A Decomposition Approach to Solve the Selective Graph Coloring Problem in Some Perfect Graph Families *

Oylum Şeker Tınaz Ekim Z. Caner Taşkın

July 16, 2018

Abstract

Graph coloring is the problem of assigning a minimum number of colors to all vertices of a graph such that no two adjacent vertices receive the same color. The Selective Graph Coloring Problem is a generalization of the standard graph coloring problem; given a graph with a partition of its vertex set into clusters, the objective is to choose exactly one vertex per cluster so that, among all possible selections, the number of colors necessary to color the vertices in the selection is minimum. This study focuses on a decomposition based exact solution framework for selective coloring in some perfect graph families: in particular, permutation, generalized split, and chordal graphs where the selective coloring problem is known to be NP-hard. Our method combines integer programming techniques and combinatorial algorithms for the graph classes of interest. We test our method on graphs with different sizes and densities, present computational results and compare them with solving an integer programming formulation of the problem by CPLEX, and a state-of-the art algorithm from the literature. Our computational experiments indicate that our decomposition approach significantly improves solution performance in low-density graphs, and regardless of edge-density in the class of chordal graphs.

Keywords: selective graph coloring; partition coloring; decomposition algorithm; integer programming; permutation graphs; generalized split graphs; chordal graphs

1 Introduction

1.1 Motivation and Literature Survey

Graph coloring is the problem of assigning a minimum number of colors to all vertices of a graph such that no two adjacent vertices, i.e., vertices that are linked by an edge,

*This study is supported by Boğaziçi University Research Fund (grant 11765); and T. Ekim is supported by Turkish Academy of Sciences GEBIP award and Visiting Fulbright Scholar Program.

receive the same color. The problem arises in a variety of areas including scheduling [18], register allocation used in compiler optimization [2], sudoku puzzles [16], and many more. In the most general terms, a problem relevant to graph coloring is comprised of entities, and incompatibilities among them. Entities are represented by vertices, and incompatibilities between pairs of entities by edges, so that no two entities posing incompatibility can result in a conflict when colored.

To embody the formal definition of the problem, a scheduling problem in its simplest form can be considered. Assume a number of jobs, some of which share common resources, are to be assigned to certain time slots. Denoting jobs with vertices, and pairs of jobs that share a common resource with edges, the problem can be expressed in the domain of graphs. When the vertices of such a graph are colored, each color corresponds to a distinct time slot and hence the minimum number of colors needed to color the vertices of such a graph gives the least number of slots necessary to finish all the jobs [18].

The selective graph coloring problem, Sel-Col, is a generalization of the classical graph coloring problem. Given a graph and a partition of its vertex set into clusters, Sel-Col aims to choose one vertex per cluster so that, among all possible selections, the number of colors necessary to color the vertices in the selection is minimum. In a graph where each cluster consists of a single vertex, Sel-Col becomes equivalent to the classical graph coloring problem [5].

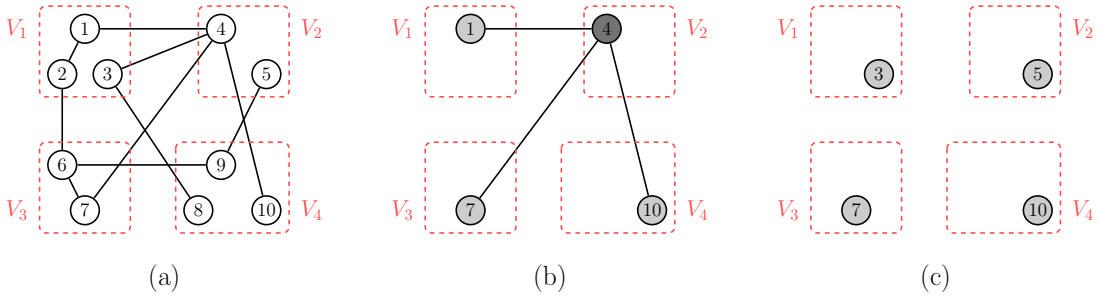

Figure 1: (a) A graph G with a partition of its vertex set shown in dashed rectangles, (b) a feasible selection in G with an optimal coloring of it, (c) an optimal selection in G with an optimal coloring of it.

Sel-Col, or *partition coloring* as it is alternatively called in the literature, is motivated by the wavelength routing and assignment problem, and was introduced in [17]. There, a telecommunication network, which is a collection of terminal nodes linked by optical fibers capable of carrying a certain number of wavelengths, and a set of source-destination node pairs are given. The problem is concerned with finding the minimum number of distinct wavelengths to assign to each route, i.e., paths connecting the given source-destination pairs, such that no two routes that have a common link are assigned the same wavelength. In this setting, the set of all possible routes and pairs of routes possessing a common link correspond to vertices and edges respectively, and groups of routes connecting a given source-destination pair constitute the clusters in the host graph. Then, selection of one vertex per cluster achieves the goal of finding a route between each pair of terminals, and coloring of the selection delivers a proper wavelength to each route.

In the standard graph coloring problem, assignments (colors) on the entities (vertices) are done without any regard to alternative choices for them. However, as the example applications reveal, there are cases where entities have their own set of feasible options (clusters) and thus should be allocated one among those alternatives. In such cases, where the basic graph coloring framework fails to suffice, Sel-Col bridges the gap by offering the required flexibility.

Sel-Col has a wide range of applications, as the classical graph coloring problem does. The authors in [4] elaborate on the selective graph coloring problem by reviewing some models from the literature and proposing some new ones from various contexts including wavelength assignment (which is originally introduced in [17]), frequency assignment, various types of scheduling problems, the travelling salesman problem with multiple stacks (introduced in [21]), and more. For each domain of application, they discuss the related model and the complexity of Sel-Col in classes of graphs pointed out by the model. Also, analogies between models as well as new solution approaches and open research directions are highlighted throughout the paper.

The selective graph coloring problem is known to be NP-hard, and remains so in many special classes of graphs [5, 17]. The work by Li and Simha [17] and that by Noronha and Ribeiro [20] focus on partition coloring in the context of wavelength routing and assignment, and offer heuristic methods to solve the problem.

To the best of our knowledge, there are three studies that primarily focus on exact solution methods for Sel-Col. In the first one [8], Frota *et al.* present a branch-and-cut algorithm for partition coloring problem. Their integer programming formulation picks one vertex to be the representative of vertices having the same color, instead of coloring all vertices in the selection. After some preprocessing operations to reduce the graph size, they implement a branch-and-cut algorithm with a branching rule custom-tailored for the partition coloring problem. The rule is a modification of the classical one used in [19] that branches on two non-adjacent vertices for the graph coloring problem. Also, a tabu search procedure is used to obtain an upper bound on each node of the branch-and-cut tree. Another study by Hoshino *et al.* [14] proposes a new integer programming model and a branch-and-price algorithm to solve the partition coloring problem. In all instance classes tested, their algorithm is said to demonstrate superior performance compared to the one proposed in [8]. Finally, a very recent study by Furini *et al.* [10] proposes a new formulation for Sel-Col with an exponential number of variables and designs a branch-and-price algorithm to solve it, where the pricing phase is based on a unique pricing problem, as opposed to the work by Hoshino *et al.*, which requires one to solve several pricing problems. Computational results indicate that the proposed branch-and-price framework improves on the previous state-of-the-art exact approaches from the literature and so we compare with this branch-and-price implementation in our experiments.

1.2 Hardness of Sel-Col and Our Contribution

As mentioned in [4], the difficulty of Sel-Col is two-fold: it may be due to the existence of exponentially many selections and/or to the hardness of optimally coloring the graph induced by the selection, even if the selection yielding the optimal solution

can be found trivially (for instance because there is only one selection as in the case of classical graph coloring problem). It is well known that although the classical graph coloring problem is NP-hard even in several restricted cases [11], it can be solved in polynomial time when restricted to perfect graphs and efficient combinatorial algorithms exist for many of its subclasses [13]. Nevertheless, Sel-Col, being at least as difficult as the classical coloring problem from a computational point of view remains NP-hard in all these special cases. In particular, Sel-Col is NP-hard in permutation graphs where the multiple stacks travelling salesman problem is modeled as Sel-Col, in split graphs, and in interval graphs where several scheduling and timetabling applications are modeled using Sel-Col [4]. Since chordal graphs and generalized split graphs contain the class of split graphs, the problem is NP-hard for chordal graphs and generalized split graphs. As in these examples, in many real life problems, the application domain yields host graphs that admit characteristics of certain graph families. The development of exact solution procedures for those special graph families where sel-col remains NP-hard is explicitly emphasized as an open question in [4].

In this paper, we focus on a decomposition based exact solution framework for sel-col in certain perfect graph families. The decomposition approach utilized in this study suggests solving sel-col by dealing with the selection and coloring tasks separately, which naturally inclines us to be interested in graph classes where the classical coloring problem can be solved efficiently, in order to facilitate the solution procedure. In line with this idea and having in mind the aforementioned hardness results, we conduct extensive computational studies on permutation graphs, generalized split graphs (a subclass of perfect graphs containing split graphs) and chordal graphs (a subclass of perfect graphs containing interval graphs). The computational experiments that we have conducted on a large set of randomly generated graphs from 100 to 1000 vertices with varying edge densities indicate that our decomposition method significantly improves solvability of the problem for low densities (see Section 3.2). For permutation graphs, our method outperforms direct solution of the integer programming formulation of the problem and the branch-and-price algorithm by Furini et al. [10] at low densities, both in terms of the amount of time spent and optimality gap percentages. In the case of generalized split graphs, the decomposition method yields better performance in terms of average optimality gaps and average times for low-density graphs having small clusters. The performance of our algorithm manifests itself most noticeably in the class of chordal graphs; even graphs on 1000 vertices can be optimally solved in under one second.

The remainder of this paper is organized as follows. Section 1.3 gives some preliminary graph-theoretic definitions and information that relate to Sel-Col in order to lay the foundation for forthcoming sections. In Section 2, we first present an integer programming formulation for Sel-Col, then describe our decomposition procedure and give details about the way it is applied in each one of the three graph classes we consider. In Section 3, we explain how the test instances are generated and present computational results for the decomposition approach in comparison to those of the integer programming formulation and a state-of-the-art algorithm from the literature [10]. Finally, Section 4 concludes the article with a brief summary and presents possible future research directions.

1.3 Definitions

A *graph* is an ordered pair $G = (V, E)$ with V being the set of *vertices* (or *nodes*) and E being the set of *edges*, which are pairs of elements of V. It provides a representation of a set of objects and their interrelations, where objects refer to vertices, and the predefined relations between pairs of objects correspond to edges in the graph.

Two vertices in a graph are called *adjacent* if they are connected by an edge. A vertex u is called a *neighbour* of another vertex v if there exists an edge $\{u, v\}$. The neighbourhood of a vertex v is the set of all vertices that are adjacent to it, and is denoted by $N(v)$.

The *complement* of a graph $G = (V, E)$, denoted as $\bar{G}$, is a graph on the same vertex set V and such that two distinct vertices of $\bar{G}$ are adjacent if and only if they are not adjacent in G. An *induced subgraph* is a graph formed by a subset $V^{'}$ of $V(G)$ and all edges connecting the pairs of vertices in $V^{'}$. For a graph $G = (V, E)$ and $V' \subseteq V$, the subgraph induced by V' will be shown as $G[V']$.

A *clique* in a graph is a subset of vertices such that every distinct pair of vertices in the subset is adjacent. In a graph, a given clique is *maximal* if its size cannot be extended with inclusion of some other vertex; in other words, if it is not a proper subset of another clique. The *clique number* of a graph G is the size of a largest clique in G and is denoted by $\omega(G)$. A *stable set*, or equivalently an *independent set*, is a set of vertices in a given graph in which no two vertices are adjacent.

A (simple) *cycle* is comprised of a sequence of consecutively adjacent vertices that starts and ends at the same vertex with no repetitions of vertices and edges. A *chord* is an edge linking two non-consecutive vertices in a cycle.

A *tree* is a graph in which any two vertices are connected by exactly one path, where a *path* is a sequence of edges that connects a series of vertices in a graph.

A coloring of a graph using at most k colors is called a (proper) *k-coloring*. A graph is called *k-colorable* if its vertices can be assigned a k-coloring. The *chromatic number* of a graph G, denoted by $\chi(G)$, is the minimum number of colors necessary to color all vertices of the graph. Note that a graph G is k-colorable for all $k \geq \chi(G)$.

Given a graph $G = (V, E)$ with a partition of its vertex set into P clusters $\mathcal{V} = \{V_1, ..., V_P\}$, a *selection* is a subset of vertices of G that contains exactly one vertex from each cluster in the partition; i.e., $V' \subseteq V$ such that $|V' \cap V_p| = 1$ for all $p \in \{1, \ldots, P\}$. A *selective k-coloring* of G is defined by a selection V' and a k-coloring of $G[V']$. The smallest integer k for which G admits a selective k-coloring is called the *selective chromatic number* of G and is denoted by $\chi_{SEL}(G, \mathcal{V})$ [4]. A *selective clique* of G is a clique in the graph induced by a selection. A maximal clique in the graph induced by a selection is called a *maximal selective clique*, and a maximal selective clique of maximum size is called a *maximum selective clique* of G.

A graph G is *perfect* if every induced subgraph $G' \subseteq G$ satisfies $\chi(G') = \omega(G')$. Permutation graphs, generalized split graphs and chordal graphs are subclasses of perfect graphs [3,13] that we consider in this study. They will be formally introduced in Sections 2.2.1, 2.2.2 and 2.2.3 respectively.

2 SEL-COL in Perfect Graph Families

In this section, we first give an integer programming formulation to solve SEL-COL in general, then introduce our decomposition framework and give details about the application of this method in the case of certain perfect graph families that we focus on. We assume that we are given a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ and a partition of its vertex set into P clusters $V_1, ..., V_P$; i.e., $\bigcup_{p=1}^{P} V_p = V$ where $V_p \subset V$, $V_p \cap V_q = \varnothing$, $V_p \neq \varnothing \ \ \forall\, p, q \in \{1, ..., P\}$ and $p \neq q$.

2.1 Integer Programming Formulation

An integer programming (IP) formulation to solve SEL-COL is expressed by Model 1.

Model 1:

$$\min \quad \sum_{k=1}^{P} y_k \tag{1}$$

s.t.

$$x_{ik} \leq y_k \quad \forall (i,k) \in V \times \{1, ..., P\} \tag{2}$$

$$x_{ik} + x_{jk} \leq 1 \quad \forall\ (i,j) \in E,\ k \in \{1, ..., P\} \tag{3}$$

$$\sum_{i \in V_p} \sum_{k=1}^{P} x_{ik} = 1 \quad \forall p \in \{1, ..., P\} \tag{4}$$

$$y_k \in \{0, 1\} \quad \forall k \in \{1, ..., P\} \tag{5}$$

$$x_{ik} \in \{0, 1\} \quad \forall (i,k) \in V \times \{1, ..., P\}, \tag{6}$$

where x_{ik} is a binary variable taking value 1 if vertex i is colored with color k and 0 otherwise, y_k is another binary variable having value 1 if color k is used and 0 otherwise.

Constraint set (2) forces y_k to be 1 if color k is used by some vertex, and (3) ensures that two adjacent vertices do not use the same color. Finally, the requirement that exactly one vertex must be chosen and colored from each cluster is achieved via constraint set (4). The fact that only one vertex must be selected from each cluster makes the number of clusters P in the partition a natural upper bound on the number of necessary colors: in the worst case the vertices in a selection would form a clique and all P colors would be needed.

One important issue about this formulation is that, due to the way x_{ik} variables are defined, a feasible solution can be equivalently expressed in a number of alternative ways by simply reindexing the colors used in that solution. Specifically, a feasible n-coloring of a selection will have $n!$ equivalent representations obtained by merely permuting the indices of those n colors. Moreover, any n-subset of P colors can equally be used in an n-coloring, and there are $\binom{P}{n}$ different ways to select them. Then, each feasible solution using n of the P available colors has $\frac{P!}{(P-n)!}$ equivalent alternative solutions. This symmetry inherent in the formulation, or existence of

"clones" in feasible space, poses additional burdens on branch-and-bound/cut procedures by causing them to explore and fathom these isomorphic copies of solutions during the search [24]. Therefore, we add the constraint set (7) to Model 1 in order to reduce symmetry.

$$y_k \geq y_{k-1} \quad \forall k \in \{2, ..., P\} \tag{7}$$

Constraints (7) place a hierarchy on available colors (similar to symmetry breaking constraints in [24]) by enforcing the program to use the colors in increasing order of their indices; that is, by only allowing the lowest numbered n colors to be used in an n-colorable feasible solution. This way, the subset of clone solutions that would arise from selecting alternative combinations of the P available colors is eliminated.

Note that the y-variables can be relaxed as continuous since constraints (2) ensure that the y-variables will take on binary values in an optimal solution. However, Model 1 still contains $O(|V| \times P)$ binary variables and $O(|E| \times P)$ constraints. Its solution time increases exponentially with the input size, and becomes intractable in relatively small graphs. The next subsection offers a decomposition framework as a promising alternative to the IP formulation.

2.2 Decomposition Approach

Let t denote an estimate of the number of colors needed. SEL-COL can alternatively be formulated as follows:

Model 2:

$$\min \ t \tag{8}$$

s.t.

$$\sum_{i \in V_p} x_i = 1 \quad \forall p \in \{1, ..., P\} \tag{9}$$

$$t \geq \chi(G[x]) \tag{10}$$

$$t \geq 0 \tag{11}$$

$$x_i \in \{0, 1\} \quad \forall i \in V, \tag{12}$$

where x_i is a binary variable taking value 1 if vertex i is selected and 0 otherwise, and $G[x]$ is the graph induced by the selection given by the variable vector $x = (x_1, \ldots, x_n)$.

The nonnegative variable t is forced to be no smaller than the chromatic number of an optimal selection by constraint set (10), which means that the minimum value it takes is exactly the selective chromatic number $\chi_{SEL}(G, \mathcal{V})$ of the input graph G. The constraint set (9) ensures that exactly one vertex is chosen from each of the P clusters. The model is not well defined in its current form because of the $\chi(.)$ operator in constraint set (10). In order to re-express (10) in a linear form, we need to embed a set of linear inequalities to carry out the coloring task. This inclines us to separate the problem into two parts, where the selection task is handled in one and the coloring of the given selection in the other.

Let us first relax constraint set (10). Removal of (10) from Model 2 leaves an integer programming formulation, which yields a feasible vertex selection for G. This relaxed version of the model constitutes the initial master problem. At each step, the master problem is solved to optimality and the vertex selection found by it is passed to a subproblem, which computes the minimum number of colors needed to color that selection. If the optimal objective value of the subproblem is higher than the optimal t-value of the master, it means that the master problem has under-estimated the chromatic number of the current vertex selection. In that case, we add a constraint to the master that forces t to be at least as large as this chromatic number, unless the particular selection is altered. One such constraint can be expressed as follows:

$$t \geq \chi(G[x^{(j)}]) - \sum_{\{i \in V | x_i^{(j)} = 1\}} (1 - x_i), \tag{13}$$

where $G[x^{(j)}]$ denotes the graph induced by the selection found at iteration j given by the variable vector $x^{(j)}$, and $\chi(G[x^{(j)}])$ the minimum number of colors needed to color it.

The rightmost term in inequality (13) is equal to zero only when $x = x^{(j)}$; i.e., when exactly the same vertices as in $x^{(j)}$ are picked, and it increases by one with each vertex removed from the selection given by $x^{(j)}$. This constraint depends on the fact that chromatic number of a graph induced by a selection can decrease by at most one for each vertex switch. Therefore, it decreases the lower bound by one for each vertex we replace.

A natural lower bound on the chromatic number $\chi(G)$ of a graph G is the size $\omega(G)$ of a maximum clique in it. If G contains a clique of size k, we need k colors to color that clique, because each vertex is adjacent to all other vertices and therefore receives a distinct color. Since this argument is valid for any clique in the graph, we need at least $\omega(G)$ colors to color G. We can translate this relationship into an inequality as given in (14), and use it as an additional cut within our decomposition procedure.

$$t \geq \sum_{i \in K^{(j)}} x_i \tag{14}$$

where $K^{(j)}$ is a maximum clique of $G[x^{(j)}]$.

For a given selection, the cuts as given in (14) force the objective value t to be at least as large as the cardinality of a maximum clique in it. In cases where the lower bound on chromatic number is tight, (14) can replace (13). As mentioned before, if we use (13), the lower bound on t reduces by one for each vertex differing from the selection given by $x^{(j)}$. Changing one vertex from the selection that belongs to a maximum clique of it will not necessarily decrease the lower bound in (14), because the selection may contain other maximum cliques which remain intact after this change. However, the lower bound in (13) would decrease by one in this case.

Therefore, constraint (14) is stronger than (13) because it does not decrease the lower bound on t by each vertex switch; it does so only when the changed vertex lies in a clique whose cut has been added before.

In this study, we focus on three graph classes where the chromatic number equals maximum clique size for all induced subgraphs of the input graph, which means that they are perfect graph families, and we make use of only (14) in our solution procedure.

We note that constraints (14) can be interpreted from a combinatorial point of view. In particular, each selective clique found in earlier iterations has a corresponding constraint (14) added to the master problem. Each such constraint forces the value of the t-variable to be greater than or equal to the number of vertices in the intersection of its corresponding selective clique and the current vertex selection (represented by the x-variables). Since the master problem minimizes t, it seeks a selection that yields a smallest intersection with previously generated selective cliques, and the subproblem finds a maximum clique in the induced subgraph identified by the vertex selection. If the subproblem agrees on the same objective value with the master problem, then it means that the union of selective cliques for which cuts (14) have been added to the master problem contains a maximum clique of the current selection. In this case, we conclude that a better selection cannot be achieved and the process should be terminated. Pseudo-code of our decomposition algorithm for perfect graphs is given below.

Decomposition Algorithm for Perfect Graphs

Input: A perfect graph $G = (V, E)$, and a partition $\mathcal{V}$ of V

$j \leftarrow 0$, $t^{(j)} \leftarrow 0$, $z_{sp}^{(j)} \leftarrow \infty$
while true
  $j \leftarrow j + 1$
  Solve the master problem to optimality, find an optimal selection $x^{(j)}$,
    having optimal objective value $t^{(j)}$
  Find a maximum clique $K^{(j)}$ of $G[x^{(j)}]$ in the subproblem
  $z_{sp}^{(j)} \leftarrow |K^{(j)}|$
  if $z_{sp}^{(j)} > t^{(j)}$
    Add (14) to the master problem
  else
    break
$x^* \leftarrow x^{(j)}$, $z^* \leftarrow t^{(j)}$

Output: An optimal selection x^* with $\chi_{SEL}(G, \mathcal{V}) = z^*$.

The algorithm given above continues to search for a selection as long as the objective value of the subproblem exceeds that of the master problem. At each step j, the master problem is solved to optimality to obtain a selection $x^{(j)}$ with associated optimal objective value $t^{(j)}$. If the subproblem finds a maximum clique in $G[x^{(j)}]$ of size $z_{sp}^{(j)} = t^{(j)}$, then the process is terminated. At this point, the incumbent solution x^* and its corresponding objective value t^* are optimal.

Permutation, generalized split and chordal graphs are all *hereditary* graph classes; that is, induced subgraphs of a graph from any of those classes belong to the same class as the original graph does. Therefore, polynomial-time combinatorial methods tailored for these given classes also work on the subgraphs induced by selections and thus can be employed in the subroutine of the decomposition procedure. As already mentioned in Section 1.2, even though the solution procedure is facilitated by making use of some efficient algorithms for the graph coloring problem in these graph classes, Sel-Col still remains NP-hard because of the difficulty inherent to the selection task. With constraints (14), it may be possible to reach an optimal solution without having to evaluate every single selection, but the number of selections to evaluate can be exponential in general.

In the next three subsections, we give details about how the proposed method is applied for each one of the graph classes we concentrate on.

2.2.1 Permutation Graphs

Permutation graphs have many real world applications from flight altitude assignment to the memory reallocation problem [13]. One specific domain of application that motivates the study of the selective coloring problem in the class of permutation graphs is given by Demange *et al.* in [4]. The application is about allocating items to tracks from their own pick-up points to their own delivery spots, where both pick-up and delivery points have a predetermined sequence to be visited. Each track has a first-in last-out structure and swapping of items on tracks is not allowed. When each item has different possibilities of pick-up and/or delivery points, the problem consists in selecting one pick-up and one delivery point per item in order to minimize the total number of tracks required, which is shown to be equivalent to Sel-Col in permutation graphs by representing each item with a set of properly defined intervals on the real line [4].

Given a permutation π, $\pi(i)$ denotes the number in position i of π, and $\pi^{-1}(i)$ denotes the position of number i in π. A graph is a *permutation graph* if there is a permutation π of its n vertices in such a way that v_i and v_j are adjacent if and only if $i < j$ and $\pi^{-1}(v_i) > \pi^{-1}(v_j)$ (or equivalently $i > j$ and $\pi^{-1}(v_i) < \pi^{-1}(v_j)$). Figure 2 shows two permutation graphs. Note that edge $\{2, 4\}$ exists in both graphs since position number of 4 is smaller than 2 in both permutations. Similarly, there is no edge between vertices 2 and 5 in either graph since 2 is positioned earlier than 5 in both permutations.

$\pi = (3, 4, 2, 5, 1)$ $\pi = (4, 6, 2, 1, 5, 3)$

Figure 2: Two example permutation graphs

In order to apply the decomposition idea presented previously, we need to find

maximum clique(s) in the graphs induced by the selections identified by the master problem. Since each induced subgraph of a permutation graph is again a permutation graph, we will make use of a general polynomial-time algorithm given in [13] to find maximum cliques in permutation graphs. It follows from the definition of permutation graphs that vertices corresponding to a decreasing subsequence in a given permutation form a clique. Then, a maximum clique in a permutation graph can be found by identifying a longest decreasing subsequence in the corresponding permutation. For this purpose, each number is considered in the order in which it appears in π, and is enqueued to the end of the first available set. A set is available for some number $\pi(i)$ if the last number in the set, say $\pi(j)$ for some $j < i$, is less than $\pi(i)$. Otherwise, $\pi(j)$ becomes the predecessor of $\pi(i)$. The sets will be complete after considering all numbers in π. It can be observed that each set corresponds to an increasing subsequence in π and thus is a stable set of the related graph. One should note that the predecessor of a number must be greater than itself. Moreover, it is known that such a partition of vertices into stable sets yields a minimum coloring of G. Consequently, we can identify a maximum clique (equivalently a longest decreasing subsequence) by starting from a number in the last stable set and tracing back the predecessors all the way up to the first stable set. Thus, the size of a maximum clique will be equal to the number of stable sets constructed.

It is also possible to generate all maximum cliques in the graph. For this purpose, we can list all predecessors of each number and trace back through each one to reveal all maximum cliques in the graph. To list all predecessors of a number $\pi(i)$, after setting the first predecessor of it, we can simply search back from the index of first predecessor in that stable set, and include all numbers that are greater than $\pi(i)$ [13]. Although the number of maximum cliques in a permutation graph can be exponential, this algorithm to find all maximum cliques turned out to perform quite efficiently in practice. However, in our preliminary experiments, we observed that generating a single maximum clique and adding related cuts each time the subproblem is called yields better results. Therefore, each time the subproblem is called, we generate one maximum clique in the graph induced by that selection.

2.2.2 Generalized Split Graphs

An important characteristic of generalized split graphs is that they are shown to comprise almost all perfect graphs [22]. Since it is such a large subclass of perfect graphs, it is important to test the performance of our decomposition approach on them.

A graph G is a *k-partite graph* if its vertex set can be partitioned into k different stable sets. In a *complete k-partite graph*, all possible edges between each pair of k stable sets exist. A graph $G = (V, E)$ is a *generalized split graph* if there exists a partition $V = V_1 \cup V_2$ of the vertex set such that either $G[V_1]$ is a union of pairwise disjoint cliques and V_2 is a clique in G, in which case the graph is called *unipolar*, or $\bar{G}[V_1]$ is a union of pairwise disjoint cliques (in other words $G[V_1]$ is a complete multi-partite graph) and V_2 is a stable set in G, in which case it is called *co-unipolar*. Figure 3 shows visual representations of generalized split graphs.

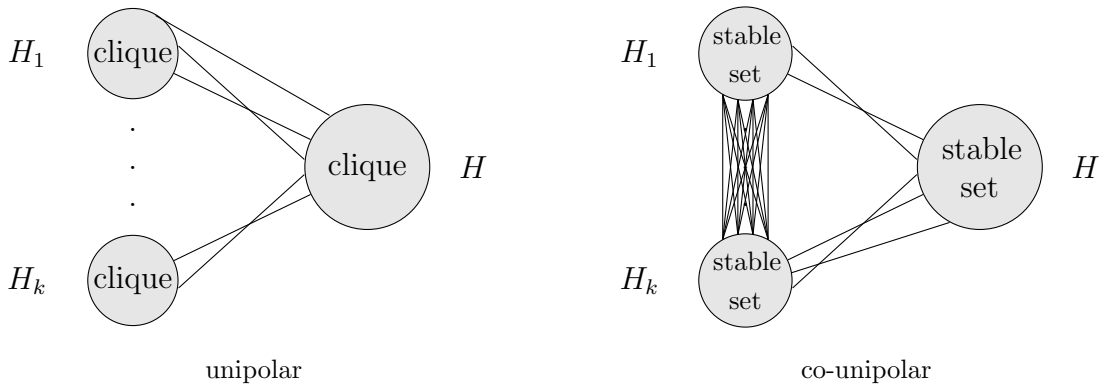

Figure 3: Two alternative forms of generalized split graphs

Let $G = (V, E)$ be a generalized split graph with $V = V_1 \cup V_2$ such that $V_1 = \bigcup_{i=1}^{k} H_i$ and $V_2 = H$. If G is unipolar, V_1 is the union of k disjoint cliques $H_1, \ldots, H_k$ and V_2 is a clique; if it is co-unipolar, V_1 induces a complete k-partite graph with k-partition $H_1, \ldots, H_k$ and V_2 is a stable set. In order to find maximum cliques in a generalized split graph, we first need to know whether it is unipolar or co-unipolar. When generating random generalized split graphs (the generation method is explained in Section 3.1.2), we label the output graph as unipolar or co-unipolar and also record its partition to make use of them later in our solution procedure.

In the unipolar case, we use the polynomial-time method that Eschen and Wang utilize in [7]. The first observation is that since the H_is for $i = 1, \ldots, k$ are disjoint, a clique cannot contain vertices from distinct H_i components. Hence, one should identify a maximum clique in each one of the subgraphs $G[H \cup H_i]$ and find the largest among the k cliques. In order to find a maximum clique in each one of these k induced subgraphs easily, the authors in [7] remark that the complement of $G[H \cup H_i]$ forms a bipartite graph with bipartition H, H_i. The well-known König–Egerváry Theorem [6, 15] states that in a bipartite graph the size of a maximum matching equals the size of a minimum vertex cover. Moreover, if V' is a minimum vertex cover in $G = (V, E)$, then $V \setminus V'$ is a maximum stable set of G. So, in order to find a maximum clique in a unipolar graph, we seek maximum stable sets in the complement of each $G[H \cup H_i]$ to find a maximum clique in the entire graph because a stable set in the complement corresponds to a clique in the original graph.

To find a maximum clique in a co-unipolar graph, we first explore a vertex v in H that is connected to the maximum number of distinct H_i components. Since the H_is form a complete k-partite graph, the neighbours in distinct H_i components together with v form a clique. If the size of that clique is greater than k, then we are done. Otherwise, we can arbitrarily select a single vertex from each H_i for $i = 1, \ldots, k$ and output it as a maximum clique. This process takes polynomial time, in particular $O(|V| + |E|)$.

As in permutation graphs, it is possible to list all maximum cliques in a given generalized split graph. However, especially in cases where the graph is co-unipolar and the maximum cliques turn out to be all combinations of vertices from each H_i, the number of possible maximum cliques can be exponentially many. In our preliminary experiments, we observed that it is not practical to enumerate all maximum cliques. Hence, we confine ourselves to finding a single maximum clique each time

the subproblem is called. Furthermore, in the case of unipolar graphs, we extend that single maximum clique of the selection to a maximal clique in the whole graph to strengthen our cut (14).

2.2.3 Chordal Graphs

Chordal graphs generalize interval graphs, which are intersection graphs of a family of intervals on a linearly ordered set (e.g., the real line) and play an important role in scheduling applications [13]. In addition, exact solution methods for Sel-Col in interval graphs are particularly emphasized as a research direction in [4]. Being a generalized version of interval graphs, an efficient exact solution procedure for chordal graphs will serve useful purposes in relevant application areas.

A graph is called *chordal* (or *triangulated*) if every cycle of length ≥ 4 contains a chord where a *chord* is an edge between two non-consecutive vertices of a cycle. Chordal graphs can alternatively be defined as graphs not containing induced cycles of length at least 4.

(a) (b)

Figure 4: (a) A chordal graph on five vertices, (b) A non-chordal graph due to the induced cycle 2-3-4-5-2

A chordal graph on n vertices has at most n maximal cliques, with equality if and only if the graph contains no edge [9]. Since the upper bound on the number of maximal cliques is reasonably small and there exists a linear-time algorithm to generate all maximal cliques in a given chordal graph [1], we generate all maximal cliques of G beforehand and construct the associated cut set that is mathematically expressed in (15).

$$t \geq \sum_{i \in K} x_i \quad \forall K \in \mathcal{K}, \tag{15}$$

where $\mathcal{K}$ denotes the set of all maximal cliques for the given chordal graph G.

We tailored our decomposition procedure for the case of chordal graphs in a slightly different (but equivalent) manner: The master problem starts with finding an optimal selection without any clique constraints imposed. Given an optimal selection identified by the master problem, the subproblem checks whether any of the constraints from the set (15) are violated and incorporates such constraints, if any, into the master. In other words, instead of generating the clique cuts one by one at each step, we construct them beforehand and add as needed, because the

number of constraints to go through is linear in the size of the vertex set of the input graph in the case of chordal graphs. If, at some iteration, the master finds a selection which does not violate any of the constraints in (15), it means that the current state of the master is able to produce an optimal solution that satisfies all the clique constraints and hence an optimal solution for the entire problem is reached. It should be noted that the constraint set (15) is defined for the maximal cliques on the entire graph, not on selections. However, the formulation of the master already involves constraints making the x-variables to take value 1 only for vertices in the selection, which guarantees that the t-variable, equivalently the objective value, is forced to be at least as large as the cardinality of the intersection of a maximal clique with the selected set of vertices. Moreover, a maximal clique of a selection can always be extended to a maximal clique in the whole graph, which means that the set of all selective maximal cliques are covered by the set of all maximal cliques of the whole graph. Since the constraint set (15) is defined for each one of the maximal cliques in the graph, the minimum value that variable t ultimately takes on for some given selection will be exactly the size of a maximum clique in that selection.

3 Computational Study

3.1 Data Generation

For all the graph classes that we focus on, we need random instances to test the performance of our solution approach. We briefly describe how the test instances are obtained for each one of the three graph classes.

3.1.1 Random Permutation Graph Generation

Our algorithm to generate permutation graphs (PermGraphGen) simply implements the definition of permutation graphs. Given n as the number of vertices and p as a probability value to manipulate the edge density of the output graph, the algorithm first generates a random ordering π of the numbers $1, \ldots, n$, where $\pi(i)$ denotes the number in position i in π. To produce a random ordering, each number $\pi(i)$ is considered to be swapped with an element at some randomly picked index later than i. If the number at the randomly chosen index is greater than $\pi(i)$, in which case a swap would lead to a rise in the number of edges, the swap takes place with probability p; otherwise, we interchange places of the two with probability $1 - p$. Thus, as p gets larger, density of the output graph tends to get higher. Finally, the algorithm makes two vertices v_i and v_j adjacent only if $i < j$ and $\pi^{-1}(i) > \pi^{-1}(j)$.

Algorithm PermGraphGen

Input: An integer n and a probability value p

Let π be an array of size n with $\pi(i) = i$ for $i = 1, \ldots, n$
for $i = 1$ **to** n **do**
  Pick a random index r greater than i
  if $\pi(r) > \pi(i)$
    Swap the two elements with probability p
  else
    Swap with probability $1 - p$
Let $G = (V, E)$ be an edgeless graph with $V = \{v_1, \ldots, v_n\}$
for $i = 1$ **to** n **do**
  for $j = i + 1$ **to** n **do**
    if $\pi^{-1}(i) > \pi^{-1}(j)$ **then** add an edge between v_i and v_j

Output: Permutation graph $G = (V, E)$ on n vertices

3.1.2 Random Generalized Split Graph Generation

As discussed in Section 2.2.2, a generalized split graph is one such that either itself or its complement is a unipolar graph. Our algorithm (GSGGen) takes the number of vertices n, an upper bound on the number of vertices in V_1, and a desired edge density value ρ as input. It first creates a random partition of the vertex set, then decides on the form of the graph to be unipolar or co-unipolar, and finally adds random edges between V_1 and $V_2 = H$.

Algorithm GSGGen

Input: Two integers n and u, and a real number ρ between 0 and 1

Create a random ordering σ of the vertex set $V = \{v_1, \ldots, v_n\}$
Let n_1 be a random integer between l and u where $l = f(n, u)$
$H \leftarrow \{\sigma_{n_1+1}, \ldots, \sigma_n\}$
Let k be a random integer set as a function of ρ and n_1
Randomly divide $\{\sigma_1, \ldots, \sigma_{n_1}\}$ into k disjoint sets $H_1, \ldots, H_k$
if unipolar
  Add edges to make $G[H]$ and $G[H_i]$ $\forall i \in \{1, \ldots, k\}$ cliques each
else
  Add edges to make $G\big[\bigcup_{i=1}^{k} H_i\big]$ complete k-partite
Add random edges between $G[H]$ and $G[\bigcup_{i=1}^{k} H_i]$ to (approximately) achieve the desired edge density ρ

Output: Generalized split graph G on n vertices with (approximate) edge density ρ

3.1.3 Random Chordal Graph Generation

We generated the random chordal graph instances via the method given in [23]. This method is shown to generate the most varied chordal graphs in terms of sizes of maximal cliques and is based on a well-known characterization of chordal graphs that states a graph is chordal if and only if it is the intersection graph of subtrees of a tree [12].

3.2 Experimental Results

In this section, we present results of a series of experiments to evaluate the efficiency of our decomposition procedure and compare them to those of the integer programming formulation Model 1 and to the branch-and-price algorithm by Furini *et al.* [10].

We implemented the algorithms described in the previous section in C++, and executed them on a computer with 2.00-GHz Intel Xeon CPU. Throughout all the experiments, we used CPLEX version 12.8, and set a time limit of 20 minutes for each one. For the decomposition algorithms used for permutation and generalized split graphs, we used the callback mechanism of CPLEX. In the case of chordal graphs, we utilized the lazy constraint feature of CPLEX. We made our problem instances and source code available at `http://www.ie.boun.edu.tr/~taskin/data/scpgf/`.

We randomly generated our test instances for different n values ranging from 100 to 1000, and four different average edge density values 0.1, 0.3, 0.5, and 0.7, where the edge density of a graph is defined as $\frac{m}{\frac{n(n-1)}{2}}$, with m denoting the number of edges. For each pair of n and average edge density value, we used 10 graph instances.

We need a partition of the vertex set into clusters for each problem instance. In order to test the effect of cluster sizes on the performance of the suggested methods, we generated three different partitions for each instance, where the sizes of clusters uniformly vary between the lower and upper bounds that are chosen as $\{2, 5\}$, $\{4, 7\}$, and $\{6, 9\}$. Our procedure to set the partition takes lower and upper bounds on the sizes of clusters and creates a random ordering of vertices. Then, according to the bounds input, it sets the sizes of clusters and places separator points on the random sequence one at a time, so that the set of vertices between two separator points serves as one cluster.

To improve the performance of our decomposition approach for permutation graphs and generalized split graphs, we applied some initial treatment before starting the solution procedure. In particular, we initially searched for maximal cliques in the entire graph by using the algorithms explained in Sections 2.2.1-2.2.2 and added them as clique constraints to the master program beforehand. Since permutation and generalized split graphs are hereditary, clique constraints generated on the entire graph serve as valid inequalities. We also applied a heuristic method to find a selection, found a maximum clique in it and used this as an initial solution to be used as upper bound. Our heuristic method chooses a vertex from each cluster that has the fewest number of neighbours in other clusters. These initial treatments are also adapted to the IP formulation Model 1 in order to compare it to our decomposition method on equal terms. We obtained slightly improved results with these in the

initial treatment steps, and we report these in Tables 1–6.

In our first set of experiments, we test the performance of the IP formulation, the B&P algorithm by Furini *et al.* [10] and our decomposition approach on permutation graphs. Table 1 summarizes the computational results for permutation graph instances with cluster sizes varying between 2 and 5. The first three columns in this table present the number of vertices ("n"), average edge density ("avg density"), and average number of clusters ("avg # clust") across ten random instances. In the next three groups of columns, we report the results of our experiments for the three algorithms under "IP formulation", "B&P", and "Decomposition" headings, respectively. For the IP formulation, columns 4–8 show the number of instances that could be optimally solved among ten ("# opt"), average optimality gap percentages (calculated as $\frac{UB-LB}{UB} \times 100$ where UB and LB denote the upper and lower bounds respectively) over instances that could not be solved to optimality within the given time limit of 20 minutes ("avg % gap in nonopt") and over all instances ("avg % gap overall"), average solution time in seconds over instances that are optimally solved ("avg time in opt") and over all instances ("avg time overall"). Columns 9–13 and 14–18 list the same set of results respectively for B&P of Furini *et al.* and our decomposition method. Finally, the rightmost column shows the average time spent in the subproblem of our decomposition algorithm in seconds across ten instances ("avg time in subpr"). In each row of these tables, we report average values across ten independent runs. The upper half of the table shows the results for low density values 0.1 and 0.3, and the second for high density values 0.5 and 0.7. At the end of each half, the total number of instances solved to optimality, and average time and optimality gaps are reported for each one of the three methods.

For each one of the three methods, we used a time limit of 1200 seconds. If an instance could not be solved optimally within the limit, the time spent for that instance is taken as 1200 seconds. In our experiments, the B&P algorithm by Furini *et al.* failed to report optimality gaps for some instances that could not be solved optimally. For such cases, we take the optimality gap as 100%.

Our observation from the results listed in Table 1 is that in permutation graphs with low density, i.e., 0.1 and 0.3, the decomposition algorithm clearly outperforms the IP formulation and B&P method in terms of the number of instances solved to optimality, average optimality gap, and average amount of time spent on instances that are solved to optimality and in general. In permutation graphs with high density, i.e., 0.5 and 0.7, performance of the IP formulation and our method worsens in general, whereas that of B&P improves. As n grows, the performance of all three methods deteriorates in general. Increasing edge density, however, results in improved performance for the B&P method, while deteriorating that of the IP formulation and the decomposition method. The results in the lower half of Table 1 indicate that the decomposition method outperforms the other two in terms of average optimality gap and average time spent in instances that could be solved optimally. However, although the decomposition method outperforms the IP formulation in every aspect, B&P is able to solve the highest number of instances to optimality.

We conducted two additional sets of experiments on permutation graphs in order to test the effect of cluster sizes. Tables 2–3 show the results obtained on the same set of graphs as before but with cluster sizes between 4–7 and 6–9, respectively.

Table 1: Experimental results for permutation graph instances with small clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.103	28.7	10		0.00	0.63	0.63	10		0.00	0.77	0.77	10		0.00	0.23	0.23	0.03
	0.300	27.2	10		0.00	4.83	4.83	10		0.00	35.24	35.24	10		0.00	0.32	0.32	0.05
200	0.102	56.3	10		0.00	8.23	8.23	9	100.00	10.00	0.48	120.43	10		0.00	0.16	0.16	0.03
	0.304	56.5	10		0.00	207.08	207.08	10		0.00	260.33	260.33	10		0.00	0.64	0.64	0.14
300	0.103	85.7	10		0.00	36.11	36.11	9	100.00	10.00	380.17	462.15	10		0.00	0.18	0.18	0.05
	0.309	85.0	0	90.45	90.45		1200.00	3	100.00	70.00	1071.58	1161.47	10		0.00	7.05	7.05	0.65
400	0.105	115.2	10		0.00	328.36	328.36	2	100.00	80.00	3.91	960.78	10		0.00	0.30	0.30	0.12
	0.302	112.7	0	96.25	96.25		1200.00	0	100.00	100.00		1200.00	10		0.00	33.32	33.32	2.51
500	0.101	141.1	7	80.46	24.14	383.44	628.41	1	100.00	90.00	14.39	1081.44	10		0.00	0.26	0.26	0.12
	0.306	140.4	0	99.05	99.05		1200.00	0	100.00	100.00		1200.00	8	20.00	4.00	236.39	429.11	9.12
			67	**93.90**	**30.99**	**138.39**	**481.37**	**54**	**100.00**	**46.00**	**220.86**	**648.26**	**98**	**20.00**	**0.40**	**27.89**	**47.16**	**1.28**
100	0.500	29.4	10		0.00	14.08	14.08	10		0.00	19.92	19.92	10		0.00	0.56	0.56	0.07
	0.707	28.1	10		0.00	51.46	51.46	10		0.00	18.35	18.35	10		0.00	5.97	5.97	0.13
200	0.497	55.9	7	76.19	22.86	624.69	797.29	10		0.00	169.24	169.24	10		0.00	100.30	100.30	1.57
	0.703	57.5	0	86.91	86.91		1200.00	10		0.00	116.00	116.00	3	14.29	10.00	276.96	923.09	3.35
300	0.499	85.5	0	98.04	98.04		1200.00	10		0.00	489.77	489.77	4	20.56	12.33	136.91	774.77	7.55
	0.701	85.2	0	100.00	100.00		1200.00	10		0.00	360.35	360.35	0	40.97	40.97		1200.00	29.32
400	0.497	114.9	0	100.00	100.00		1200.00	4	100.00	60.00	1028.71	1131.48	0	24.38	24.38		1200.00	22.11
	0.698	114.9	0	100.00	100.00		1200.00	10		0.00	799.98	799.98	0	48.84	48.84		1200.00	24.80
500	0.502	141.9	0	100.00	100.00		1200.00	0	100.00	100.00		1200.00	0	34.64	34.64		1200.00	31.75
	0.700	142.8	0	100.00	100.00		1200.00	0	100.00	100.00		1200.00	0	62.01	62.01		1200.00	33.56
			27	**96.96**	**70.78**	**230.08**	**926.28**	**74**	**100.00**	**26.00**	**375.29**	**550.51**	**37**	**37.01**	**23.32**	**104.14**	**780.47**	**15.42**

Table 2: Experimental results for permutation graph instances with medium-sized clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.103	17.8	10		0.00	0.42	0.42	10		0.00	0.03	0.03	10		0.00	0.24	0.24	0.03
	0.300	18.1	10		0.00	1.70	1.70	10		0.00	1.93	1.93	10		0.00	0.25	0.25	0.04
200	0.102	36.6	10		0.00	5.21	5.21	10		0.00	0.21	0.21	10		0.00	0.22	0.22	0.04
	0.304	36.3	10		0.00	37.99	37.99	10		0.00	45.15	45.15	10		0.00	0.50	0.50	0.10
300	0.103	54.9	10		0.00	15.37	15.37	10		0.00	1.23	1.23	10		0.00	0.15	0.15	0.03
	0.309	54.3	6	70.83	28.33	416.25	729.75	10		0.00	289.44	289.44	10		0.00	1.63	1.63	0.25
400	0.105	72.6	10		0.00	56.19	56.19	10		0.00	3.02	3.02	10		0.00	0.22	0.22	0.09
	0.302	72.3	0	84.07	84.07		1200.00	8	100.00	20.00	967.45	1013.96	10		0.00	4.12	4.12	0.59
500	0.101	90.3	10		0.00	117.40	117.40	10		0.00	12.43	12.43	10		0.00	0.28	0.28	0.13
	0.306	89.6	0	93.71	93.71		1200.00	0	100.00	100.00		1200.00	10		0.00	30.47	30.47	1.95
			76	**85.88**	**20.61**	**81.32**	**336.40**	**88**	**100.00**	**12.00**	**146.76**	**256.74**	**100**	**-**	**0.00**	**3.81**	**3.81**	**0.32**
100	0.500	18.4	10		0.00	6.04	6.04	10		0.00	8.76	8.76	10		0.00	0.39	0.39	0.06
	0.707	18.1	10		0.00	21.97	21.97	10		0.00	11.49	11.49	10		0.00	3.50	3.50	0.10
200	0.497	35.6	9	100.00	10.00	194.64	295.18	10		0.00	98.17	98.17	10		0.00	19.54	19.54	0.59
	0.703	35.8	7	75.00	22.50	565.28	755.70	10		0.00	80.88	80.88	2	25.00	20.00	600.30	1080.06	2.90
300	0.499	54.3	0	97.89	97.89		1200.00	10		0.00	376.98	376.98	7	33.33	10.00	331.44	592.01	3.04
	0.701	54.2	0	100.00	100.00		1200.00	10		0.00	277.88	277.88	0	51.38	51.38		1200.00	26.10
400	0.497	72.5	0	100.00	100.00		1200.00	10		0.00	981.12	981.12	3	45.48	31.83	669.48	1040.84	17.85
	0.698	72.1	0	100.00	100.00		1200.00	10		0.00	658.14	658.14	0	65.82	65.82		1200.00	32.29
500	0.502	89.2	0	98.71	98.71		1200.00	0	100.00	100.00		1200.00	0	46.00	46.00		1200.00	18.31
	0.700	90.4	0	100.00	100.00		1200.00	0	100.00	100.00		1200.00	0	72.79	72.79		1200.00	37.94
			36	**98.30**	**62.91**	**196.98**	**827.89**	**80**	**100.00**	**20.00**	**311.68**	**489.34**	**42**	**51.35**	**29.78**	**270.77**	**753.63**	**13.92**

Table 3: Experimental results for permutation graph instances with large clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.103	13.4	10		0.00	0.17	0.17	10		0.00	0.03	0.03	10		0.00	0.10	0.10	0.01
	0.300	12.8	10		0.00	0.60	0.60	10		0.00	0.04	0.04	10		0.00	0.12	0.12	0.02
200	0.102	26.3	10		0.00	2.73	2.73	10		0.00	0.09	0.09	10		0.00	0.12	0.12	0.02
	0.304	26.5	10		0.00	7.04	7.04	10		0.00	0.28	0.28	10		0.00	0.24	0.24	0.06
300	0.103	39.7	10		0.00	14.10	14.10	10		0.00	0.82	0.82	10		0.00	0.13	0.13	0.03
	0.309	40.0	10		0.00	217.55	217.55	10		0.00	17.20	17.20	10		0.00	0.72	0.72	0.26
400	0.105	53.6	10		0.00	33.01	33.01	10		0.00	2.40	2.40	10		0.00	0.25	0.25	0.10
	0.302	52.8	8	100.00	20.00	217.56	414.05	10		0.00	242.04	242.04	10		0.00	6.53	6.53	0.98
500	0.101	66.4	10		0.00	61.27	61.27	10		0.00	5.38	5.38	10		0.00	0.25	0.25	0.10
	0.306	65.9	7	77.78	23.33	837.36	946.15	9	100.00	10.00	287.90	379.11	10		0.00	2.35	2.35	0.82
			95	**86.67**	**4.33**	**139.14**	**169.67**	**99**	**100.00**	**1.00**	**55.62**	**64.74**	**100**	**-**	**0.00**	**1.08**	**1.08**	**0.24**
100	0.500	13.0	10		0.00	1.94	1.94	10		0.00	0.30	0.30	10		0.00	0.17	0.17	0.03
	0.707	13.1	10		0.00	10.36	10.36	10		0.00	6.49	6.49	10		0.00	1.13	1.13	0.07
200	0.497	26.4	9	100.00	10.00	76.90	189.21	10		0.00	49.18	49.18	10		0.00	10.36	10.36	0.34
	0.703	26.4	8	83.33	16.67	313.96	491.17	10		0.00	45.06	45.06	7	38.89	11.67	414.58	650.21	1.94
300	0.499	39.0	3	86.90	60.83	830.53	1089.16	10		0.00	196.23	196.23	10		0.00	104.04	104.04	3.22
	0.701	39.6	0	100.00	100.00		1200.00	10		0.00	166.37	166.37	0	54.33	54.33		1200.00	22.88
400	0.497	52.9	0	95.76	95.76		1200.00	10		0.00	536.61	536.61	5	40.00	20.00	222.86	711.43	15.96
	0.698	53.9	0	100.00	100.00		1200.00	10		0.00	459.61	459.61	0	68.88	68.88		1200.00	29.59
500	0.502	66.9	0	97.50	97.50		1200.00	5	100.00	50.00	1151.27	1175.64	0	39.17	39.17		1200.00	15.73
	0.700	66.3	0	100.00	100.00		1200.00	10		0.00	794.16	794.16	0	75.42	75.42		1200.00	56.99
			40	**96.79**	**58.08**	**246.74**	**778.18**	**95**	**100.00**	**5.00**	**340.53**	**342.97**	**52**	**56.14**	**26.95**	**125.52**	**627.73**	**14.68**

Comparing the results in Table 1 to those in Tables 2–3, we observe considerable improvement in the performance of all methods. For a given n value, as the average size of clusters increases, the total number of clusters and hence the number of variables and constraints in Model 1 reduce. This reduction in the size of the IP formulation leads to improved performance. Comparative performance of the three methods is similar to the case with small clusters; the decomposition method outperforms the other two in all respects for low density graphs, while B&P manages to yield the highest number of optimally solved instances. The average time that the decomposition method spends for optimally solved instances is again significantly lower than those of the other two, regardless of edge density.

Next, we present the results of our computational experiments for generalized split graphs in Tables 4–6 in the same order and structure as in the case of permutation graphs. We observe from the upper half of Table 4 that for relatively low-density graphs, the IP formulation was able to solve a slightly larger percentage of test instances to optimality as compared to the decomposition method. However, the decomposition method yields better results in terms of all other measures compared to both methods. As for the results obtained from high density instances shown in the lower half of the table, B&P is able to solve slightly higher numbers of instances optimally; but the decomposition approach is able to result in better average optimality gap percentages and average times. The performance of all three methods gets worse in general as n increases. As in the case of permutation graphs, when we compare the results in Table 4 to those in Tables 5–6, we observe enhanced performance in the IP formulation with the increase in cluster sizes due to reduction in the size of the IP formulation. As far as the number of instances solved to optimality is concerned, the decomposition method produces more robust results. As cluster sizes increase, the performance of B&P also improves in low-density instances, but we cannot observe a monotonic trend in high-density instances.

Our last set of experiments is conducted on chordal graph instances. Tables 7–12 show the results for small, medium and large cluster sizes and for low and high edge densities, respectively. The general structure of these tables is the same as before, except that we additionally include the average number of maximal cliques across ten graph instances ("avg # cliques"), which is an important indicator of how well the algorithm performs, and exclude the column for time spent in the subproblem.

All the chordal graph instances are solved to optimality under one second by our decomposition approach, as the results in Tables 7–12 reveal. Here, we experimented with graphs up to 1000 vertices in order to be able to clearly demonstrate the gap between performances of the three algorithms. For a given n, the solution times of our algorithm decrease as edge density increases due to a parallel decline in the number of maximal cliques. The IP formulation can also solve a large percentage of chordal graph instances to optimality within the permitted time limit; yet, our method clearly outperformed it time-wise. When the clusters in the partition are small-sized, the IP failed to optimally solve many of the chordal graph instances with 800 or more vertices and could not give reasonable optimality gaps or even find a feasible solution within the allowed time limit. The B&P method performs poorly as compared to both of the other methods; it fails to solve instances with 600 or more vertices in all cases.

In all three graph classes we consider, our decomposition method significantly

Table 4: Experimental results for generalized split graph instances with small clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.101	29.9	10		0.00	0.74	0.74	10		0.00	15.53	15.53	10		0.00	0.30	0.30	0.03
	0.299	28.7	10		0.00	4.09	4.09	10		0.00	22.38	22.38	9	14.29	1.43	0.31	120.28	0.21
200	0.101	58.4	10		0.00	4.42	4.42	10		0.00	119.30	119.30	7	25.00	7.50	4.78	363.34	1.13
	0.300	58.3	10		0.00	78.86	78.86	10		0.00	173.60	173.60	7	13.10	3.93	0.46	360.32	33.17
300	0.102	86.1	10		0.00	38.95	38.95	9	100.00	10.00	432.26	509.03	7	23.33	7.00	100.29	430.21	8.60
	0.300	85.3	8	87.50	17.50	298.75	479.00	8	100.00	20.00	475.71	620.57	6	24.79	9.92	1.13	480.68	29.86
400	0.101	115.9	10		0.00	163.67	163.67	6	100.00	40.00	705.57	903.34	8	22.50	4.50	36.83	269.46	4.27
	0.300	114.2	2	60.21	48.17	739.80	1107.96	5	100.00	50.00	1067.11	1133.55	7	22.22	6.67	4.45	363.11	27.49
500	0.101	143.5	9	40.00	4.00	363.05	446.75	2	100.00	80.00	944.11	1148.82	7	27.50	8.25	0.77	360.54	51.78
	0.300	144.0	1	94.60	85.14	1153.10	1195.31	0	100.00	100.00		1200.00	7	26.01	7.80	3.68	362.57	42.60
			80	**77.40**	**15.48**	**284.54**	**351.98**	**70**	**100.00**	**30.00**	**439.51**	**584.61**	**75**	**22.80**	**5.70**	**15.30**	**311.08**	**19.91**
100	0.497	28.8	10		0.00	7.30	7.30	10		0.00	17.04	17.04	9	25.00	2.50	2.07	121.86	9.56
	0.701	29.8	10		0.00	17.78	17.78	10		0.00	17.48	17.48	9	21.43	2.14	1.00	120.90	5.92
200	0.504	58.0	8	83.33	16.67	172.99	378.39	10		0.00	157.56	157.56	8	22.18	4.44	0.52	240.42	13.85
	0.700	57.5	7	52.78	15.83	355.95	609.17	10		0.00	109.51	109.51	9	21.43	2.14	32.52	149.27	13.04
300	0.501	87.5	6	72.78	29.11	368.70	701.22	10		0.00	338.01	338.01	9	25.00	2.50	140.84	246.76	11.50
	0.695	85.2	5	100.00	50.00	152.55	676.28	10		0.00	377.56	377.56	5	19.63	9.82	18.81	609.41	10.00
400	0.500	114.9	0	69.22	69.22		1200.00	9	100.00	10.00	739.04	785.13	7	21.13	6.34	11.85	368.30	13.59
	0.699	115.8	1	79.94	71.95	192.23	1099.22	8	100.00	20.00	689.02	791.21	3	18.55	12.98	0.47	840.14	28.21
500	0.500	142.4	0	96.85	96.85		1200.00	5	100.00	50.00	1120.67	1160.33	5	24.75	12.37	24.47	612.23	43.72
	0.700	143.8	1	82.41	74.17	949.35	1174.94	4	100.00	60.00	834.08	1053.63	2	21.51	17.21	0.79	960.16	139.41
			48	**81.50**	**42.38**	**277.11**	**706.43**	**86**	**100.00**	**14.00**	**440.00**	**480.75**	**66**	**21.31**	**7.24**	**23.34**	**426.94**	**28.88**

Table 5: Experimental results for generalized split graph instances with medium-sized clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.101	18.2	10		0.00	0.25	0.25	10		0.00	2.84	2.84	10		0.00	1.95	1.95	0.03
	0.299	18.2	10		0.00	1.35	1.35	10		0.00	3.84	3.84	9	20.00	2.00	1.53	121.38	3.17
200	0.101	36.4	10		0.00	3.46	3.46	10		0.00	28.07	28.07	10		0.00	2.25	2.25	0.08
	0.300	36.5	10		0.00	16.96	16.96	10		0.00	77.64	77.64	8	19.64	3.93	74.15	299.32	7.78
300	0.102	54.6	10		0.00	16.57	16.57	10		0.00	94.93	94.93	6	41.67	16.67	5.11	483.07	10.35
	0.300	54.3	9	100.00	10.00	140.88	246.79	10		0.00	449.18	449.18	7	44.29	13.29	129.46	450.62	18.90
400	0.101	73.4	10		0.00	77.94	77.94	10		0.00	294.48	294.48	8	33.33	6.67	45.20	276.16	6.87
	0.300	73.3	8	75.00	15.00	456.11	604.89	4	100.00	60.00	757.85	1023.14	7	33.33	10.00	5.62	363.93	14.77
500	0.101	90.5	10		0.00	231.34	231.34	6	100.00	40.00	79.93	527.96	7	47.78	14.33	19.10	373.37	19.61
	0.300	90.5	3	56.62	39.63	496.33	988.90	1	100.00	90.00	607.83	1140.78	6	38.87	15.55	2.16	481.30	37.40
			90	**64.63**	**6.46**	**144.12**	**218.85**	**81**	**100.00**	**19.00**	**239.66**	**364.28**	**78**	**37.47**	**8.24**	**28.65**	**285.34**	**11.90**
100	0.497	18.2	10		0.00	4.61	4.61	10		0.00	7.24	7.24	9	40.00	4.00	14.14	132.73	7.82
	0.701	17.7	10		0.00	5.01	5.01	10		0.00	28.99	28.99	9	33.33	3.33	0.59	120.53	13.03
200	0.601	36.8	10		0.00	117.66	117.66	10		0.00	89.12	89.12	8	40.87	8.17	120.36	336.29	12.67
	0.700	36.8	10		0.00	167.16	167.16	10		0.00	147.37	147.37	4	5.56	3.33	78.69	751.48	7.32
300	0.501	54.5	10		0.00	166.49	166.49	10		0.00	263.47	263.47	6	31.88	12.75	24.38	494.63	19.65
	0.695	54.5	5		0.00	53.06	626.53	6	100.00	40.00	766.78	940.07	7	21.43	6.43	41.96	389.37	3.23
400	0.500	72.4	5	90.00	45.00	545.79	872.90	7	100.00	30.00	823.50	936.45	6	37.80	15.12	69.57	521.74	20.63
	0.699	71.9	3	89.66	62.76	439.65	971.89	1	100.00	90.00	797.84	1159.78	3	26.39	18.47	27.63	848.29	20.18
500	0.500	91.3	0	97.62	97.62		1200.00	2	100.00	80.00	1022.11	1164.42	6	40.16	16.06	88.63	533.18	40.06
	0.700	90.5	2	87.50	70.00	536.96	1067.39	1	100.00	90.00	883.85	1168.38	1	32.00	28.80	1.22	1080.12	43.82
			65	**78.68**	**27.54**	**226.27**	**519.97**	**67**	**100.00**	**33.00**	**483.03**	**590.53**	**59**	**28.41**	**11.65**	**46.72**	**520.84**	**18.84**

Table 6: Experimental results for generalized split graph instances with large clusters

			IP formulation					B&P					Decomposition					
n	avg density	avg # clust	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	avg time in subpr
100	0.101	12.7	10		0.00	0.25	0.25	10		0.00	0.04	0.04	10		0.00	0.13	0.13	0.01
	0.299	13.2	10		0.00	0.71	0.71	10		0.00	5.02	5.02	9	25.00	2.50	0.77	120.69	0.78
200	0.101	26.1	10		0.00	1.08	1.08	10		0.00	24.56	24.56	8	50.00	10.00	25.81	260.65	0.18
	0.300	26.3	10		0.00	5.65	5.65	10		0.00	56.00	56.00	9	20.00	2.00	101.72	211.55	1.80
300	0.102	39.8	10		0.00	8.53	8.53	10		0.00	149.38	149.38	7	38.89	11.67	23.90	376.73	6.53
	0.300	39.6	10		0.00	40.41	40.41	10		0.00	205.34	205.34	7	38.89	11.67	11.87	368.31	12.72
400	0.101	53.4	10		0.00	29.49	29.49	9	100.00	10.00	466.75	540.08	6	50.00	20.00	104.94	542.96	1.24
	0.300	53.4	10		0.00	216.67	216.67	7	100.00	30.00	610.20	787.14	5	40.00	20.00	5.34	602.67	9.37
500	0.101	66.8	10		0.00	125.22	125.22	9	100.00	10.00	362.52	446.27	6	50.00	20.00	4.86	482.92	19.58
	0.300	65.9	9	50.00	5.00	418.26	496.43	9	100.00	10.00	368.12	451.31	6	48.33	19.33	17.41	490.45	26.89
			99	**50.00**	**0.50**	**84.63**	**92.44**	**94**	**100.00**	**6.00**	**224.79**	**266.51**	**73**	**43.40**	**11.72**	**29.67**	**345.70**	**7.91**
100	0.497	13.2	10		0.00	1.33	1.33	10		0.00	7.77	7.77	9	25.00	2.50	14.59	133.14	2.55
	0.701	13.1	10		0.00	2.52	2.52	10		0.00	15.33	15.33	9	28.57	2.86	0.45	120.41	14.15
200	0.504	26.6	10		0.00	12.68	12.68	10		0.00	86.15	86.15	8	45.00	9.00	1.15	240.92	9.50
	0.700	26.7	10		0.00	41.60	41.60	10		0.00	156.95	156.95	8	41.43	8.29	140.78	352.63	12.28
300	0.501	39.6	10		0.00	73.63	73.63	10		0.00	373.93	373.93	6	37.50	15.00	0.61	480.37	10.21
	0.695	39.6	5	60.00	30.00	59.51	629.75	10		0.00	674.24	674.24	6	25.42	10.17	95.40	537.24	7.04
400	0.500	53.1	7	61.11	18.33	295.64	566.95	5	100.00	50.00	956.08	1078.04	6	40.00	16.00	14.42	488.65	14.85
	0.699	52.8	4	85.09	51.06	449.67	899.87	10		0.00	497.87	497.87	5	35.00	17.50	257.89	728.94	22.04
500	0.500	67.4	1	82.91	74.62	621.18	1142.12	4	100.00	60.00	741.31	1016.52	7	55.71	16.71	24.72	377.30	34.67
	0.700	66.7	3	85.71	60.00	370.23	951.07	6	100.00	40.00	565.94	819.56	1	37.26	33.54	0.07	1080.01	32.82
			70	**78.00**	**23.40**	**192.80**	**432.15**	**85**	**100.00**	**15.00**	**407.56**	**472.64**	**65**	**37.59**	**13.16**	**55.01**	**453.96**	**16.01**

Table 7: Experimental results for chordal graph instances of density 0.1 and 0.3 with small clusters

				IP formulation					B&P					Decomposition				
n	avg density	avg # clust	avg # cliques	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.103	29.3	38.2	10		0.00	0.91	0.91	10		0.00	19.22	19.22	10		0.00	0.11	0.11
	0.299	28.3	25.0	10		0.00	1.26	1.26	10		0.00	15.75	15.75	10		0.00	0.15	0.15
200	0.106	56.6	59.2	10		0.00	4.35	4.35	10		0.00	156.28	156.28	10		0.00	0.16	0.16
	0.306	58.0	37.8	10		0.00	4.00	4.00	10		0.00	83.80	83.80	10		0.00	0.13	0.13
300	0.108	86.4	75.8	10		0.00	11.45	11.45	10		0.00	482.17	482.17	10		0.00	0.22	0.22
	0.298	86.1	48.3	10		0.00	18.57	18.57	10		0.00	252.91	252.91	10		0.00	0.18	0.18
400	0.101	113.8	96.3	10		0.00	28.19	28.19	8	100.00	20.00	1052.36	1081.88	10		0.00	0.21	0.21
	0.304	113.8	58.0	10		0.00	37.28	37.28	10		0.00	608.79	608.79	10		0.00	0.17	0.17
500	0.101	143.0	107.9	10		0.00	74.68	74.68	7	100.00	30.00	996.29	1057.40	10		0.00	0.17	0.17
	0.300	140.9	68.3	10		0.00	93.44	93.44	8	100.00	20.00	858.92	927.14	10		0.00	0.13	0.13
600	0.103	173.5	124.0	10		0.00	142.78	142.78	0	100.00	100.00		1200.00	10		0.00	0.36	0.36
	0.304	172.0	76.6	10		0.00	205.08	205.08	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
700	0.101	197.7	130.6	10		0.00	342.65	342.65	0	100.00	100.00		1200.00	10		0.00	0.25	0.25
	0.299	200.2	84.3	10		0.00	389.96	389.96	0	100.00	100.00		1200.00	10		0.00	0.18	0.18
800	0.103	229.0	147.4	10		0.00	566.86	566.86	0	100.00	100.00		1200.00	10		0.00	0.26	0.26
	0.304	232.3	90.6	10		0.00	583.48	583.48	0	100.00	100.00		1200.00	10		0.00	0.19	0.19
900	0.099	257.4	156.2	8	96.81	19.36	652.82	762.26	0	100.00	100.00		1200.00	10		0.00	0.29	0.29
	0.299	255.5	98.6	7	91.86	27.56	1145.15	1161.61	0	100.00	100.00		1200.00	10		0.00	0.22	0.22
1000	0.105	286.2	166.1	6	97.29	38.92	829.80	977.88	0	100.00	100.00		1200.00	10		0.00	0.32	0.32
	0.297	285.9	107.2	9	93.29	9.33	938.76	964.89	0	100.00	100.00		1200.00	10		0.00	0.23	0.23
				190	**95.17**	**4.76**	**303.57**	**318.58**	**93**	**100.00**	**53.50**	**452.65**	**834.27**	**200**	**-**	**0.00**	**0.20**	**0.20**

Table 8: Experimental results for chordal graph instances of density 0.5 and 0.7 with small clusters

				IP formulation					B&P					Decomposition				
n	avg density	avg # clust	avg # cliques	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.494	28.7	19.8	10		0.00	1.17	1.17	10		0.00	11.76	11.76	10		0.00	0.16	0.16
	0.699	28.8	15.1	10		0.00	1.25	1.25	10		0.00	11.14	11.14	10		0.00	0.15	0.15
200	0.495	57.8	28.9	10		0.00	5.16	5.16	10		0.00	69.63	69.63	10		0.00	0.14	0.14
	0.695	57.2	22.0	10		0.00	6.19	6.19	10		0.00	61.43	61.43	10		0.00	0.09	0.09
300	0.501	86.6	39.1	10		0.00	18.57	18.57	10		0.00	201.03	201.03	10		0.00	0.11	0.11
	0.706	86.7	27.0	10		0.00	23.54	23.54	10		0.00	181.26	181.26	10		0.00	0.09	0.09
400	0.499	114.8	46.1	10		0.00	41.88	41.88	10		0.00	505.76	505.76	10		0.00	0.12	0.12
	0.697	114.7	34.2	10		0.00	60.69	60.69	10		0.00	444.91	444.91	10		0.00	0.10	0.10
500	0.503	143.3	54.1	10		0.00	124.58	124.58	5	100.00	50.00	1012.02	1106.01	10		0.00	0.13	0.13
	0.705	141.7	40.6	10		0.00	141.98	141.98	0	100.00	100.00		1200.00	10		0.00	0.13	0.13
600	0.505	172.1	59.7	10		0.00	260.72	260.72	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
	0.704	172.7	44.4	10		0.00	325.45	325.45	0	100.00	100.00		1200.00	10		0.00	0.12	0.12
700	0.496	200.2	64.5	10		0.00	432.24	432.24	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
	0.704	200.1	49.1	10		0.00	695.57	695.57	0	100.00	100.00		1200.00	10		0.00	0.13	0.13
800	0.497	226.6	67.8	10		0.00	723.77	723.77	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
	0.698	229.2	50.9	7	70.21	21.06	1051.41	1095.98	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
900	0.504	255.0	74.6	2	86.32	69.06	936.59	1147.32	0	100.00	100.00		1200.00	10		0.00	0.18	0.18
	0.699	255.5	55.5	0	94.95	94.95		1200.00	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
1000	0.500	285.8	78.8	3	93.54	65.47	1577.90	1313.37	0	100.00	100.00		1200.00	10		0.00	0.19	0.19
	0.702	284.5	58.3	0	93.99	93.99		1200.00	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
				162	**90.67**	**17.23**	**357.15**	**440.97**	**85**	**100.00**	**57.50**	**277.66**	**789.65**	**200**	**-**	**0.00**	**0.14**	**0.14**

Table 9: Experimental results for chordal graph instances of density 0.1 and 0.3 with medium-sized clusters

				IP formulation					B&P					Decomposition				
n	avg density	avg # clust	avg # cliques	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.103	18.3	38.2	10		0.00	0.28	0.28	10		0.00	4.68	4.68	10		0.00	0.10	0.10
	0.299	17.7	25.0	10		0.00	0.44	0.44	10		0.00	33.71	33.71	10		0.00	0.09	0.09
200	0.106	36.9	59.2	10		0.00	1.32	1.32	10		0.00	223.93	223.93	10		0.00	0.12	0.12
	0.306	36.2	37.8	10		0.00	2.20	2.20	10		0.00	233.31	233.31	10		0.00	0.10	0.10
300	0.108	54.7	75.8	10		0.00	4.56	4.56	7	100.00	30.00	964.26	1034.98	10		0.00	0.15	0.15
	0.298	54.1	48.3	10		0.00	5.90	5.90	10		0.00	823.78	823.78	10		0.00	0.11	0.11
400	0.101	72.3	96.3	10		0.00	10.17	10.17	1	100.00	90.00	895.76	1169.58	10		0.00	0.18	0.18
	0.304	72.3	58.0	10		0.00	14.69	14.69	5	100.00	50.00	1063.91	1131.96	10		0.00	0.13	0.13
500	0.101	90.7	107.9	10		0.00	24.35	24.35	0	100.00	100.00		1200.00	10		0.00	0.20	0.20
	0.300	90.3	68.3	10		0.00	29.90	29.90	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
600	0.103	108.6	124.0	10		0.00	49.68	49.68	0	100.00	100.00		1200.00	10		0.00	0.23	0.23
	0.304	108.8	76.6	10		0.00	55.76	55.76	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
700	0.101	126.7	130.6	10		0.00	86.66	86.66	0	100.00	100.00		1200.00	10		0.00	0.23	0.23
	0.299	128.1	84.3	10		0.00	111.53	111.53	0	100.00	100.00		1200.00	10		0.00	0.18	0.18
800	0.103	146.3	147.4	10		0.00	119.76	119.76	0	100.00	100.00		1200.00	10		0.00	0.29	0.29
	0.304	144.2	90.6	10		0.00	160.96	160.96	0	100.00	100.00		1200.00	10		0.00	0.20	0.20
900	0.099	162.2	156.2	10		0.00	302.82	302.82	0	100.00	100.00		1200.00	10		0.00	0.32	0.32
	0.299	164.3	98.6	10		0.00	255.49	255.49	0	100.00	100.00		1200.00	10		0.00	0.22	0.22
1000	0.105	182.7	166.1	10		0.00	322.25	322.25	0	100.00	100.00		1200.00	10		0.00	0.34	0.34
	0.297	181.0	107.2	10		0.00	397.19	397.19	0	100.00	100.00		1200.00	10		0.00	0.25	0.25
				200	**-**	**0.00**	**97.80**	**97.80**	**63**	**100.00**	**68.50**	**530.42**	**952.80**	**200**	**-**	**0.00**	**0.19**	**0.19**

Table 10: Experimental results for chordal graph instances of density 0.5 and 0.7 with medium-sized clusters

n	avg density	avg # clust	avg # cliques	IP formulation					B&P					Decomposition				
				# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.494	18.1	19.8	10		0.00	0.55	0.55	10		0.00	32.04	32.04	10		0.00	0.08	0.08
	0.699	18.3	15.1	10		0.00	0.61	0.61	10		0.00	38.33	38.33	10		0.00	0.07	0.07
200	0.495	36.4	28.9	10		0.00	2.75	2.75	10		0.00	132.44	132.44	10		0.00	0.09	0.09
	0.695	36.9	22.0	10		0.00	3.26	3.26	10		0.00	105.92	105.92	10		0.00	0.09	0.09
300	0.501	55.7	39.1	10		0.00	9.16	9.16	10		0.00	730.09	730.09	10		0.00	0.10	0.10
	0.706	55.7	27.0	10		0.00	11.46	11.46	10		0.00	515.32	515.32	10		0.00	0.09	0.09
400	0.499	72.7	46.1	10		0.00	21.51	21.51	6	100.00	40.00	1043.58	1106.15	10		0.00	0.11	0.11
	0.697	72.8	34.2	10		0.00	29.58	29.58	0	100.00	100.00		1200.00	10		0.00	0.11	0.11
500	0.503	90.3	54.1	10		0.00	44.77	44.77	0	100.00	100.00		1200.00	10		0.00	0.13	0.13
	0.705	89.2	40.6	10		0.00	62.08	62.08	0	100.00	100.00		1200.00	10		0.00	0.11	0.11
600	0.505	109.6	59.7	10		0.00	83.57	83.57	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
	0.704	107.7	44.4	10		0.00	116.59	116.59	0	100.00	100.00		1200.00	10		0.00	0.12	0.12
700	0.496	125.9	64.5	10		0.00	140.76	140.76	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
	0.704	128.5	49.1	10		0.00	206.58	206.58	0	100.00	100.00		1200.00	10		0.00	0.13	0.13
800	0.497	145.8	67.8	10		0.00	223.10	223.10	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
	0.698	146.9	50.9	10		0.00	360.06	360.06	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
900	0.504	162.9	74.6	10		0.00	375.52	375.52	0	100.00	100.00		1200.00	10		0.00	0.18	0.18
	0.699	164.5	55.5	10		0.00	623.74	623.74	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
1000	0.500	181.9	78.8	10		0.00	568.22	568.22	0	100.00	100.00		1200.00	10		0.00	0.20	0.20
	0.702	181.7	58.3	10		0.00	1109.45	1109.45	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
				200	**-**	**0.00**	**199.67**	**199.67**	**66**	**100.00**	**67.00**	**371.10**	**913.01**	**200**	**-**	**0.00**	**0.13**	**0.13**

Table 11: Experimental results for chordal graph instances of density 0.1 and 0.3 with large clusters

				IP formulation					B&P					Decomposition				
n	avg density	avg # clust	avg # cliques	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.103	13.1	38.2	10		0.00	0.20	0.20	10		0.00	4.68	4.68	10		0.00	0.10	0.10
	0.299	13.4	25.0	10		0.00	0.37	0.37	4	100.00	60.00	147.05	778.82	10		0.00	0.09	0.09
200	0.106	26.4	59.2	10		0.00	0.80	0.80	10		0.00	38.33	38.33	10		0.00	0.13	0.13
	0.306	26.7	37.8	10		0.00	1.67	1.67	10		0.00	223.93	223.93	10		0.00	0.10	0.10
300	0.108	39.5	75.8	10		0.00	2.25	2.25	10		0.00	105.92	105.92	10		0.00	0.15	0.15
	0.298	39.2	48.3	10		0.00	4.07	4.07	7	100.00	30.00	964.26	1034.98	10		0.00	0.11	0.11
400	0.101	53.2	96.3	10		0.00	4.67	4.67	10		0.00	515.32	515.32	10		0.00	0.20	0.20
	0.304	53.1	58.0	10		0.00	10.02	10.02	1	100.00	90.00	895.76	1169.58	10		0.00	0.12	0.12
500	0.101	66.6	107.9	10		0.00	8.53	8.53	0	100.00	100.00		1200.00	10		0.00	0.19	0.19
	0.300	66.9	68.3	10		0.00	19.96	19.96	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
600	0.103	79.8	124.0	10		0.00	21.14	21.14	0	100.00	100.00		1200.00	10		0.00	0.22	0.22
	0.304	80.5	76.6	10		0.00	37.86	37.86	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
700	0.101	93.3	130.6	10		0.00	47.69	47.69	0	100.00	100.00		1200.00	10		0.00	0.25	0.25
	0.299	92.8	84.3	10		0.00	60.54	60.54	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
800	0.103	106.7	147.4	10		0.00	83.86	83.86	0	100.00	100.00		1200.00	10		0.00	0.28	0.28
	0.304	105.8	90.6	10		0.00	137.56	137.56	0	100.00	100.00		1200.00	10		0.00	0.20	0.20
900	0.099	119.8	156.2	10		0.00	134.43	134.43	0	100.00	100.00		1200.00	10		0.00	0.31	0.31
	0.299	119.0	98.6	10		0.00	158.73	158.73	0	100.00	100.00		1200.00	10		0.00	0.23	0.23
1000	0.105	133.7	166.1	10		0.00	210.71	210.71	0	100.00	100.00		1200.00	10		0.00	0.35	0.35
	0.297	133.1	107.2	10		0.00	257.12	257.12	0	100.00	100.00		1200.00	10		0.00	0.25	0.25
				200	**-**	**0.00**	**60.11**	**60.11**	**62**	**100.00**	**69.00**	**361.91**	**913.58**	**200**	**-**	**0.00**	**0.19**	**0.19**

Table 12: Experimental results for chordal graph instances of density 0.5 and 0.7 with large clusters

				IP formulation					B&P					Decomposition				
n	avg density	avg # clust	avg # cliques	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall	# opt	avg % gap in nonopt	avg % gap overall	avg time in opt	avg time overall
100	0.494	13.1	19.8	10		0.00	0.53	0.53	7	100.00	30.00	28.44	379.91	10		0.00	0.07	0.07
	0.699	13.3	15.1	10		0.00	0.69	0.69	10		0.00	32.04	32.04	10		0.00	0.08	0.08
200	0.495	26.6	28.9	10		0.00	1.99	1.99	10		0.00	233.31	233.31	10		0.00	0.09	0.09
	0.695	26.2	22.0	10		0.00	2.30	2.30	10		0.00	132.44	132.44	10		0.00	0.08	0.08
300	0.501	40.1	39.1	10		0.00	6.40	6.40	10		0.00	823.78	823.78	10		0.00	0.10	0.10
	0.706	40.0	27.0	10		0.00	11.61	11.61	10		0.00	730.09	730.09	10		0.00	0.09	0.09
400	0.499	53.4	46.1	10		0.00	15.10	15.10	5	100.00	50.00	1063.91	1131.96	10		0.00	0.12	0.12
	0.697	53.1	34.2	10		0.00	21.13	21.13	6	100.00	40.00	1043.58	1106.15	10		0.00	0.10	0.10
500	0.503	67.0	54.1	10		0.00	31.87	31.87	0	100.00	100.00		1200.00	10		0.00	0.13	0.13
	0.705	66.6	40.6	10		0.00	45.04	45.04	0	100.00	100.00		1200.00	10		0.00	0.11	0.11
600	0.505	80.5	59.7	10		0.00	60.70	60.70	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
	0.704	80.3	44.4	10		0.00	87.92	87.92	0	100.00	100.00		1200.00	10		0.00	0.12	0.12
700	0.496	93.3	64.5	10		0.00	96.09	96.09	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
	0.704	93.5	49.1	10		0.00	172.24	172.24	0	100.00	100.00		1200.00	10		0.00	0.14	0.14
800	0.497	107.7	67.8	10		0.00	199.78	199.78	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
	0.698	106.1	50.9	10		0.00	238.96	238.96	0	100.00	100.00		1200.00	10		0.00	0.15	0.15
900	0.504	119.3	74.6	10		0.00	244.41	244.41	0	100.00	100.00		1200.00	10		0.00	0.18	0.18
	0.699	120.2	55.5	10		0.00	368.76	368.76	0	100.00	100.00		1200.00	10		0.00	0.16	0.16
1000	0.500	132.4	78.8	10		0.00	397.46	397.46	0	100.00	100.00		1200.00	10		0.00	0.21	0.21
	0.702	133.4	58.3	10		0.00	626.37	626.37	0	100.00	100.00		1200.00	10		0.00	0.17	0.17
				200	**-**	**0.00**	**131.47**	**131.47**	**68**	**100.00**	**66.00**	**510.95**	**948.48**	**200**	**-**	**0.00**	**0.13**	**0.13**

improves the solvability of the problem in low-density graphs. The best performance is presented in the class of chordal graphs; it clearly outperforms regardless of the edge density or cluster size of the input graph. In terms of change in the number of instances solved to optimality as n and the average density increases, we observe a more robust performance in the class of generalized split graphs as compared to permutation graphs.

4 Conclusions and Future Research

In this paper, we presented a decomposition-based exact solution approach for the selective graph coloring problem in three different perfect graph families: permutation, generalized split, and chordal graphs. Given an input graph with a partition of its vertex set into clusters, the master problem of our decomposition procedure seeks an optimal selection, and the subproblem seeks maximum clique(s) in the graph induced by that selection by utilizing efficient combinatorial algorithms. We tested the performance of our algorithm on a large suite of randomly generated problem instances, and compared the results to those of an IP formulation and a branch-and-price algorithm from the literature. Our computational results show that the decomposition approach significantly improved the solution performance especially in low-density graphs, and the improvement manifests most evidently in the class of chordal graphs.

As future research, our decomposition approach can be adapted to more general graph classes. Namely, it would be of interest to investigate perfect graphs in general or other graph classes that are not perfect but that possess structural properties which may be utilized within this decomposition framework. The problem can also be investigated in general graphs, where the insights obtained on specific graph classes can be used to design efficient solution algorithms.

Acknowledgements

We are grateful to an anonymous referee and associate editor for their constructive comments and useful suggestions, which helped us improve the content and presentation of the paper.

References

[1] J.R. Blair and B. Peyton, "*An introduction to chordal graphs and clique trees*," *Graph Theory and Sparse Matrix Computation*, Springer, (1993), pp. 1–29.

[2] G.J. Chaitin, *Register allocation & spilling via graph coloring*, ACM SIGPLAN Notices **17** (1982), 98–101.

[3] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, *The strong perfect graph theorem*, Ann. Math. **164** (2006), 51–229.

[4] M. Demange, T. Ekim, B. Ries, and C. Tanasescu, *On some applications of the selective graph coloring problem*, Eur. J. Oper. Res. **240** (2015), 307–314.

[5] M. Demange, J. Monnot, P. Pop, and B. Ries, *On the complexity of the selective graph coloring problem in some special classes of graphs*, Theor. Comput. Sci. **540** (2014), 89–102.

[6] E. Egerváry, *On combinatorial properties of matrices, translated by H.W. Kuhn, Office of Naval Research Logistics Project Report*, Dept. Math. Princeton University (1953).

[7] E.M. Eschen and X. Wang, *Algorithms for unipolar and generalized split graphs*, Discr. Appl. Math. **162** (2014), 195–201.

[8] Y. Frota, N. Maculan, T.F. Noronha, and C.C. Ribeiro, *A branch-and-cut algorithm for partition coloring*, Networks **55** (2010), 194–204.

[9] D. Fulkerson and O. Gross, *Incidence matrices and interval graphs*, Pacific journal mathematics **15** (1965), 835–855.

[10] F. Furini, E. Malaguti, and A. Santini, *An exact algorithm for the partition coloring problem*, Comput. Oper. Res. **92** (2018), 170–181.

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

[12] F. Gavril, *The intersection graphs of subtrees in trees are exactly the chordal graphs*, J. Combinatorial Theory, Ser. B **16** (1974), 47–56.

[13] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Vol. 57, Elsevier, 2004.

[14] E.A. Hoshino, Y.A. Frota, and C.C. De Souza, *A branch-and-price approach for the partition coloring problem*, Oper. Res. Lett. **39** (2011), 132–137.

[15] D. König, *Graphen und matrizen*, Matematikai Lapok **38** (1931), 116–119.

[16] R. Lewis, *A Guide to Graph Colouring*, Springer, 2015.

[17] G. Li and R. Simha, *The partition coloring problem and its application to wavelength routing and assignment*, Proceedings of the First Workshop on Optical Networks, 2000.

[18] D. Marx, *Graph colouring problems and their applications in scheduling*, Periodica Polytechnica Electrical Eng. **48** (2004), 11–16.

[19] A. Mehrotra and M.A. Trick, *A column generation approach for graph coloring*, INFORMS J. Comput. **8** (1996), 344–354.

[20] T.F. Noronha and C.C. Ribeiro, *Routing and wavelength assignment by partition colouring*, Eur. J. Oper. Res. **171** (2006), 797–810.

[21] H.L. Petersen and O.B. Madsen, *The double travelling salesman problem with multiple stacks–formulation and heuristic solution approaches*, Eur. J. Oper. Res. **198** (2009), 139–147.

[22] H.J. Prömel and A. Steger, *Almost all Berge graphs are perfect*, Combin. Prob. Comput. **1** (1992), 53–79.

[23] O. Şeker, P. Heggernes, T. Ekim, and Z.C. Taşkın, *Linear-time generation of random chordal graphs*, Lecture Notes in Computer Science, volume 10236, Springer, 2017, pp. 442–453.

[24] H.D. Sherali and J.C. Smith, *Improving discrete model representations via symmetry considerations*, Manage. Sci. **47** (2001), 1396–1407.