithm is presented in Algorithm 1. We start with z^0 as defined above. This specifies that $G(z^0) = G$. Given solution z^k, the algorithm tests whether or not the graph $G(z^k)$ is chordal. If it is, the solution relates

Algorithm 1 Benders decomposition for the MCCP in graph $G = (V, E)$

1: $k \leftarrow 0$
2: let z^0 be the optimal solution to (MP), with no Benders cuts // $\forall f \in E^c, z_f^0 = 0$
3: **while** $G(z^k)$ is not chordal **do**
4: $\quad k \leftarrow k + 1$
5: $\quad$ Let $\mathcal{C}^k$ be a set of chordless cycles in $G(z^{k-1})$
6: $\quad$ **for all** $C \in \mathcal{C}^k$ **do**
7: $\quad\quad$ generate Benders cuts $B(C)$ and add them to (MP)
8: $\quad z^k \leftarrow$ optimal solution to (MP) with all accumulated Benders cuts
9: **return** $F(z^k)$

to an optimal solution and $F(z^k)$ is returned. If not, the algorithm increases the iteration count by one, identifies a set of chordless cycles $\mathcal{C}^k$ in $G(z^k)$, generates a set of Benders cuts $B(C)$ for each $C \in \mathcal{C}^k$, and adds these cuts to the master problem. The master problem is then re-solved, to find a possible solution.

Several elements of Algorithm 1 need to be specified. Namely, line 3 which determines whether or not a given graph is chordal, line 5 which finds a set of chordless cycles in a graph which is not chordal, and line 7 which generates Benders cuts based on the cycles. In general, a relaxation is also added to the master problem in order guide the initial solution. One can view the first round of Benders cuts as such a relaxation since, as explained below, the first round of cuts is based entirely on the original graph. The following sections specify these particulars.

5 Finding Chordless Cycles

Determining whether a graph is chordal or not can be accomplished in linear time in the size of the graph [30]. This can be shown to be equivalent to finding a *perfect elimination order* of a graph, and even finding all perfect elimination orders has been investigated [10].

Papers have been published which seek to identify all chordless cycles in a graph too [32,35], but the running time of these algorithms are exponential in the size of the graph. For the purpose of this paper, it is not necessary to list *all* chordless cycles; in any iteration of Algorithm 1, it is only necessary to find at least one chordless cycle, if one exists.

A simple strategy can be employed, based on searching through triples of vertices, that can be used to find a set of chordal cycles (or stopped prematurely to find a single chordal cycle) if one exists. This is presented in Algorithm 2.

The algorithm starts with no cycles in $\mathcal{C}$. For every ordered triple of vertices i, j, k for which i, j, k is an *induced path* (i.e., j is adjacent to both i and k, but i and k are not adjacent), the algorithm checks whether i and k are connected in the graph $G\left[(V \setminus N[j]) \cup \{i, k\}\right]$ induced by all vertices in G besides the closed neighborhood of j (include i and k). If so, the algorithm adds the set of vertices

$\{i, j, k\}$ together with the shortest path between i and k in $G[(V \setminus N[j]) \cup \{i, k\}]$ to $\mathcal{C}$.

Theorem 1. *Algorithm 2 returns a set $\mathcal{C}$ for which every $C \in \mathcal{C}$ is a chordless cycle in G, and is empty at the end of the execution of the algorithm if and only if G is chordal.*

Proof. Suppose $\mathcal{C} \neq \emptyset$. Let C be any cycle in $\mathcal{C}$ and i, j, k the ordered triple of vertices that produced C. C is a cycle because i is adjacent to j, j is adjacent to k, and i, k are connected in G through the path $P = (i, v_1, \ldots, v_\ell, k)$ (determined during the algorithm) that connects i and k, does not contain j, and is connected in the subgraph $G[V \setminus N[j] \cup \{i, k\}]$ of G.

Furthermore, C must be a chordless cycle. By the construction of C, j is only adjacent to i and k (with $\{i, k\} \notin P$), so that j does not participate in any chord of C. In addition, since P is the shortest path in $G[V \setminus N[j] \cup \{i, k\}]$, there cannot be any edge $\{v_a, v_b\}$, for $a \leq b - 2$ (otherwise P would not be a shortest path) in the subgraph. Therefore, C is a chordless cycle in G.

What remains to be shown is that $\mathcal{C}$ is empty if and only if G is chordal. From the previous arguments, if $\mathcal{C}$ is non-empty, then every set $C \in \mathcal{C}$ is a cycle in G. Therefore in this case G is not chordal.

Finally, suppose G is not chordal. Let $C = (v_1, \ldots, v_\ell)$ be a smallest length chordless cycle in G ($\ell \geq 4$). Consider when the algorithm examines $i = v_1, j = v_2, k = v_3$. C is a chordless cycle, therefore v_2 is only adjacent to v_1 and v_3 in G. Therefore, the path $P = (v_1, v_\ell, \ldots, v_3)$ is in $G[V \setminus N[j] \cup \{i, k\}]$. Furthermore, this must be the shortest path in this subgraph, because otherwise C would not be a smallest length chordless cycle. Therefore, C is in $\mathcal{C}$ at the end of the execution of Algorithm 2, as desired, completing the proof. □

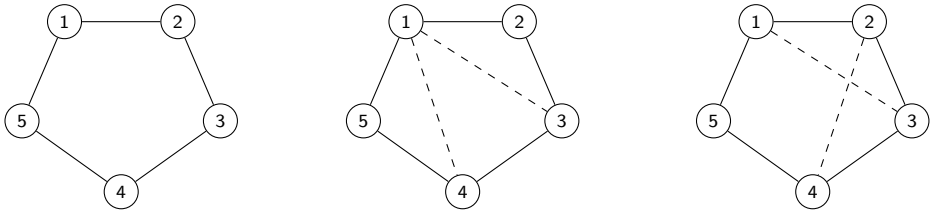
Algorithm 2 Find a set of chordless cycles $\mathcal{C}$ in $G = (V, E)$ (or return $\mathcal{C} = \emptyset$ if G is chordal)

```
1: C ← ∅
2: for all i, j, k ∈ V // all ordered triples of vertices do
3:     if {i, j}, {j, k} ∈ E and {i, k} ∉ E then
4:         if i, k are connected in G[V \ N[j] ∪ {i, k}] then
5:             P ← shortest path from i to k in G[V \ N[j] ∪ {i, k}]
6:             C ← C ∪ (P ∪ {j})
7: return C
```

6 Benders Cuts

In this section a class of Benders cuts is presented, each generated by a chordless cycle in a graph. The main idea behind the Benders cuts developed here is that if a graph G contains a k-chordless cycle C then at least $k-3$ of the edges interior to the cycle must be in any chordal completion of the graph.

Example 2. Consider the graph is Figure 2 (a). The graph is a cycle $C = \{1, 2, 3, 4, 5\}$ which is not chordal. Any chordal completion requires at least two edges. One such completion is depicted in Figure 2 (b). Note only specific subsets of size two can be added to the graph to result in a chordal completion. Consider for example the graph depicted in Figure 2 (c), where the edges $\{1, 3\}$ and $\{2, 4\}$ are added to the graph although the graph is not chordal.



(a) (b) (c)

Fig. 2: (a) A graph. (b) A chordal completion with two fill edges. (c) A completion with two edges that is not chordal.

Lemma 1. *Let G be a graph containing a chordless cycle C of length $k \geq 4$. Then at least $k - 3$ edges in int(C) are in any chordal completion of G.*

Proof. We proceed by induction on k. For $k = 4$, if fewer that $4 - 3$ edges in the interior of the chordless cycle are added to G the graph will still contain this chordless cycle and hence not be chordal.

For the base case, let $k = 5$ and $C = (1, 2, 3, 4, 5)$. Suppose we add fewer than $5 - 3 = 2$ edges to the interior of C. Adding zero edges to the interior of C leaves G not chordal. Suppose we add only one edge to the interior of C. Since the cycle is symmetric with respect to the edges interior to the cycle, suppose $\{1, 3\}$ is added. This leaves chordless cycle $(1, 3, 4, 5)$.

Fix $k > 5$. Suppose that $\forall \ell, 4 \leq \ell \leq k - 1$, if a chordless cycle C' with length ℓ appears in G then at least $\ell - 3$ edges in the interior of C' must appear in any chordless completion of G.

Let C be a chordless cycle in G of length k. Let the vertices in C be the first k integers: $C = (1, 2, \ldots, k)$. There must be at least one edge in the interior of C that appears in any chordless completion of G. Since C is symmetric with respect to the vertices, let suppose that $\{1, p\}$ is added to G.

If $p = 3$, then $C' = (1, 3, 4, \ldots, k)$ is a chordless cycle of length $k - 1$. By the inductive hypothesis, any chordal completion requires at least $(k-1)-3 = k-4$ edges in the interior of C' (which are also interior to C). Therefore any chordal completion of G requires at least $1 + k - 4 = k - 3$ edges in the interior of C.

If $p = k - 1$, the vertices can be renamed in opposite order, resulting in the same case as $p = 3$.

Let $4 \leq p \leq k-2$. This chord cuts C and creates two, separate chordless cycles: $C^1 = (1, 2, \ldots, p)$ and $C^2 = (1, p, p+1, \ldots, k)$. By the inductive hypothesis, any chordless completion will require at least $p-3$ edges in the interior of C^1 and at least $k-p+2-3 = k-p-1$ edges in the interior of C^2 (all of which are also in the interior of C). Therefore, any chordal completion requires at least $(1) + (p-3) + (k-p-1) = k-3$ edges in the interior of C, as desired. □

A *valid Benders cut* (or simply Benders cut) for a solution z^k is any inequality $az \geq b$ that satisfies the following two properties:

(B.1) z^k must violate the constraint: $az^k < b$
(B.2) Any solution z' to the master problem which generates a graph $G(z')$ which is chordal should satisfy the constraint: $az' \geq b$

Given a solution z^k in Algorithm 1 which yields graph $G(z^k)$ which is not chordal, after identifying a chordless cycle C (or set of chordless cycles) in line 5, the goal is to find a linear inequality which can be added to the master problem which prohibits z^k from appearing again and is satisfied by every solution z' which corresponds to a chordal completion $F(z^k)$. The simplest such cut, which is readily applicable in Benders schemes, is the following:

$$\sum_{f:z_f^k=1} (1 - z_f) + \sum_{f:z_f^k=0} (z_f) \geq 1 \tag{1}$$

This simple cut forbids only the given solution z^k

Example 3. Take for example z^0 with $z_f^0 = 0$ for all $f \in E^c$ for the graph in Figure 2. There are many possible Benders cuts associated with this solution. These include, for example:

$$z_{\{1,3\}} + z_{\{1,4\}} + z_{\{2,4\}} + z_{\{2,5\}} + z_{\{3,5\}} \geq 1 \tag{2}$$

(2) is the standard Benders cut (1).

The application of Benders decomposition necessitates the generation of problem specific valid Benders cuts so that the inequalities are tighter and eliminate additional infeasibility, for otherwise too many iterations are realized. The remainder of this section is devoted to proving that the following inequality, for a chordless cycle C identified in Algorithm 1, is a valid Benders cut:

$$\sum_{f \in \operatorname{int}(C)} z_f \geq (|V(C)| - 3) \cdot \left(\sum_{f \in E(C) \cap E^c} z_f - |E(C) \cap E^c| + 1 \right) \tag{3}$$

Example 4. Let G be the graph in Figure 2. Suppose the solution z^k to the master problem is $z^k_{\{1,3\}} = z^k_{\{1,4\}} = z^k_{\{2,4\}} = z^k_{\{2,5\}} = z^k_{\{3,5\}} = 0$. (3) becomes

$$z_{\{1,3\}} + z_{\{1,4\}} + z_{\{2,4\}} + z_{\{2,5\}} + z_{\{3,5\}} \geq 2 \cdot 1 = 2$$

This is a valid Benders cut. The solution violates this constraint and in any chordal completion at least two of the edges in the interior of the cycle must be present.

Example 5. Take again the graph G in Figure 2. Suppose the solution z^k to the master problem is $z^k_{\{1,3\}} = 1$ and $z^k_{\{1,4\}} = z^k_{\{2,4\}} = z^k_{\{2,5\}} = z^k_{\{3,5\}} = 0$. Let $C = (v_1, v_3, v_4, v_5)$, which is a chordless cycle in the graph $G(z^k)$. In this case,

$$\begin{aligned} V(C) &= \{1,3,4,5\} \\ E(C) &= \{\{1,3\},\{3,4\},\{4,5\},\{1,5\}\} \\ \text{int}(C) &= \{\{1,4\},\{3,5\}\} \\ E(C) \cap E^c &= \{\{1,3\}\} \end{aligned}$$

(3) now becomes

$$z_{\{1,4\}} + z_{\{3,5\}} \geq (4-3) \cdot \left(z_{\{1,3\}} - 1 + 1\right) = z_{\{1,3\}}$$

This inequality will force at least one of $z_{\{1,4\}}$ or $z_{\{3,5\}}$ to be equal to one if $z_{\{1,3\}} = 1$ in any subsequent master problem solution. This inequality is a valid Benders cut because **(B.1)** is satisfied (plugging in z^k into the converse of the inequality yields $0 < 1$) and **(B.2)** is satisfied (if a completion of G contains edge $\{1,3\}$, it *must* contain either edge $\{1,4\}$ or $\{3,5\}$, for otherwise cycle $C = (1,3,4,5)$ will be a chordless cycle in G).

Theorem 2. *Suppose C is a chordless cycle, identified in line 5, in the graph $G(z^k)$ derived from solution z^k obtained by solving the master problem in iteration k of Algorithm 1. (3) is a valid Benders cut.*

Proof. Let C be such a cycle.

(B.1): Because C is a chordless cycle, there cannot be any edges in the interior of C in $G(z^k)$ so that the left-hand side of (3) becomes $\sum_{f \in \text{int}(C)} z^k_f = 0$.

Therefore it suffices to show that the right-hand side of (3) is strictly greater than 0. C is a chordless cycle, so $|V(C)| \geq 4$ and $(|V(C)| - 3) > 0$. In addition, each edge in $E(C) \cap E^c$ must be in C, for otherwise C would not be a cycle in $G(z^k)$, so that for the variables in this set $z^k_f = 1$. Therefore,

$$\sum_{f \in E(C) \cap E^c} z^k_f - |E(C) \cap E^c| + 1 = 1,$$

and the right-hand side of (3) evaluates to $(|V(C)| - 3)$ which is greater than 0.

(B.2): Let z' be any solution that generates a graph $G(z')$ that is chordal. First note that (3) is satisfied by *any* solution (not just feasible ones) if for any $f \in E(C) \cap E^c, z'_f = 0$. This is because the left-hand side of the inequality is always greater than or equal to 0, and if $z'_f = 0$ for some such z', the right-hand side of the inequality will be less than or equal to 0. Therefore consider only those z' for which the right hand side is greater than or equal to 1 i.e. $\forall f \in E(C) \cap E^c, z'_f = 1$.

For such z', the second of the two terms multiplied on the right-hand side of (3) evaluates to 0 at z'. It therefore suffices to show that

$$\sum_{f \in \text{int}(C)} z'_f \geq |V(C)| - 3.$$

Because $\forall f \in E(C) \cap E^c, z'_f = 1$, $G(z')$ contains every edge in $E(C)$. C is a cycle in $G(z')$. Therefore, by Lemma 1, at least $|V(C)| - 3$ edges in the interior of C are in any chordal completion of $G(z')$. □

7 Minimum/minimal chordal completion algorithms

As discussed in Section 1, there is a vast literature on minimal chordal completions while the literature investigated computational approaches to the MCCP is surprisingly thin. In order to evaluate, computationally, Algorithm 1, it is necessary to compare with another MCCP algorithm. In the experimental results section, Section 8, the Benders approach is compared with a simple variant of an algorithm for the MCCP that has exponential running time [8]. This algorithm is explained below, along with a state-of-the-art heuristic.

Exact Algorithm Developing any exact algorithm designed to solve the MCCP is non-trivial. Even formulating the problem with an IP model, a standard approach to solving combinatorial optimization problems, is difficult.

The fastest (in terms of time performance guarantee) algorithm for solving the MCCP [13], to the best knowledge of the authors, has time complexity $O\left(2^{O\left(\sqrt{k}\log(k)+k^2 nm\right)}\right)$, although it is not implemented nor have computational experiments on the algorithm been reported. The algorithm branches on chordless cycles of length four, and then on *moplexes*, keeping track of certain indicators which allow pruning. The interested reader can refer to the paper in the references for an explanation of the algorithm.

In place of implementing this complex algorithm, a simpler version is presented in Algorithm 3, which will be used for computational comparison. This algorithm searches on the complement edge set. It starts with a single search tree node s. Search tree nodes have two sets associated with them: $I(s)$ is the set of complement edges included in this search node, and $O(s)$ are the set of edges to be excluded. For any search tree node, if it is chordal, the upper bound is updated and the set $I(s)$ becomes the incumbent solution. If it isn't, two new nodes are created, only if they can lead to better solutions (it is required that $|I(s)| \leq z_{ub} - 2$ because at least one more additional edge has to be included and so it can only lead to a better solution if the number of edges is at least two less than the size of the incumbent solution). Then, a cycle is identified in $G' = (V, E \cup I(s))$ and for some edge in the interior of this cycle but not in $O(s)$, a search node is created which includes the edge and a search node is created which excludes this edge. The search concludes when there are no more nodes to explore. This algorithm can have a *warm start* where any heuristic can be ran prior to the algorithm for a starting incumbent solution F. The computational experiments in Section 8 assumes a depth-first search ordering.

Heuristic Algorithms This section describes a heuristic that will be used for comparison with the Benders approach, to see the gap that results from running only a minimal chordal completion algorithm.

Algorithm 3 Branch-and-bound algorithm for the MCCP in graph $G = (V, E)$

$F \leftarrow E^c$
$z_{ub} \leftarrow |F|$
create search node s with $I(s) = O(s) = \emptyset$
$Q \leftarrow \{s\}$
while $Q \neq \emptyset$ **do**
  let s be some node in Q
  $Q \leftarrow Q \backslash \{s\}$
  if $G' = (V, E \cup I(s))$ is chordal **then**
    $F \leftarrow I(s)$
    $z_{ub} \leftarrow |F|$
  else
    if $|I(s)| \leq z_{ub} - 2$ **then**
      find chordless cycle C of length greater than 3 in G'
      let e be any edge in $\text{int}(C) \backslash O(s)$
      if e exists **then**
        create search node s' with $I(s') = I(s) \cup \{e\}, O(s') = O(s)$
        create search node s'' with $I(s'') = I(S), O(s'') = O(s) \cup \{e\}$
        $Q \leftarrow Q \cup \{s', s''\}$
return F

A classical algorithm that can be used to find small minimal chordal completions of graphs is the elimination graph model: In this algorithm, the vertices are sorted by some permutation σ. The vertices are then relabeled according to σ. Then, for $i = 1, \ldots, n$, edges are added to G in order to make the neighbors of i in $\{1, \ldots, i-1\}$ adjacent. The chordal completions of G exactly coincide with the graphs that results from this process [14].

Finding the ordering that results in the smallest set of additional edges exactly corresponds to solving the MCCP. Finding this ordering is NP-hard, and many heuristics have been developed, most popular and highly implemented being the minimum degree ordering [16] which in general finds very good, nearly optimal solutions. This algorithm will henceforth be referred to as *MDOC* (for minimum degree ordering completion).

8 Computational Results

This section reports computational results on the Benders approach to the MCCP. All algorithms are implemented in C++ and ran on a 3.20 GHz Intel(R) Core(TM) i7-3930K CPU processor with 32 GB RAM. The IP solver used is Gurobi version 5.63. All settings were set to default, except that the solver was restricted to solving on one processor by setting `Threads` to 1.

The algorithms are tested on random graphs generated according to the Erdős-Réyni model. The graphs generated have n vertices and density d, with n ranging from 15 to 35, in increments of 5, and d ranging from 0.1 to 0.9, in increments of 0.1. 10 graphs are generated per configuration.

For the remainder of this section, the means reported are geometric means with shift 1.

Comparison with Simple branch-and-bound Algorithm Figure 3 depicts the time to solve the graphs generated with $n = 15$ vertices, displaying the geometric mean, shifted by 1, of the complete solution time for four different algorithms. `BandB(no warm start)` represents the simple branch-and-bound algorithm in Algorithm 3. `BandB(no warm start)` is the same algorithm, but initialized with a heuristic solution for bounding purposes, provided by MDOC. `Benders(single cycle)` represents Algorithm 1, where only one cycle is generated each time the subproblem is solved (Algorithm 2 is stopped once the first chordless cycle is identified), and `Benders(multi cycle)` is the same algorithm but with Algorithm 2 ran until completion.

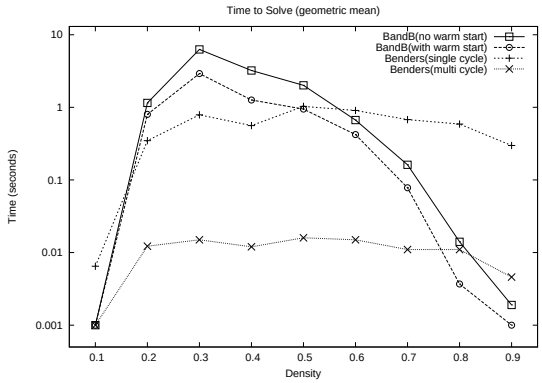


Fig. 3: Time to solve graphs with $n = 15$ vertices

As can be seen in Figure 3, the Benders approach outperforms the simple branch-and-bound algorithm except on those instances that are solved very quickly. Within 10,000,000 search nodes not all instances with $n = 20$ and above are solved by the simple branch-and-bound algorithm and so the plot is only generated for $n = 15$.

Even more elucidating of the difference in computational time is the number of instances that are solved within 30 seconds, depicted in Figure 4. For the same algorithms as above, this plot shows that within 30 seconds each of the instances with $n = 20$ are solved by `Benders(multi cycle)` (the maximum time on this set of instances for this technique is 1.13 seconds with mean of 0.053 seconds). In the alternative methods, there are many instances which remain unsolved in this time horizon, and several remaining unsolved even after 1800 seconds.

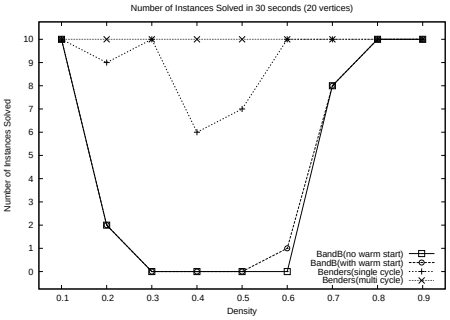


Fig. 4: Number of solved instances

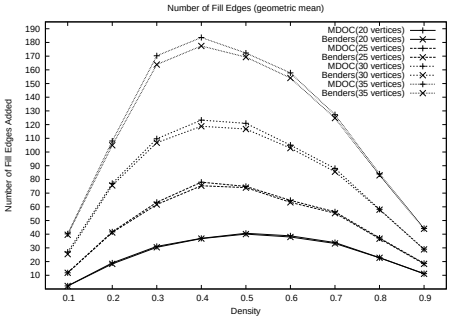


Fig. 5: Size of chordal completions

Improvement over MDOC The heuristic, MDOC, works surprisingly well in practice and hence is heavily implemented in practice. In this section the difference between the solution provided by MDOC and the Benders approach (`Benders(multi cycle)`) are provided in order to evaluate the importance of searching for optimal solution, compared with a standard, simple, and powerful heuristic.

In general, as reported in the literature, the solutions outputted by MDOC are of high quality. Figure 5 shows the comparison between the optimal solutions found by the Benders approach with the solutions obtained by MCOD for $n = 20, 25, 30, 35$. As can be seen in this figure, the gap between the heuristic solution and the optimal solution is not too substantial, however the gap grows as the graphs increase in size.

To better see the differences in the chordal completions, consider Figure 6, where plots are generated for this data, for $n = 20, 25, 30, 35$, individually. In addition to the mean, the minimum and maximum difference per configuration is depicted. This plot more readily shows that as n grows the gap between the heuristic solution and the optimal solution grows.

This of course comes at the expense of extra computation time. The time necessary to run the heuristic is a fraction of a second, compared to the time to run the Benders algorithm, reported in Table 1, although solutions are provided with optimality guarantees. This table reports the numerical values depicted in Figure 6 along with the times to solve the MCCP using the Benders approach described in this paper. n and d are the number of vertices and density in the random graphs tested (10 instances per row in the table). `MinFillDiff`, `MeanFillDiff`, and `MaxFillDiff` report the minimum, mean, and maximum difference in the number of edges in the chordal completions calculated using MDOC and the Benders approach, respectively. `MinBendersTime`, `MeanBendersTime`, and `MaxBendersTime` report the minimum, mean, and maximum time to solve the instances using the Benders approach, respectively.

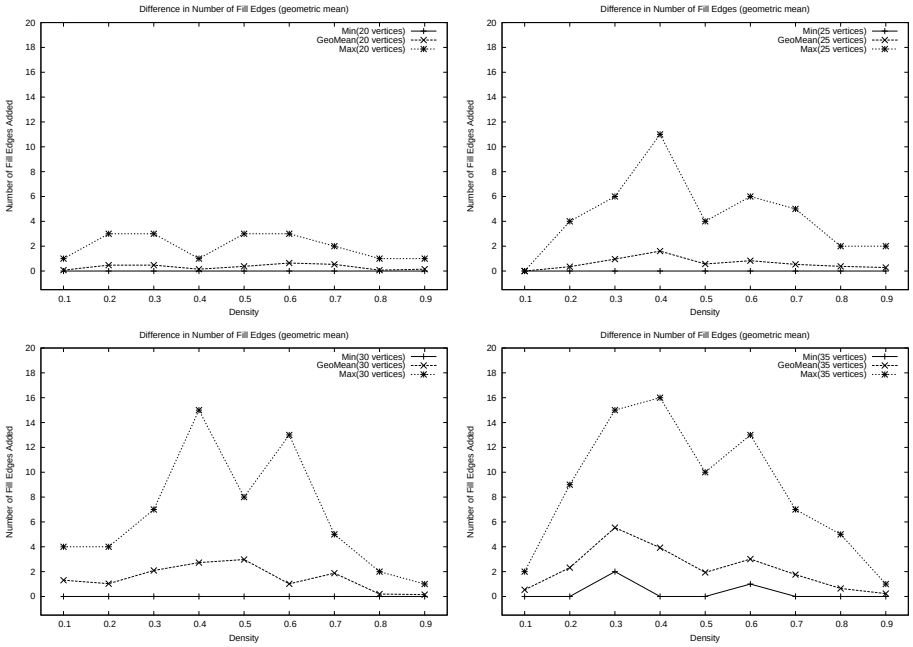


Fig. 6: Difference in number of fill edges added, $n = 20, 25, 30, 35$

n	d	MinFillDiff	MeanFillDiff	MaxFillDiff	MinBendersTime	MeanBendersTime	MaxBendersTime
20	0.1	0	0.07	1	0.00	0.00	0.01
	0.2	0	0.47	3	0.02	0.04	0.13
	0.3	0	0.47	3	0.02	0.04	0.07
	0.4	0	0.15	1	0.02	0.15	1.13
	0.5	0	0.37	3	0.03	0.11	0.37
	0.6	0	0.64	3	0.02	0.05	0.10
	0.7	0	0.53	2	0.03	0.05	0.10
	0.8	0	0.07	1	0.02	0.03	0.04
	0.9	0	0.15	1	0.01	0.01	0.02
25	0.1	0	0.00	0	0.01	0.02	0.04
	0.2	0	0.35	4	0.03	0.87	24.04
	0.3	0	0.96	6	0.08	1.00	5.28
	0.4	0	1.61	11	0.17	1.05	2.72
	0.5	0	0.57	4	0.10	0.42	1.83
	0.6	0	0.83	6	0.05	0.11	0.33
	0.7	0	0.53	5	0.08	0.17	0.66
	0.8	0	0.37	2	0.04	0.06	0.12
	0.9	0	0.28	2	0.02	0.03	0.04
30	0.1	0	1.31	4	0.02	0.87	5.36
	0.2	0	1.03	4	0.28	8.05	181.68
	0.3	0	2.09	7	2.85	15.44	371.47
	0.4	0	2.73	15	0.23	6.39	38.70
	0.5	0	2.98	8	0.24	4.91	194.28
	0.6	0	1.02	13	0.24	0.91	2.17
	0.7	0	1.88	5	0.13	0.82	2.60
	0.8	0	0.20	2	0.08	0.23	0.75
	0.9	0	0.15	1	0.03	0.05	0.09
35	0.1	0	0.53	2	0.04	2.90	70.83
	0.2	0	2.32	9	1.31	86.49	1822.97
	0.3	2	5.54	15	2.92	635.38	11782.80
	0.4	0	3.93	16	4.51	91.90	2394.53
	0.5	0	1.94	10	0.75	17.24	624.04
	0.6	1	3.01	13	3.73	15.03	100.23
	0.7	0	1.76	7	0.52	5.39	19.57
	0.8	0	0.64	5	0.24	0.57	1.15
	0.9	0	0.23	1	0.07	0.10	0.19

Table 1: Difference in number of edges resulting from chordal completions using Benders versus MDOC ; Benders time

Instance	$\|V\|$	$\|E\|$	BendersTime	BendersValue	MDOCValue
1-FullIns_3	30	100	5.85	80	80
1-FullIns_4	93	593	-	623*	839
2-FullIns_3	52	201	-	206*	273
2-Insertions_3	37	72	-	68*	103
3-FullIns_3	80	346	-	379*	661
3-Insertions_3	56	110	-	102*	198
4-Insertions_3	79	156	-	148*	331
david	87	406	1.99	64	66
mug100_1	100	166	0.56	64	91
mug100_25	100	166	0.66	64	93
mug88_1	88	146	0.37	56	82
mug88_25	88	146	0.52	56	84
myciel3	11	20	0	10	10
myciel4	23	71	0.04	46	46
myciel5	47	236	35.71	196	197
myciel6	95	755	-	713*	753
queen10_10	100	1470	-	2045*	2671
queen5_5	25	160	262.57	93	94
queen6_6	36	290	-	218*	244
queen7_7	49	460	-	402*	444
queen8_12	96	1368	-	1863*	2401
queen8_8	64	728	-	772*	970
queen9_9	81	1056	-	1301*	1664

Table 2: Chordal completion sizes on benchmark graphs ; * indicates that the value is a bound due to the Benders approach timing out in one hour.

Benchmark Graphs Table 2 reports results on benchmark graphs, appearing frequently in the literature in papers which test graph coloring algorithms and other problems as well [28]. The algorithm was tested only on those instances with 100 or fewer vertices that are connected.

The results indicate that there can be a significant gap between what the state-of-the-art minimal chordal completion algorithm returns and the optimal value of the MCCP. The table reports `Instance` (the name of the instance) the number of vertices and edges in the graph, followed by `BendersTime` (the amount of time, in seconds, to solve the instance), `BendersValue` (the optimal solution if the instance was solved within an hour and lower bound otherwise and indicated by an asterisk), and finally `MDOCValue` (the size of the chordal completion obtained by MDOC). Of particular note are the `mug` instances for which the gap between the heuristic solution value and the optimal value can be very substantial.

9 Conclusion

In conclusion, this paper introduces a Benders approach to the minimum chordal completion problem. Computational results indicate that the approach is promising as the algorithm significantly outperforms a simple branch-and-bound algorithm for the problem. In addition the paper reports that the gap between the value obtained using a state-of-the-art heuristic and the formally unknown optimal solutions to random graphs and benchmark instances can be significant.

References

1. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. J. ACM 30(3), 479–513 (Jul 1983), http://doi.acm.org/10.1145/2402.322389
2. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4(1), 238–252 (1962), http://dx.doi.org/10.1007/BF01386316
3. Berry, A., Bordat, J.P., Heggernes, P., Simonet, G., Villanger, Y.: A wide-range algorithm for minimal triangulation from an arbitrary ordering. Journal of Algorithms 58(1), 33 – 66 (2006), http://www.sciencedirect.com/science/article/pii/S0196677404001142
4. Berry, A., Heggernes, P., Simonet, G.: The minimum degree heuristic and the minimal triangulation process. In: Bodlaender, H. (ed.) Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 2880, pp. 58–70. Springer Berlin Heidelberg (2003), http://dx.doi.org/10.1007/978-3-540-39890-5_6
5. Blair, J.R., Heggernes, P., Telle, J.A.: A practical algorithm for making filled graphs minimal. Theoretical Computer Science 250(12), 125 – 141 (2001), http://www.sciencedirect.com/science/article/pii/S0304397599001267
6. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernetica 11, 1–23 (1993)
7. Bodlaender, H.L., Kloks, T., Kratsch, D., Mueller, H.: Treewidth and minimum fill-in on d-trapezoid graphs (1998)
8. Bodlaender, H., Heggernes, P., Villanger, Y.: Faster parameterized algorithms for minimum fill-in. In: Hong, S.H., Nagamochi, H., Fukunaga, T. (eds.) Algorithms and Computation, Lecture Notes in Computer Science, vol. 5369, pp. 282–293. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-92182-0_27
9. Broersma, H., Dahlhaus, E., Kloks, T.: Algorithms for the treewidth and minimum fill-in of hhd-free graphs. In: Mhring, R. (ed.) Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 1335, pp. 109–117. Springer Berlin Heidelberg (1997), http://dx.doi.org/10.1007/BFb0024492
10. Chandran, L.S., Ibarra, L., Ruskey, F., Sawada, J.: Generating and characterizing the perfect elimination orderings of a chordal graph. Theor. Comput. Sci. 307(2), 303–317 (Oct 2003), http://dx.doi.org/10.1016/S0304-3975(03)00221-4
11. Chang, M.S.: Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In: Asano, T., Igarashi, Y., Nagamochi, H., Miyano, S., Suri, S. (eds.) Algorithms and Computation, Lecture Notes in Computer Science, vol. 1178, pp. 146–155. Springer Berlin Heidelberg (1996), http://dx.doi.org/10.1007/BFb0009490
12. Chung, F., Mumford, D.: Chordal completions of planar graphs. Journal of Combinatorial Theory, Series B 62(1), 96 – 106 (1994), http://www.sciencedirect.com/science/article/pii/S0095895684710562
13. Fomin, F.V., Villanger, Y.: Subexponential parameterized algorithm for minimum fill-in. In: Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1737–1746. SODA '12, SIAM (2012), http://dl.acm.org/citation.cfm?id=2095116.2095254
14. Fulkerson, D.R., Gross, O.A.: Incidence matrices and interval graphs. Pacific Journal of Mathematics 15(3), 835–855 (1965), http://projecteuclid.org/euclid.pjm/1102995572

15. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
16. George, A., Liu, W.H.: The evolution of the minimum degree ordering algorithm. SIAM Rev. 31(1), 1–19 (Mar 1989), http://dx.doi.org/10.1137/1031001
17. Golumbic, M., Kaplan, H., Shamir, R.: Graph sandwich problems. Journal of Algorithms 19(3), 449 – 473 (1995), http://www.sciencedirect.com/science/article/pii/S0196677485710474
18. Grone, R., Johnson, C.R., S, E.M., Wolkowicz, H.: Positive definite completions of partial hermitian matrices. Linear Algebra and its Applications 58(0), 109 – 124 (1984), http://www.sciencedirect.com/science/article/pii/0024379584902076
19. Heggernes, P.: Minimal triangulations of graphs: A survey. Discrete Mathematics 306(3), 297 – 317 (2006), http://www.sciencedirect.com/science/article/pii/S0012365X05006060, minimal Separation and Minimal Triangulation
20. Heggernes, P., Eisenstat, S.C., Kumfert, G., Pothen, A.: The computational complexity of the minimum degree algorithm (2001)
21. Hooker, J., Ottosson, G.: Logic-based benders decomposition. Mathematical Programming 96(1), 33–60 (2003), http://dx.doi.org/10.1007/s10107-003-0375-9
22. Kaplan, H., Shamir, R., Tarjan, R.E.: Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping (extended abstract). SIAM J. Comput 28, 780–791 (1994)
23. Kim, S., Kojima, M., Mevissen, M., Yamashita, M.: Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. Mathematical Programming 129(1), 33–68 (2011), http://dx.doi.org/10.1007/s10107-010-0402-6
24. Kloks, T., Kratsch, D., Wong, C.: Minimum fill-in on circle and circular-arc graphs. Journal of Algorithms 28(2), 272 – 289 (1998), http://www.sciencedirect.com/science/article/pii/S0196677498909361
25. Lekkeikerker, C., Boland, J.: Representation of a finite graph by a set of intervals on the real line. Fundamenta Mathematicae 51, 45–64 (1962)
26. Mezzini, M., Moscarini, M.: Simple algorithms for minimal triangulation of a graph and backward selection of a decomposable markov network. Theoretical Computer Science 411(79), 958 – 966 (2010), http://www.sciencedirect.com/science/article/pii/S030439750900735X
27. Nakata, K., Fujisawa, K., Fukuda, M., Kojima, M., Murota, K.: Exploiting sparsity in semidefinite programming via matrix completion ii: implementation and numerical results. Mathematical Programming 95(2), 303–327 (2003), http://dx.doi.org/10.1007/s10107-002-0351-9
28. Nguyen, T.H., Bui, T.: Graph coloring benchmark instances. http://www.cs.hbg.psu.edu/txn131/graphcoloring.html, accessed: 2014-07-14
29. Peyton, B.W.: Minimal orderings revisited. SIAM J. Matrix Anal. Appl. 23(1), 271–294 (Jan 2001), http://dx.doi.org/10.1137/S089547989936443X
30. Rose, D., Tarjan, R., Lueker, G.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal on Computing 5(2), 266–283 (1976), http://dx.doi.org/10.1137/0205021
31. Rose, D.J.: A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In: Graph Theory and Computing, pp. 183 – 217. Academic Press (1972), http://www.sciencedirect.com/science/article/pii/B9781483231877500180

32. Sokhn, N., Baltensperger, R., Bersier, L.F., Hennebert, J., Ultes-Nitsche, U.: Identification of chordless cycles in ecological networks. In: Glass, K., Colbaugh, R., Ormerod, P., Tsao, J. (eds.) Complex Sciences, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 126, pp. 316–324. Springer International Publishing (2013), http://dx.doi.org/10.1007/978-3-319-03473-7_28
33. Spinrad, J., Brandstdt, A., Stewart, L.: Bipartite permutation graphs. Discrete Applied Mathematics 18(3), 279 – 292 (1987), http://www.sciencedirect.com/science/article/pii/S0166218X87800033
34. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM J. Comput. 13(3), 566–579 (Jul 1984), http://dx.doi.org/10.1137/0213035
35. Uno, T., Satoh, H.: An efficient algorithm for enumerating chordless cycles and chordless paths. CoRR abs/1404.7610 (2014), http://arxiv.org/abs/1404.7610
36. Yannakakis, M.: Computing the minimum fill-in is np-complete. SIAM Journal on Algebraic Discrete Methods 2(1), 77–79 (1981)