Integer Programming, Constraint Programming, and Hybrid Decomposition Approaches to Discretizable Distance Geometry Problems

Moira MacNeil*[1] and Merve Bodur†[1]

[1]Department of Mechanical and Industrial Engineering, University of Toronto

October 10, 2020

Abstract

Given an integer dimension K and a simple, undirected graph G with positive edge weights, the Distance Geometry Problem (DGP) aims to find a realization function mapping each vertex to a coordinate in $\mathbb{R}^K$ such that the distance between pairs of vertex coordinates is equal to the corresponding edge weights in G. The so-called discretization assumptions reduce the search space of the realization to a finite discrete one, which can be explored via the branch-and-prune (BP) algorithm. Given a *discretization vertex order* in G, the BP algorithm constructs a binary tree where the nodes at a layer provide all possible coordinates of the vertex corresponding to that layer. The focus of this paper is finding *optimal BP trees* for a class of Discretizable DGPs. More specifically, we aim to find a discretization vertex order in G that yields a BP tree with the least number of branches. We propose an integer programming formulation and three constraint programming formulations that all significantly outperform the state-of-the-art cutting plane algorithm for this problem. Moreover, motivated by the difficulty in solving instances with a large and low density input graph, we develop two hybrid decomposition algorithms, strengthened by a set of valid inequalities, which further improve the solvability of the problem.

Keywords. Distance geometry, discretization order, integer programming, constraint programming, decomposition algorithms

1 Introduction

Distance Geometry is the study of problems where we wish to determine positions in a geometric space of points while preserving some known distances between the points [7, 14]. It has wide application areas, including astronomy, where we position stars relative to each other, and robotics, where the distances are arm lengths and we are trying to determine a set of positions within reach of a robot [7, 9, 14]. In molecular geometry, Nuclear Magnetic Resonance spectroscopy is used to find interatomic distances of large molecules, which are often proteins. This process gives measurements in two dimensions, but finding the three dimensional structure of such molecules is key for determining their functional properties. In this case, we are positioning the atoms in three-dimensional Euclidean space [7]. In wireless sensor localization, the network has components

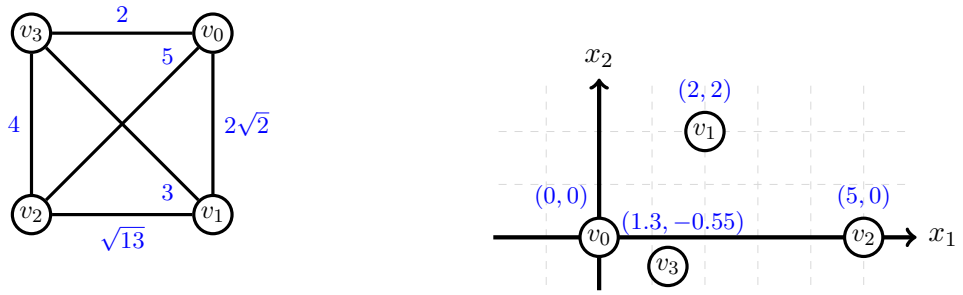
*m.macneil@mail.utoronto.ca

†bodur@mie.utoronto.ca

with fixed positions, such as routers, and we wish to determine the positions of mobile wireless sensors, such as smartphones [9]. A new variant of the Distance Geometry Problem (DGP), namely dynamical DGP, has stemmed from applications such as air traffic control, crowd simulation, multi-robot formation, and human motion retargeting, all of which involve a temporal aspect [12, 15]. Other applications include statics and graph rigidity, graph drawing, and clock synchronization [7, 9, 14].

The DGP can be represented on a graph where the vertices are the points we would like to position and weighted edges represent known distances between pairs of points, note that this graph need not be complete. In addition to the graph, DGP takes as input an integer K, the dimension of $\mathbb{R}$ into which the graph is positioned. Formally, we give the definition of [7].

Definition 1 (Distance Geometry Problem)**.** *Given, an integer $K > 0$ and a simple, undirected graph $G = (\mathcal{V}, \mathcal{E})$ with edge weights $w : \mathcal{E} \to (0, \infty)$, find a function $x : \mathcal{V} \to \mathbb{R}^K$ such that for all $\{u, v\} \in \mathcal{E}$: $\|x(u) - x(v)\| = w(u, v)$.*



(a) A complete graph with four vertices. (b) An embedding of the complete graph in $\mathbb{R}^2$.

Figure 1: A realization for a complete graph in $\mathbb{R}^2$.

The function x is called a *realization* for G or an *embedding* of G (see Figure 1). If G is not connected, determining if it has a realization is equivalent to determining if its connected components have a realization so we assume G is connected [1]. We use the Euclidean norm, however it can be any metric. We mention as well the interval DGP, where in Definition 1, the norm $\|x(u) - x(v)\|$ belongs to a given interval of weights, instead of being equal to a particular one [4, 8].

The DGP is $\mathcal{NP}$-Complete for $K = 1$ and $\mathcal{NP}$-Hard for $K > 1$ [17], and solution methods include nonlinear programming, semi-definite programming, and the geometric build-up methods [9, 12, 14]. In the special case where the distance between all pairs of vertices in G are known, that is G is complete, and we assume they yield a realization in $\mathbb{R}^K$, this realization can be found by solving a series of linear equations [7].

In most applications, the instance is not a complete graph. The distance between some pairs of vertices is not available, and so this procedure does not apply. In such a case we would like to make use of combinatorial methods to solve the DGP, thus we must establish conditions under which the solution space of the DGP can be discretized. For this, we assume that there exists a solution to the DGP for the instance. The solution set is then non-empty and thus is either finite or uncountable modulo translations, rotations, and reflections [7, 9]. The solution to the system of linear equations is now the intersection of a line segment and a sphere [6, 7, 9, 12]. The solution set is finite under the following conditions [6, 7, 9, 12, 13]:

(i) There is a realization for K vertices of the instance, and

(ii) For every other vertex, $v \in \mathcal{V}$, there exist edges $\{v,i\},\{v,j\},\{v,l\} \in \mathcal{E}$, where the i,j,l positions have already been fixed, so that we will be able to fix a position for v.

These assumptions mean if the positions of K vertices are fixed, the $(K+1)^{\text{th}}$ vertex has at most two possible positions in relation to the previously fixed vertices since we are solving for the intersection of a line segment and a sphere. Similarly, the $(K+2)^{\text{th}}$ vertex has at most four possible positions relative to the previous vertices, and so on, so that the last vertex to be placed has $2^{|\mathcal{V}|-K}$ possible positions. Thus this search space induces a binary tree structure where each layer of the tree enumerates all possible positions for a fixed vertex, where a realization is a path in the tree from the root to a leaf [12].

If there are more edges in the graph than those that satisfy (i) and (ii), it is possible to prune positions for a vertex from the solution space. This leads to the notion of the branch-and-prune (BP) algorithm, which enumerates the possible positions of vertices one-by-one and prunes a branch whenever there is an extra edge between the current vertex and previous vertices that is incompatible with the position [7, 9, 12]. We can think of an optimal BP search tree as the smallest search tree for a given instance [4]. In fact, (i) and (ii) are satisfied, and the solution space is finite only if there exists a total order on the vertices satisfying the following definition [6, 7, 9, 12]:

Definition 2 (Discretizable DGP)**.** *Given, an integer $K > 0$, and a simple, unweighted, undirected graph $G = (\mathcal{V},\mathcal{E})$, G is an instance of Discretizable DGP if there exists a total order on $\mathcal{V}$, $(v_0, v_1, \ldots, v_{|\mathcal{V}|-1})$, such that:*

(i) $G[\{v_0, v_1, \ldots, v_{K-1}\}]$ is a clique.

(ii) For all $v_i \in \{v_K, v_{K+1}, \ldots, v_{|\mathcal{V}|-1}\}$, v_i has

- *(a) at least K adjacent predecessors*
- *(b) a set of exactly K adjacent predecessors $\{v_{j_1}, v_{j_2}, \ldots, v_{j_K}\}$ where $G[\{v_{j_1}, v_{j_2}, \ldots, v_{j_K}\}]$ is a clique and the volume of the simplex formed by the realizations of $\{v_{j_1}, v_{j_2}, \ldots, v_{j_K}\}$ is positive.*

We let $G[\mathcal{V}']$ be the subgraph of G induced by $\mathcal{V}' \subseteq \mathcal{V}$ and a clique is a complete subgraph. We define an adjacent predecessor of a vertex $v \in \mathcal{V}$ as $u \in \mathcal{V}$ with $\{u,v\} \in \mathcal{E}$ such that u precedes v in the order. We note that such Discretizable DGPs (DDGPs) are feasibility problems with no objective, wherein we wish to determine only if a vertex order exists for an instance.

The focus of this paper is finding optimal BP trees for a class of DDGPs, namely the Discretization Vertex Ordering Problem (DVOP)[1], that is, we wish to find a vertex order with the smallest search tree over all possible DVOP orders. This is an $\mathcal{NP}$-Complete problem [10]. We present the DVOP in detail in Section 2, following the convention of [6], which distinguishes DVOP from DDGP and establishes the DVOP as a total order that does not verify the simplex-related conditions, (ii) (b), of the DDGP definition. The DDGP is $\mathcal{NP}$-Hard, and the DVOP is $\mathcal{NP}$-Complete [6, 7, 9]. However, if K is fixed, there exists a greedy algorithm to solve DVOP, given all possible initial cliques. Thus DVOP with fixed K is polynomial [6, 9].

The rest of the paper is organized as follows. In Section 2, we present the DVOP, and explain in detail the problem of finding an optimal discretization order, `MIN DOUBLE`. We also review two existing integer programming (IP) formulations, and a branch-and-cut procedure from the literature.

[1] In the literature, somewhat confusingly, the Discretization Vertex Ordering Problem (DVOP) is sometimes referred to as the problem of finding an order for the DDGP [1].

In Section 3, we introduce a novel IP formulation and three novel constraint programming (CP) formulations for `MIN DOUBLE`. In Section 4, we present two hybrid IP-CP decomposition algorithms, as well as some valid inequalities for the problem. Finally, in Section 5, we present a computational study.

We note that an overview of our paper, namely the models/methods from the literature as well as our proposed models/methods are provided in Table 3 of Appendix C.

2 Preliminaries

2.1 Notation

All sets are denoted calligraphically. Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. The *adjacency matrix* of G is denoted by A, i.e., $A_{v,u} = 1$ if and only if edge $\{u, v\} \in \mathcal{E}$. We define a directed graph $\overrightarrow{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{A}$ is the set of directed arcs in $\overrightarrow{G}$, i.e., $\mathcal{A} = \{(u, v) \cup (v, u) : \{u, v\} \in \mathcal{E}\}$. We adopt the convention of denoting an undirected edge, $\{u, v\}$, and a directed arc, (u, v). Denote the *neighbourhood* of a vertex v as $\mathcal{N}(v)$, i.e., $\mathcal{N}(v) = \{u \in \mathcal{V} : \{u, v\} \in \mathcal{E}\}$, thus $v \notin \mathcal{N}(v)$ and the *degree* of v as $d(v) = |\mathcal{N}(v)|$. We let $G[\mathcal{V}'] = (\mathcal{V}', \mathcal{E}')$ be the subgraph of G *induced* by $\mathcal{V}' \subseteq \mathcal{V}$, and thus $\mathcal{E}' = \{\{u, v\} \in \mathcal{E} : u, v \in \mathcal{V}'\}$. A *clique*, $\mathcal{K}$, in G is a set of vertices $\{v_1, v_2, \ldots, v_{|\mathcal{K}|}\} \subseteq \mathcal{V}$ such that $\{v_i, v_j\} \in \mathcal{E}$ for all $v_i, v_j \in \mathcal{K}$ such that $v_i \neq v_j$. We define a directed cycle, C as a subgraph of $\overrightarrow{G}$, $C = (\mathcal{V}^C, \mathcal{A}^C)$, where C forms a path where the first node is the same as the last node. We define an *adjacent predecessor* of a vertex $v \in \mathcal{V}$ as $u \in \mathcal{V}$ with $\{u, v\} \in \mathcal{E}$ such that u precedes v in a vertex order.

For $a, b \in \mathbb{Z}_+$, $a \leq b$, we use the notation $[a] = \{0, 1, \ldots, a\}$ and $[a, b] = \{a, a + 1, \ldots, b\}$. If $a > b$, then $[a, b] = \emptyset$, similarly if $a < 0$, then $[a] = \emptyset$. We use $\mathbb{I}(\cdot)$ as the *indicator function*, which evaluates to 1 if the boolean expression it operates on is true and 0 if it is false.

Indices follow these conventions: indices start at 0, so that the possible positions of a vertex order are $[|\mathcal{V}| - 1]$. We let $|\mathcal{V}| = n$, and use $|\mathcal{V}|$ in relation to vertices and n in relation to ranks of a vertex order.

2.2 Problem Definition

The *Discretization Vertex Order Problem* (DVOP) [6] is the search for a total order of the vertices of a simple, connected, undirected graph G, given an integer dimension K, that satisfies the following:

(i) the first K vertices in the order form a clique in the input graph, G, and

(ii) the following vertices each have at least K adjacent vertices in G as predecessors in the order.

We refer to a total order that satisfies (i) and (ii) as a *DVOP order*, in this case we say the instance (G, K) is *feasible*, otherwise it is *infeasible*.

In order to characterize the DVOP we introduce a general function $rank : \mathcal{V} \to [n-1]$. Let $\mathcal{R}^{LO}$ be the set of $rank(\cdot)$ that give a linear ordering, i.e., $rank(\cdot)$ is bijective [3]. Let $\mathcal{R}^{DVOP} \subseteq \mathcal{R}^{LO}$ be the set of $rank(\cdot)$ that give a linear ordering and satisfies (i) and (ii). We say $rank(\cdot)$ *characterizes* a DVOP order if $rank(\cdot) \in \mathcal{R}^{DVOP}$. Note that (ii) implies that the vertex at rank K must be adjacent to all the vertices at ranks $[K - 1]$. Combined with (i) this implies that the first $K + 1$ vertices in the DVOP order induce a clique in G, we call this clique the first or initial clique. Formally, we have:

(i) $G[\{v \in \mathcal{V} : rank(v) \leq K\}]$ is a clique, and

(ii) $|\{u \in \mathcal{N}(v) : rank(u) \leq rank(v) - 1\}| \geq K$ for all $v \in \mathcal{V}$ with $rank(v) \geq K + 1$.

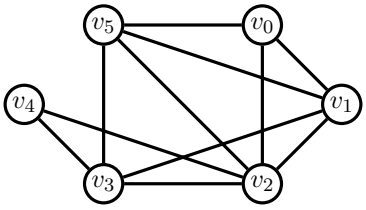


Figure 2: A graph instance which is feasible for DVOP with $K = 2$.

Example 1. *Figure 2 shows a graph for which* $(v_0, v_1, v_2, v_3, v_4, v_5)$ *is a DVOP order for* $K = 2$. *That is for some* $rank(\cdot) \in \mathcal{R}^{DVOP}$: $rank(v_0) = 0$, $rank(v_1) = 1$, $rank(v_2) = 2$ *and so on. We can see the vertices* v_0, *and* v_1 *are adjacent and thus form a 2-clique, the vertices* v_2, v_3, *and* v_4 *each have two adjacent predecessors and* v_5 *has four adjacent predecessors in the order. Note that the first* $K + 1$ *vertices in the order, namely* v_0, v_1, v_2, *form a clique since again the vertex at position* K *must be adjacent to the* K *previous vertices.*

However, there is no DVOP order for the $K = 3$ case for the graph in Figure 2, since v_4 is not in a 4-clique and cannot have 3 adjacent predecessors as it has 2 neighbours in G.

Recall, from Section 1, the solution space of DDGPs can be represented as a binary tree structure that may be searched using the branch and prune (BP) algorithm. Here the vertex order dictates the manner in which we search over the continuous $\mathbb{R}^K$ space to find a solution to the DGP. Given a DVOP order, the BP algorithm solves the DDGP by fixing the coordinates of the first K vertices in the order and enumerating the possible realizations of the remaining vertices. In the BP search tree, branching on a vertex with exactly K predecessors in the order yields at most two child nodes which are called *double vertices*. Otherwise if the vertex has more than K predecessors it has at most one child in the BP tree and is a *single vertex* [16]. In order to characterize the notion of a double vertex we define function which we call $double(\cdot)$ as

$$double(v) = \begin{cases} 1, & \text{if } v \text{ has exactly } K \text{ adjacent predecessors in the order} \\ 0, & \text{otherwise} \end{cases} \qquad \forall\ v \in \mathcal{V}.$$

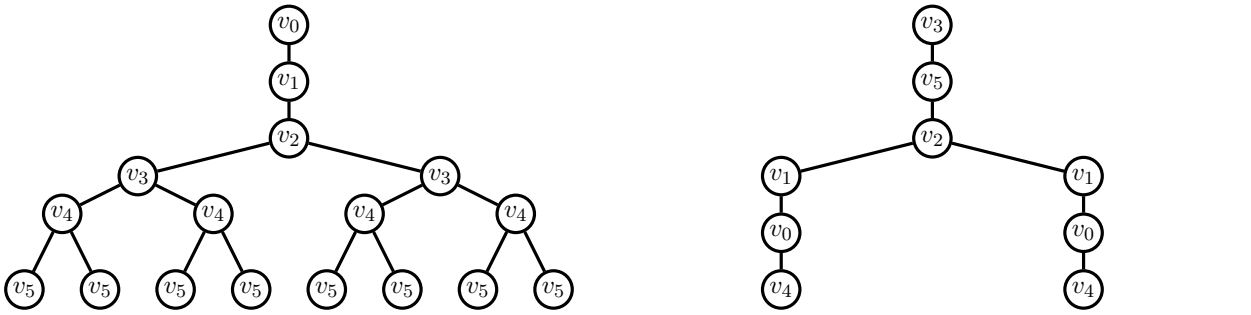
The number of double vertices can be used as a measure of the size of the BP tree and defined by the recursion,

$$nodes(r) = \begin{cases} 1, & r \in [K-1] \\ (double(rank^{-1}(r)) + 1) \cdot nodes(r-1), & r \in [K, n-1] \end{cases}$$

where $nodes(r)$ is the number of nodes at level r of the BP tree [16]. Notice that because each level of the tree gives all positions of a single vertex in the order, the levels of the tree are equivalent to the positions of the order. The maximum number of nodes in the BP tree is the sum of $nodes(\cdot)$ over all positions. Note that we say the *maximum* number of nodes in the BP tree, not the number of nodes, since G may have extra edges that allow us to prune positions and reduce the number of nodes in the tree.

Given this measure of BP tree size, [16] defined two optimization problems: `MIN DOUBLE` and `MIN NODES`, both of which try to find an optimal DVOP order but under different objective functions. `MIN DOUBLE` seeks to minimize the number of nodes which are doubles, while `MIN NODES`

seeks to minimize the maximum number of nodes in the BP tree and thus requires both a minimum number of double vertices and to fix the positions of these vertices as close to the end of the order as possible so as to have the smallest effect on tree size, due to doubling the number of nodes.



(a) A BP tree for DVOP order $(v_0, v_1, v_2, v_3, v_4, v_5)$ of the graph in Figure 2.

(b) A BP tree for DVOP order $(v_3, v_5, v_2, v_1, v_0, v_4)$ of the graph in Figure 2.

Figure 3: BP trees for two DVOP orders of the graph in Figure 2.

Example 2. *The order previously given for the graph in Figure 2, $(v_0, v_1, v_2, v_3, v_4, v_5)$, has three double vertices, v_2, v_3, v_4, which all have exactly K adjacent predecessors in the order. However, a DVOP order that minimizes the number of doubles for this instance is $(v_3, v_5, v_2, v_1, v_0, v_4)$, which has two double vertices, v_2, v_4. Figure 3a shows the BP tree for the first order, with three doubles, it has at most 17 nodes in total. Recall, that each node of the BP tree is a possible coordinate position for that vertex in $\mathbb{R}^K$. Figure 3b shows the BP tree for the second order, with two doubles, and at most 9 nodes almost half as many as the BP tree of first order. We also note that the tree in Figure 3a has width 8, whereas the tree in Figure 3b has width 2. Thus fewer doubles reduced the size of the BP tree both with respect to the width and the number of nodes, giving a clear motivation to solve the aforementioned optimization problems.*

The general framework for `MIN DOUBLE` is

$$\min \sum_{r\in[n-1]} double(rank^{-1}(r)) \tag{1a}$$

$$\text{s.t. } rank(\cdot) \in \mathcal{R}^{DVOP} \tag{1b}$$

$$\sum_{u\in\mathcal{N}(v)} \mathbb{I}(rank(u) \le rank(v)-1) \begin{cases} = K & \Rightarrow double(v) = 1 \\ \ge K+1 & \Rightarrow double(v) = 0 \end{cases} \quad \forall\, v \in \mathcal{V} : rank(v) \ge K \tag{1c}$$

Constraints (1c) can be simplified since at optimality the objective will ensure there are the fewest doubles possible, thus we need only to enforce that v has at least $K+1$ adjacent predecessors if it is not a double. Otherwise, we need to enforce that v has a least K adjacent predecessors. Thus, we continue with the following reduced version of (1):

$$\min \sum_{r\in[n-1]} double(rank^{-1}(r)) \tag{2a}$$

$$\text{s.t. } rank(\cdot) \in \mathcal{R}^{DVOP} \tag{2b}$$

$$\sum_{u\in\mathcal{N}(v)} \mathbb{I}(rank(u) \le rank(v)-1) \ge K+1-double(v) \quad \forall\, v \in \mathcal{V} \text{ s.t. } rank(v) \ge K \tag{2c}$$

Note that constraints (2c) imply that every vertex v with $rank(v) \geq K$ has at least K adjacent predecessors since $double(v) \in \{0, 1\}$. Moreover, if v also has $\leq K$, i.e., $= K$, adjacent predecessors, then constraints (2c) makes it a double. Otherwise, constraint (2c) for v becomes redundant, so the value of $double(v)$ can be either 0 or 1. But due to the objective, which is minimizing the number of doubles, we will have $double(v) = 0$, which is what is desired by definition of $double(\cdot)$.

We are able to extend the general formulation for `MIN DOUBLE` to one for `MIN NODES` by simply changing the objective function to $\sum_{v \in \mathcal{V}} nodes(rank(v))$ since the number of nodes in the tree, $nodes(rank(v))$, at each rank follows by definition from whether the vertex at that position in the order is a double or not, $double(v)$. Furthermore, we consider the multi-objective case of minimizing both the number of doubles and the maximum number of nodes. We define the multi-objective problem as follows.

$$\min \left\{ \sum_{v \in \mathcal{V}} nodes(rank(v)), \sum_{r \in [n-1]} double(rank^{-1}(r)) \right\} \text{ s.t. (2b)} - \text{(2c)} \tag{3}$$

Example 3. *Consider the graph in Figure 2. For this example, the multi-objective problem has 180 feasible (DVOP) solutions in the decision space, which yield the set of five images in the objective space as shown in Figure 4. Its Pareto frontier consists of the square point.*

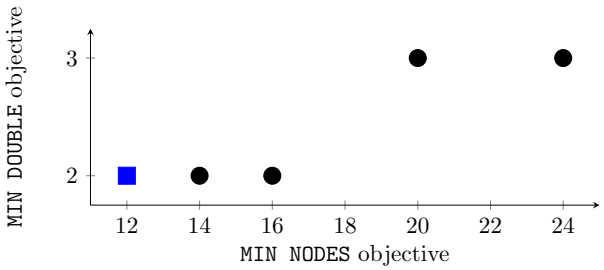


Figure 4: Set of feasible solutions in the objective space of the multi-objective problem (3) for the example in Figure 2, where the non-dominated point is marked with a square.

We conjecture that the `MIN NODES` $-$ `MIN DOUBLE` multi-objective problem has a feasible ideal point. If Conjecture 1 is true we can solve `MIN DOUBLE` and retrieve from its solution an optimal solution to `MIN NODES`.

Conjecture 1. *The Pareto frontier of the `MIN NODES` $-$ `MIN DOUBLE` multi-objective problem (3) consists of a single non-dominated point.*

For the remainder of this paper, we study solution methods to `MIN DOUBLE`, but these methods can be easily extended to `MIN NODES` by a couple simple modifications to the proposed models (more explicitly by defining additional variables for $nodes(\cdot)$ values and linking them with the variables describing the DVOP order, leaving the subproblems and cuts for the decompositions models intact). An example of an IP model for `MIN NODES` can be found in Appendix A.

2.3 Existing Mathematical Models

Prior to this work, [16] present two IP formulations and one branch-and-cut procedure for `MIN DOUBLE`. These are summarized below, while full details can be found in Appendix B.

- The cycles formulation ($\mathbb{CYCLES}$): They introduce three sets of binary variables to indicate precedence between vertices, the double vertices, and the initial clique, respectively. The constraints break 2-cycles and 3-cycles in the precedence variables, select the first clique, and link precedence and the initial clique variables to the doubles.
- The rank formulation ($\mathbb{RANKS}$): They replace the cycle breaking constraints in ($\mathbb{CYCLES}$) with an adaptation of the [11] formulation for the travelling salesman problem, for which they introduce rank variables, and link the precedence variables with the ranks.
- Cycle cut generation ($\mathbb{CCG}$): They break cycles in the precedence variables iteratively within a branch-and-cut procedure.

3 Mathematical Models

To formulate the mathematical models we define binary variables $y_r = 1$ if the vertex at position $r \in [n-1]$ is a double, 0 otherwise. It is also possible to define this variable with respect to the vertex index $v \in \mathcal{V}$ as in the existing IP formulations. When the rank-based double variables are used, we fix the first K positions to be single vertices, as by definition they cannot be doubles. We also fix position K to be a double, since we have an initial clique of size $K+1$, which implies that the $(K+1)^{\text{th}}$ vertex will always be adjacent to exactly K vertices and thus is always double. For clarity we write these as constraints in the formulations, however they are implemented as bounds and they may be omitted by projecting out the y_r for $r \in [K]$ and adding one to the objective function.

3.1 Integer Programming Formulation

Our first IP formulation extends the DVOP formulation presented by [6] to incorporate constraints for `MIN DOUBLE`. Define binary variables $x_{vr} = 1$ if the vertex $v \in \mathcal{V}$ is at rank $r \in [n-1]$, 0 otherwise. We also introduce binary indicator variables z_{vr} for all $v \in \mathcal{V},\ r \in [n-1]$ to express logical constraints. Then, the formulation is:

$$(\mathbb{IP}) : \min \sum_{r \in [n-1]} y_r \tag{4a}$$

$$\text{s.t.} \sum_{r \in [n-1]} x_{vr} = 1 \qquad \forall\ v \in \mathcal{V} \tag{4b}$$

$$\sum_{v \in V} x_{vr} = 1 \qquad \forall\ r \in [n-1] \tag{4c}$$

$$\sum_{u \in \mathcal{N}(v)} \sum_{j \in [r-1]} x_{uj} \geq r x_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [1, K] \tag{4d}$$

$$\sum_{u \in \mathcal{N}(v)} \sum_{j \in [r-1]} x_{uj} \geq K x_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K+1, n-1] \tag{4e}$$

$$y_r = 0 \qquad \forall\ r \in [K-1] \tag{4f}$$

$$y_K = 1 \tag{4g}$$

$$\sum_{u \in \mathcal{N}(v)} \sum_{j \in [r-1]} x_{uj} \geq (K+1) z_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K, n-1] \tag{4h}$$

$$x_{vr} - y_r \leq z_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K, n-1] \tag{4i}$$

$$x \in \{0,1\}^{|\mathcal{V}|\times n}, y \in \{0,1\}^n, z \in \{0,1\}^{|\mathcal{V}|\times n} \tag{4j}$$

Constraints (4b) and (4c) ensure that there is a bijection from the vertices to the ranks, that is each vertex has exactly one rank and vice versa. Constraints (4d) ensure that we have an initial clique of size $K+1$, since each vertex in the clique will be adjacent to all its predecessors. Constraints (4e) ensure that each vertex after the initial clique has at least K adjacent predecessors. Constraints (4f) and (4g) fix the double value of positions $[K]$. Constraints (4h) and (4i) link the x_{vr} and y_v variables, where we want to enforce that if vertex v is a non-double, then for any rank r, either v is not in position r or it has at least $K+1$ adjacent predecessors, that is $y_r = 0 \wedge x_{vr} = 1 \implies \sum_{u\in\mathcal{N}(v)} \sum_{j\in[r-1]} x_{uj} \geq K+1$. The left-hand-side can be rewritten giving $x_{vr} - y_r \geq 1 \implies \sum_{u\in\mathcal{N}(v)} \sum_{j\in[r-1]} x_{uj} \geq K+1$. We then use the indicator variable z_{vr} taking the value of 1 if the left-hand-side of the inequality holds to rewrite the implication as constraints (4h) and (4i).

For any feasible solution to ($\mathbb{IP}$), $rank(v) = \sum_{r\in[r-1]} r x_{vr}, \forall\ v \in \mathcal{V}$ characterizes the order, while $double(rank^{-1}(r)) = y_r, \forall\ r \in [n-1]$ provides the information about the doubles.

3.2 Constraint Programming Formulations

The effectiveness of Constraint Programming (CP) for solving permutation-based problems can be leveraged to solve `MIN DOUBLE`. We present three CP formulations for `MIN DOUBLE`.

We begin with a natural translation of ($\mathbb{IP}$) into CP to define the primal CP formulation. Define integer variables $r_v \in [n-1]$ denoting the rank of vertex $v \in \mathcal{V}$.

$$(\mathbb{CP}^{\mathbb{RANK}}) : \min \sum_{v\in\mathcal{V}} y_v + 1 \tag{5a}$$

$$\text{s.t. AllDifferent}(r_0, r_1, ..., r_{n-1}) \tag{5b}$$

$$r_i \geq K+1 \vee r_j \geq K+1 \qquad \forall\ i,j \in \mathcal{V} : i \neq j, \{i,j\} \notin \mathcal{E} \tag{5c}$$

$$r_v \geq K+1 \Rightarrow \sum_{u\in\mathcal{N}(v)} \mathbb{I}(r_u \leq r_v - 1) \geq K + (1 - y_v) \quad \forall\ v \in \mathcal{V} \tag{5d}$$

$$y \in \{0,1\}^n, r \in [n-1]^{|\mathcal{V}|} \tag{5e}$$

Objective (5a) minimizes the number of vertices which are double and adds one for the vertex in position K which is always a double by definition. Constraint (5b) is the CP global constraint AllDifferent, which forces every variable in the set to have a unique value, because this constraint acts on n variables, all of which have the same domain which also has n values, (6b) ensures that each vertex and rank map one to one. Constraints (5c) constrain that we have an initial clique of size $K+1$, by having only pairs of vertices which are adjacent in the first $K+1$ ranks. Logical constraints (5d) ensure that all vertices with ranks outside the initial clique have K adjacent predecessors if they are double and at least $K+1$ adjacent predecessors otherwise. These constraints use CP cardinality clause constraints, which specify a particular number of boolean variables must be true. However, constraints (5d) do not use boolean variables directly, instead each $(r_u \leq r_v - 1)$ is a predicate taking a boolean value.

The function which characterizes the order for ($\mathbb{CP}^{\mathbb{RANK}}$) is $rank(v) = r_v, \forall\ v \in \mathcal{V}$ and the double function is $double(v) = y_v, \forall\ v \in \mathcal{V}$.

The second CP formulation for `MIN DOUBLE` is the dual formulation to (5). We define integer variables v_r equal to the vertex index at rank $r \in [n-1]$. As such, the value of these dual variables are equivalent to the primal variables in (5) [18]. We note many of the constraints of this formulation

depend on the entries of the adjacency matrix, A.

$$(\mathbb{CP}^{\text{VERTEX}}): \min \sum_{r\in[n-1]} y_r \tag{6a}$$

$$\text{s.t. AllDifferent}(v_0, v_1, ..., v_{n-1}) \tag{6b}$$

$$A_{v_i,v_j} = 1 \qquad \forall\ i \in [K-1],\ j \in [i+1, K] \tag{6c}$$

$$\sum_{j=0}^{r-1} A_{v_r,v_j} \geq K + (1 - y_r) \qquad \forall\ r \in [K+1, n-1] \tag{6d}$$

$$y_r = 0 \qquad \forall\ r \in [K-1] \tag{6e}$$

$$y_K = 1 \tag{6f}$$

$$y \in \{0,1\}^n, v \in [|\mathcal{V}| - 1]^n \tag{6g}$$

Objective (6a) minimizes the number of ranks with double variables. Constraint (6b) enforces the one to one mapping of ranks to vertices. Constraints (6c) ensure that we have an initial clique of size $K+1$ by forcing all vertices in the first $K+1$ ranks to all be pairwise adjacent. These constraints use the so-called element constraints, which allow the use of variables as array indices in CP. Constraints (6d) force all vertices in ranks greater than K to have at least K adjacent predecessors if the are double and at least $K+1$ adjacent predecessors if they are not. Constraints (6e) and (6c) are the same as constraints (4f) and (4g) which allow fixing of the variables whose double values are known.

For $(\mathbb{CP}^{\text{VERTEX}})$, $rank^{-1}(r) = v_r, \forall\ r \in [n-1]$ characterizes the DVOP order and the double function is $double(rank^{-1}(r)) = y_r, \forall\ r \in [n-1]$. The third CP formulation uses both primal and dual variables and constraints, creating one larger combined formulation which can leverage redundant constraints to make stronger inferences in the CP search.

$$(\mathbb{CP}^{\text{COMBINED}}): \min \sum_{r\in[n-1]} y_r \tag{7a}$$

$$\text{s.t. inverse}(v, r) \tag{7b}$$

$$A_{v_i,v_j} = 1 \qquad \forall\ i \in [K-1],\ j \in [i+1, K] \tag{7c}$$

$$\sum_{j=0}^{r-1} A_{v_r,v_j} \geq K + (1 - y_r) \qquad \forall\ r \in [K+1, n-1] \tag{7d}$$

$$y_r = 0 \qquad \forall\ r \in [K-1] \tag{7e}$$

$$y_K = 1 \tag{7f}$$

$$r_i \geq K+1 \vee r_j \geq K+1 \qquad \forall\ i,j \in \mathcal{V} : i \neq j, \{i,j\} \notin \mathcal{E} \tag{7g}$$

$$r_v \geq K+1 \Rightarrow \sum_{u\in\mathcal{N}(v)} \mathbb{I}(r_u \leq r_v - 1) \geq K \qquad \forall\ v \in \mathcal{V} \tag{7h}$$

$$y \in \{0,1\}^n, v \in [|\mathcal{V}| - 1]^n, r \in [n-1]^{|\mathcal{V}|} \tag{7i}$$

Objective (7a) minimizes the number of ranks which have double vertices. Constraint (7b) channels the primal and dual variables by enforcing the relationship $(r_i = j) \equiv (v_j = i)$. This inverse constraint allows for the elimination of the AllDifferent constraints because the domains of both types of variables are the same. Constraints (7h) do not include the double variables as we have chosen to index the doubles in the natural way using ranks instead of vertices. All the other constraints are as defined in formulations (5) and (6).

For ($\mathbb{CP}^{\mathbb{COMBINED}}$), both $rank(v) = r_v, \forall\ v \in \mathcal{V}$ and $rank^{-1}(r) = v_r, \forall\ r \in [n-1]$ characterize the DVOP order, while $double(rank^{-1}(r)) = y_r, \forall\ r \in [n-1]$ is the double function.

4 Hybrid Decomposition Approaches and Valid Inequalities

4.1 A Naive Decomposition

We present a decomposition of ($\mathbb{CP}^{\mathbb{VERTEX}}$), where the master problem will be solved using IP. The master problem, ($\mathbb{MP1}$), provided in (8), fixes which positions of the order have double vertices. As such, since the double variables are rank-indexed we are able to fix the values of $K+1$ variables (which is not possible with vertex-indexed double variables).

$$(\mathbb{MP1}):\ \min\left\{\sum_{r\in[n-1]} y_r : y_r = 0\ \forall\ r \in [K-1],\ y_K = 1,\ y \in \{0,1\}^n\right\} \tag{8}$$

Given a feasible solution to the master problem, $\hat{y}$, the subproblem then tries to build a DVOP order with doubles in the correct location.

$$(\mathbb{SP1}): \text{AllDifferent}(v_0, v_1, ..., v_{n-1}) \tag{9a}$$
$$A_{v_i,v_j} = 1 \qquad \forall\ i \in [K-1],\ j \in [i+1, K], \tag{9b}$$
$$\sum_{j=0}^{r-1} A_{v_r,v_j} \geq K \qquad \forall\ r \in [K+1, n-1] \text{ with } \hat{y}_r = 1 \tag{9c}$$
$$\sum_{j=0}^{r-1} A_{v_r,v_j} \geq K+1 \qquad \forall\ r \in [K+1, n-1] \text{ with } \hat{y}_r = 0 \tag{9d}$$
$$v \in [|\mathcal{V}|-1]^n \tag{9e}$$

A feasible subproblem means we have found an optimal solution since ($\mathbb{MP1}$) will give the smallest number of doubles and ($\mathbb{SP1}$) ensures that there is in fact an order with this number of doubles. If ($\mathbb{SP1}$) is infeasible, that means there is no order with the $\hat{y}$ sequence and we must cut off the candidate $\hat{y}$ solution.

The simplest cut for ($\mathbb{MP1}$) is a no-good cut which will force it to choose a different position for at least one double vertex or single vertex:

$$\sum_{\substack{r\in[n-1]:\\ \hat{y}_r=1}} (1 - y_r) + \sum_{\substack{r\in[n-1]:\\ \hat{y}_r=0}} y_r \geq 1.$$

However, as this cut eliminates only the $\hat{y}$ solution, it is very weak.

Fortunately, this naive decomposition exactly fits into the definition of *combinatorial Benders decomposition* [2], since we have one set of constraints in the extensive form ($\mathbb{CP}^{\mathbb{VERTEX}}$), namely (6d) that links the y and r variables, and there is exactly one y in each of the constraints. If ($\mathbb{SP1}$) is infeasible we can solve for an irreducible infeasible subsystem (IIS), let $\mathcal{IIS} \subseteq [K+1, n-1]$, and pass the following cut to ($\mathbb{MP1}$) instead of the no-good cut:

$$\sum_{\substack{r\in[n-1]:\\ \hat{y}_r=0\\ r\in\mathcal{IIS}}} y_r \geq 1 \tag{10}$$

As we are minimizing the number of doubles, we need to only consider those positions in which the master problem selects non-doubles, namely where $\hat{y}_r = 0$.

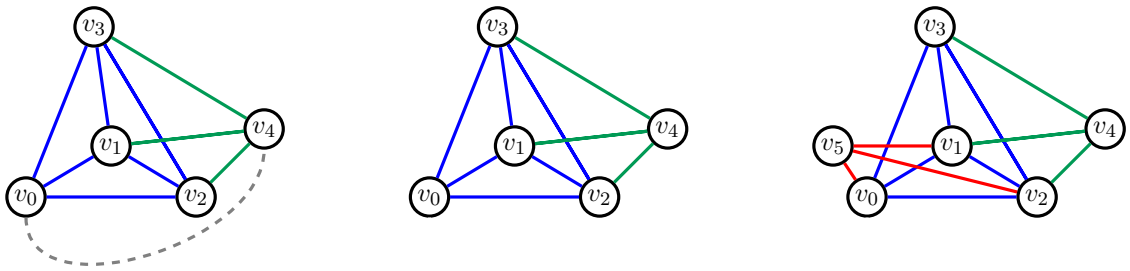
Algorithm 1: Naive Decomposition

```
1  optimal := false
2  while not optimal do
3  |  solve (𝕄ℙ𝟙) and get solution (ŷ), solve (𝕊ℙ𝟙) with (ŷ)
4  |  if (𝕊ℙ𝟙) infeasible then
5  |  |  find ℐℐ𝒮, add (10) to (𝕄ℙ𝟙)
6  |  else
7  |  |  let v̂ be an optimal solution of (𝕊ℙ𝟙)
8  |  |  accept (ŷ, v̂), optimal := true
9  |  end
10 end
```

4.2 Valid Inequalities

Analysis of the preliminary computational results showed that the doubles in an order were likely to be close to the first clique, which is logical as K is small compared to the number of vertices and these are the vertices with the least predecessors. The valid inequalities obtained via Algorithm 2 are the result of analyzing the structure of graphs with various double sequences. Given an instance (G, K) and the set of cliques, $\mathcal{K}$, in G, Algorithm 2 attempts to find structures for which positions $K+1, K+2$, and $K+3$ in the order can be non-doubles. We note that these valid inequalities can be added to any other formulation which use rank-based double variables. We next explain the procedure in Algorithm 2, step by step.

The algorithm begins by checking if there is a clique of size $K+2$ in G. If such a clique does not exist, the vertex at position $K+1$ must be a double since we are unable to extended the initial $(K+1)$-clique to a $(K+2)$-clique. Thus the vertex at position $K+1$ cannot be adjacent to all $K+1$ of its predecessors and must have exactly K predecessors instead. In this case we obtain $y_{K+1} = 1$ as a valid inequality. As seen in Figure 5a, if the dashed edge does not exists, there is no $(K+2)$-clique, so the initial clique can be v_0, v_1, v_2, v_3 and v_4 is a double or the initial clique can be v_1, v_2, v_3, v_4 and v_0 is double, thus the vertex at position $K+1$ is always a double.



(a) Structure for the first $K+1$ ranks, there is no $(K+2)$-clique if the dashed edge is not in the input graph.

(b) Two overlapping $(K+1)$-cliques with intersection of exactly K vertices. Here y_{K+1} is a double.

(c) Three overlapping $(K+1)$-cliques with intersection of exactly K vertices. Here y_{K+1} and y_{K+2} are doubles.

Figure 5: Graph substructures which lead to valid inequalities for $(\mathbb{MP}\mathbb{1})$, with $K = 3$.

If y_{K+1} has been fixed to be a double, we search further for a structure in the graph that would allow position $K+2$ to be a non-double. On line 5, we consider all possible graph sub-structures which can fill positions 0 to $K+1$ in the order. Since there is no $(K+2)$-clique, these structures are

two $(K+1)$-cliques, whose intersection is exactly K vertices, as seen in Figure 5b, one clique is the initial clique and the second contains the vertex at position $K+1$ and its K adjacent predecessors. We would like to extend these structures by another vertex which is adjacent to at least $K+1$ of the vertices in the structure. If such a vertex exists then we have a candidate structure for which position $K+2$ can be a non-double, if no such candidate is found, we know y_{K+2} is a double, this is the case in Figure 5c, where we have three cliques $(K+1)$-cliques, whose intersection is exactly K vertices but no $(K+2)$-clique as a substructure. Finally if we have at least one candidate we try to extend it in the same manner as before by looking for a vertex which could be a non-double in position $K+3$, if one exits we cannot say anything about the whether position $K+3$ has a double. However, if no such structure exists, we can add the valid inequality $y_{K+2} + y_{K+3} \geq 1$ (line 22), since we know positions $K+2$ and $K+3$ are not both non-double.

If there is a $(K+2)$-clique (line 25), it is possible for position $K+1$ to be a non-double, so we similarly search for a vertex which can extend one of the $(K+2)$-cliques so that the vertex is a non-double. If no such vertex exists, we know positions $K+1$ and $K+2$ are not both non-double, so we can add the valid inequality $y_{K+1} + y_{K+2} \geq 1$ (line 34). Finally, if we have one such vertex clique candidate, we can try to extend it a final time to find a structure which would allow all three positions, $K+1, K+2$, and $K+3$, to be non-doubles. If we are unable to find such a candidate structure, we add the valid inequality $y_{K+1} + y_{K+2} + y_{K+3} \geq 1$ (line 44) as we need at least one double vertex in these positions.

We are also able to fix the double value of positions starting from the end of the order. We remark that if there is no vertex with degree exactly K, then the last position in the order cannot be a double because it must have at least $K+1$ adjacent predecessors. Similarly, if there is no vertex with degree greater than or equal to $K+1$, positions $n-1$ and $n-2$ cannot be doubles because the vertex with smallest degree has at least $K+2$ neighbours so in the best case if it is in position $n-1$, it has $K+2$ adjacent predecessors, and if it is in position $n-2$, it has at least $K+1$ adjacent predecessors. We extend this notion to a general rule based on the minimum degree of a vertex in the input graph.

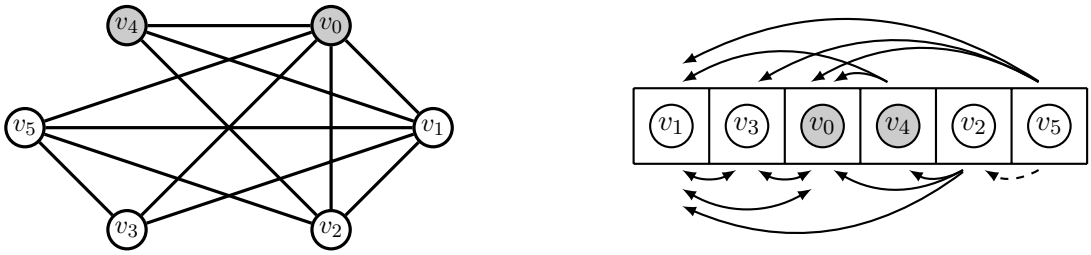
Variable Fixing Rule 1. *Given, (G, K), let m be the minimum degree of $v \in \mathcal{V}$. Then, we can fix $y_{n-i} = 0 \quad \forall\ i = [1, m-K]$.*

Preliminary computational results show that the performance of the naive decomposition is weak. The master problem solves in less than a second, and using combinatorial Benders, we were able to observe that the problem converges in relatively few iterations, however finding an IIS requires a considerable amount of time. Thus this naive decomposition approach is unbalanced, it has a very weak master problem and a very strong subproblem. This motivates the next decomposition approach. We remark, however, that the addition of valid inequalities and variable fixing strengthens the naive decomposition significantly.

4.3 A Witness-based Decomposition and an Extended Formulation

Ideally, our decomposition would be more balanced; the master problem would make decisions about the doubles but also about some aspects of the vertex ordering. However, since our current problem space only has two types of decisions, namely the order and the doubles, we are unable to decompose any further. Thus, to help improve the decomposition we add more decisions. First, we allow the problem to reason directly about which vertices will be in the initial clique. We would also like to add some decisions that restrict the space of vertex orders but is less restrictive than the linear ordering constraints used in ($\mathbb{CYCLES}$), to do so we introduce the idea of a *witness*. We say a vertex u witnesses vertex v in a given DVOP order if u supports the validity of the position of

v. More specifically, the witnesses of v are a minimal set of adjacent predecessors of v so that each double vertex has exactly K witnesses, and each non-double vertex has exactly $K+1$ witnesses. We also make the convention that all the vertices of the initial $(K+1)$-clique witness each other and all of their neighbours. The latter means that if a vertex, v, outside the initial clique is adjacent to a vertex u inside the initial clique, then u will always be a witness to v.



(a) A feasible DVOP order $(v_1, v_3, v_0, v_4, v_2, v_5)$, with $K = 2$.

(b) The witness graph.

Figure 6: An instance with a DVOP order, and its witness graph where there is a solid arc (v, u) if u witnesses v, while there is a dashed arc (v, u) if u is adjacent to v in the input graph but is not selected to witness v. (The double vertices in the order are highlighted in gray.)

For example, when $K = 2$, the graph in Figure 6a has DVOP order $(v_1, v_3, v_0, v_4, v_2, v_5)$ with two doubles, namely v_0 and v_4. Since the vertices in the first clique witness each other we have v_1 is witnessed by v_3 and v_0, v_3 is witnessed by v_1 and v_0, and v_0 is witnessed by v_3 and v_1. Vertex v_4 is a double outside the initial clique and thus has $K = 2$ adjacent predecessors, meaning it needs K witnesses, v_4 is witnessed by v_0 and v_1. Vertex v_2 is not a double, so it must have at least $K+1$ adjacent predecessors, meaning it needs $K+1$ witnesses. Since v_2 has exactly $K+1$ adjacent predecessors, it can only be witnessed by them, i.e., v_0, v_1 and v_4 are all witnesses for v_2. Finally, vertex v_5 is not a double so it needs $K+1$ witnesses. However, v_5 has four adjacent predecessors, due to our convention, we choose the initial clique vertices v_1, v_3, and v_0 to witness v_5, although any combination of three vertices from v_1, v_3, v_0 and v_2 would be a valid choice for the witness set of v_5. The relationship between vertices and their witnesses can be seen in Figure 6b.

In our new decomposition, the master problem decides which vertices are doubles and which are in the initial clique, as well as an appropriate number of witnesses for each vertex. The subproblem then looks for an order that respects the clique and witness solutions. For the master problem, in addition to double variables, y_v, we introduce binary variables $\kappa_v = 1$ if vertex $v \in \mathcal{V}$ is in the initial $(K+1)$-clique, and $w_{vu} = 1 \ \forall \ v \in \mathcal{V}, u \in \mathcal{N}(v)$ if v uses u as a witness, i.e., if u is a witness for v. The κ fix the vertices in the initial clique, and the w enforce the required number of witnesses for each vertex, depending on y. This formulation incorporates some of the ideas from linear ordering seen in [16], including defining ($2|\mathcal{E}|$-many) witness variables on the edges of the graph. The key difference between the two formulations is the treatment of the initial clique. Whereas [16] enumerate the possible first cliques of the order a priori, we constrain the vertices in these positions to form a clique. Further comparisons of the similarities and differences are presented in Appendix D. The new master problem is defined below:

$$(\mathbb{MP}2) : \min \sum_{v \in \mathcal{V}} y_v + 1 \tag{11a}$$

$$\text{s.t.} \sum_{v \in \mathcal{V}} \kappa_v = K + 1 \tag{11b}$$

$$\kappa_v + \kappa_u \leq 1 \qquad \forall\, v, u \in \mathcal{V} : u \neq v, \{v, u\} \notin \mathcal{E} \tag{11c}$$

$$\kappa_v \leq w_{uv} \qquad \forall\, v \in \mathcal{V}, u \in \mathcal{N}(v) \tag{11d}$$

$$\sum_{u \in \mathcal{N}(v)} w_{vu} = (K+1)(1-\kappa_v) - y_v + K\kappa_v \qquad \forall\, v \in \mathcal{V} \tag{11e}$$

$$y \in \{0,1\}^{|\mathcal{V}|}, \kappa \in \{0,1\}^{|\mathcal{V}|}, w \in \{0,1\}^{2|\mathcal{E}|} \tag{11f}$$

Constraint (11b) ensures that exactly $K+1$ vertices are selected for the initial $(K+1)$-clique and constraints (11c) ensure if two vertices are not adjacent, they cannot both be in the initial clique. Constraints (11d) ensure a vertex will always be witnessed by its neighbours in the initial clique. Constraints (11e) link the double variables, the clique variables and the witness variables. If a vertex v is a double, $y_v = 1$, and outside the initial clique, $\kappa_v = 0$, it must have K witnesses over all its neighbours, given by the first and second terms of the right-hand-side. If v is not a double, $y_v = 0$, but still outside the initial clique, $\kappa_v = 0$, the number of witnesses it requires is $K+1$, since all but the first term cancel. If v is in the initial clique, $\kappa_v = 1$ it must be witnessed by all other vertices in the initial clique, by (11d), which has size $K+1$ so v has K witnesses, implying $y_v = 0$. Finally, objective function (11a) minimizes the number of doubles vertices but also accounts for the linking constraint setting the vertex in position K to be a non-double when it is always a double by adding 1 to the objective.

The subproblem can be any $rank(\cdot)$ as long as it is properly linked to ($\mathbb{MP}2$) variables. We use ($\mathbb{CP}^{\mathbb{RANK}}$) constraints in the subproblem. Given an ($\mathbb{MP}2$) candidate solution $(\hat{\kappa}, \hat{w}, \hat{y})$, the subproblem looks for a consistent DVOP order:

$$(\mathbb{SP}2) : \text{AllDifferent}(r_0, r_1, ..., r_{n-1}) \tag{12a}$$

$$\hat{\kappa}_v = 1 \implies r_v \leq K \qquad \forall\, v \in \mathcal{V} \tag{12b}$$

$$\hat{\kappa}_v + \hat{\kappa}_u = 0 \text{ and } \hat{w}_{vu} = 1 \implies r_u \leq r_v - 1 \qquad \forall\, v \in \mathcal{V}, u \in \mathcal{N}(v) \tag{12c}$$

$$r \in [n-1]^{|\mathcal{V}|} \tag{12d}$$

Constraints (12a) and (12d) ensure there is a linear order. Constraints (12b) force all those vertices which have been chosen to be in the initial clique to take a rank in $[K]$, and since there are $K+1$ vertices selected in the clique all these positions will be filled. Note that the actual order of the clique does not matter, as we are merely looking for the existence of an order and any permutation of the initial clique will give a valid DVOP order. Constraints (12c) ensure that for all vertices outside the initial clique, if a vertex u witnesses vertex v, then u cannot come after v in the order as otherwise it would not be a valid witness.

Given a feasible master problem solution $(\hat{\kappa}, \hat{w}, \hat{y})$, if ($\mathbb{SP}2$) is feasible, we say that there exists a subproblem feasible solution, or a $rank(\cdot)$, *respecting* $(\hat{\kappa}, \hat{w})$. Similarly, if there exists a subproblem solution satisfying (12a), (12b), and (12d), but not necessarily (12c), we say that there exists a subproblem solution, or a $rank(\cdot)$, *respecting* $\hat{\kappa}$.

If the master problem solution does not lead to a subproblem feasible solution respecting it, thus a feasible DVOP, we would like to cut it off. To develop such cuts, we consider the possible "sources of infeasibility" in a master problem solution $(\hat{\kappa}, \hat{w}, \hat{y})$. Clearly, since $\hat{y}$ does not appear in our subproblem it is not a source of infeasibility. The $\hat{w}$, assuming $\hat{\kappa}_v = 0$ for the vertices in question may lead to an infeasible solution due to the logical constraints (12c), which ensure the proper ranks of the vertices, contradicting with the constraints (12a) and (12d) which ensure a linear order. As we will show later, this will indeed arise when the given master problem solution has "cycles on the w". For example, recall the graph in Figure 6a with order $(v_1, v_3, v_0, v_4, v_2, v_5)$, a solution which assigns $\hat{\kappa}_{v_2} = \hat{\kappa}_{v_5} = 0$ and $\hat{w}_{v_2 v_5} = \hat{w}_{v_5 v_2} = 1$ is feasible to the master problem.

However, by (12a) and (12d), the constraints (12c) give a contradiction as we would like to have v_2 precedes v_5 in the order and v_5 precedes v_2 simultaneously. This can be interpreted as having a 2-cycle, consisting of nodes v_2 and v_5, in the $\hat{w}$ solution.

As intuitively explained above, we can think of infeasibilities caused by $\hat{w}$ as *cycles*. Formally, let $\overrightarrow{G} = (\mathcal{V}, \mathcal{A})$ be the directed graph equivalent of $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{A} = \{(i,j),(j,i) \ : \ \forall \ \{i,j\} \in \mathcal{E}\}$ where (i,j) denotes a directed arc from i to j. If vertex u witnesses vertex v, i.e., $\hat{w}_{vu} = 1$, this is equivalent to the arc (v,u) being selected in $\overrightarrow{G}$, as in the solid arcs in the witness graph in Figure 6b. So, if vertex u witnesses vertex v, and v witnesses u, i.e., $\hat{w}_{vu} = \hat{w}_{uv} = 1$, we have selected (v,u) and (u,v), which forms a 2-cycle in $\overrightarrow{G}$. We define *cycles in* w as $C = (\mathcal{V}^C, \mathcal{A}^C)$, where C is a cycle subgraph of $\overrightarrow{G}$ and let $\mathcal{C}$ denote the set of all cycles in $\overrightarrow{G}$.

The only other potential source of subproblem infeasibility in a master problem solution $(\hat{\kappa}, \hat{w}, \hat{y})$ is the $\hat{\kappa}$ vector. However, we will next prove that the cycles in w are indeed the only source of infeasibility. In that regard, the first observation is that $\hat{\kappa}$ provides a feasible initial $(K+1)$-clique for a DVOP order that the subproblem is searching for, which is stated in the following lemma. This is illustrated by the solid arcs in Figure 7.

Lemma 1. *If* $(\hat{\kappa}, \hat{w}, \hat{y})$ *is feasible to* (MIP2) *then there exists* $\hat{r} \in \mathbb{Z}^{|\mathcal{V}|}$ *such that* (12a), (12b), *and* (12d) *are satisfied at* $(\hat{\kappa}, \hat{r})$.

Proof. Proof. Let $\hat{\mathcal{V}}^1 = \{v \in \mathcal{V} : \hat{\kappa}_v = 1\}$ and $\hat{\mathcal{V}}^0 = \{v \in \mathcal{V} : \hat{\kappa}_v = 0\}$. Note that $|\hat{\mathcal{V}}^1| = K+1$ and $|\hat{\mathcal{V}}^0| = |\mathcal{V}| - (K+1)$ due to (11b) and (11f). Arbitrarily ordering $\hat{\mathcal{V}}^1$ and $\hat{\mathcal{V}}^0$ as $\hat{v}^1_1, \hat{v}^1_2, \ldots, \hat{v}^1_{K+1}$ and $\hat{v}^2_1, \hat{v}^2_2, \ldots, \hat{v}^2_{|\mathcal{V}|-(K+1)}$, respectively, construct $\hat{r}$ as

$$\hat{r}_v = \begin{cases} i-1, & \text{if } v = \hat{v}^1_i \text{ for some } i \in [1, K+1] \\ i+K, & \text{if } v = \hat{v}^2_i \text{ for some } i \in [1, |\mathcal{V}| - (K+1)] \end{cases} \qquad \forall \ v \in \mathcal{V}.$$

Then, it is easy to confirm that (12a), (12b), and (12d) hold at $(\hat{\kappa}, \hat{r})$. □

The next result shows that in a master feasible solution $\hat{w}$, every vertex in the initial clique is witnessed only by vertices in the initial clique, i.e., there cannot be the dotted back-edges illustrated in Figure 7.

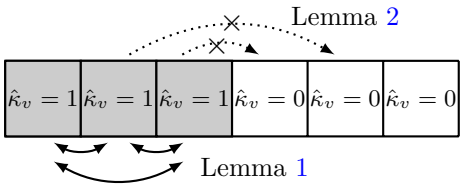


Figure 7: Witness graph for a given master candidate solution $(\hat{\kappa}, \hat{w}, \hat{y})$ for a case with $K = 2$. The initial $(K+1)$-clique positions are in gray. The arcs correspond to the witness relationships provided by $\hat{w}$.

Lemma 2. *If* $(\hat{\kappa}, \hat{w}, \hat{y})$ *is feasible to* (MIP2), *then* $\hat{w}_{vu} = 0 \ \forall \ \{v,u\} \in \mathcal{E}$ *with* $\hat{\kappa}_v = 1, \hat{\kappa}_u = 0$.

Proof. Proof. Given $(\hat{\kappa}, \hat{w}, \hat{y})$ feasible to (MIP2), let $v \in \mathcal{V}$ with $\hat{\kappa}_v = 1$, that is v has been selected for the initial clique. By constraints (11b), (11c), and (11d) and the w domains, $\hat{w}_{vi} = 1$ for all $i \in \mathcal{V}$, such that $v \neq i$ and $\hat{\kappa}_i = 1$, i.e., v is witnessed at least by K other vertices in the first clique. By (11e), we have $\sum_{j \in \mathcal{N}(v)} \hat{w}_{vj} = K - \hat{y}_v$ but since $\hat{y} \geq 0$, $\sum_{j \in \mathcal{N}(v)} \hat{w}_{vj} \leq K$. Thus, v is witnessed by exactly K vertices, and those are exactly the other vertices in the initial clique. As such, $\hat{w}_{vu} = 0 \ \forall \ \{v,u\} \in \mathcal{E}$ with $\hat{\kappa}_v = 1, \hat{\kappa}_u = 0$ as required. □

Lemma 2 implies that there is no cycle in $\hat{w}$ including vertices both from inside and outside of the first $(K+1)$-clique described by $\hat{\kappa}$ since completing such a cycle would require the use of a back-edge from the initial clique to outside.

Corollary 1. *Let $(\hat{\kappa}, \hat{w}, \hat{y})$ be feasible to $(\mathbb{MP}2)$. Define $\hat{\mathcal{V}}^1 = \{v \in \mathcal{V} : \hat{\kappa}_v = 1\}$ and $\hat{\mathcal{V}}^0 = \{v \in \mathcal{V} : \hat{\kappa}_v = 0\}$. For any $C = (\mathcal{V}^C, \mathcal{A}^C) \in \mathcal{C}$ with $\hat{w}_{vu} = 1$ for all $(v,u) \in \mathcal{A}^C$, either $\mathcal{V}^C \cap \hat{\mathcal{V}}^1 = \emptyset$ or $\mathcal{V}^C \cap \hat{\mathcal{V}}^0 = \emptyset$.*

Given an $(\mathbb{MP}2)$ solution $(\hat{\kappa}, \hat{w}, \hat{y})$, as there always exists a feasible initial $(K+1)$-clique by Lemma 1, if the subproblem is infeasible, the issue is failing to build the rest of the order. As mentioned before, a reason could be the existence of a cycle in the $\hat{w}$ solution. Due to Corollary 1, such a cycle can only occur outside of the initial clique. We propose the following inequality to remove infeasibilities caused by a cycle $C = (\mathcal{V}^C, \mathcal{A}^C)$ in the w,

$$\sum_{(v,u)\in\mathcal{A}^C} w_{vu} \leq |\mathcal{V}^C| - 1 + \mathbb{I}(|\mathcal{V}^C| \leq K+1)\kappa_\iota \tag{13}$$

where ι is any (e.g., the smallest) index of a vertex in $\mathcal{V}^C$. When there is a cycle C with $|\mathcal{V}^C| > K+1$ we have the classical cycle breaking cut. However, if $|\mathcal{V}^C| \leq K+1$ it is possible we have detected a cycle in the initial $(K+1)$-clique, which is valid. In this case, we need to *lift* the classical cycle breaking into the (κ, w)-space to be valid for the master problem, which is accomplished by adding κ_ι to the right-hand-side of the cut, so that if the cycle is in the initial clique, $\kappa_\iota = 1$ and the inequality is redundant. By Corollary 1, κ_ι is only 1 if the cycle is in the initial clique, and otherwise the cycle breaking inequality will hold. The following lemma formalizes the validity of the proposed cuts.

Lemma 3. *The inequality (13) is valid for $\{(\kappa, w) : \exists\, y, r$ s.t. (κ, w, y) is feasible to $(\mathbb{MP}2)$ and r respects $(\kappa, w)\}$.*

Proof. Proof. *Case 1:* If $|\mathcal{V}^C| > K+1$, $\hat{w}_{ij} = 1$ for all $(i,j) \in \mathcal{A}^C$. By Lemma 2 we know $\hat{\kappa}_v = 0$ for all $v \in \mathcal{V}^C$. So we are outside the initial clique and would like to break cycles of any length, and the cut reduces to the standard cycle breaking inequality.
Case 2: If $|\mathcal{V}^C| \leq K+1$ and $\kappa_\iota = 1$ the inequality is redundant and thus valid since we only want to break cycles outside the initial clique. If $|\mathcal{V}^C| \leq K+1$ and $\kappa_\iota = 0$ then the cycle is not in the initial clique and we would like to break it, and again the inequality is the standard cycle breaking inequality. □

In order to show $(\mathbb{MP}2)$ and the inequalities (13) for all $C \in \mathcal{C}$ are sufficient to prove a solution is optimal to `MIN DOUBLE`, we introduce the extended formulation obtained by combining $(\mathbb{MP}2)$ and $(\mathbb{SP}2)$.

$$(\mathbb{EF}) : \min \sum_{v\in\mathcal{V}} y_v + 1 \tag{14a}$$

$$\text{s.t.} \sum_{v\in\mathcal{V}} \kappa_v = K+1 \tag{14b}$$

$$\kappa_v + \kappa_u \leq 1 \qquad \forall\, v,u \in \mathcal{V} : u \neq v, \{v,u\} \notin \mathcal{E} \tag{14c}$$

$$\kappa_v \leq w_{uv} \qquad \forall\, v \in \mathcal{V}, u \in \mathcal{N}(v) \tag{14d}$$

$$\sum_{u\in\mathcal{N}(v)} w_{vu} = (K+1)(1-\kappa_v) - y_v + K\kappa_v \qquad \forall\, v \in \mathcal{V} \tag{14e}$$

$$\text{AllDifferent}(r_0, r_1, ..., r_{n-1}) \tag{14f}$$
$$\kappa_v = 1 \Rightarrow r_v \leq K \qquad \forall\ v \in \mathcal{V} \tag{14g}$$
$$\kappa_v + \kappa_u = 0 \text{ and } w_{vu} = 1 \Rightarrow r_u \leq r_v - 1 \qquad \forall\ v \in \mathcal{V}, u \in \mathcal{N}(v) \tag{14h}$$
$$y \in \{0,1\}^{|\mathcal{V}|}, w \in \{0,1\}^{|\mathcal{A}|}, \kappa \in \{0,1\}^{|\mathcal{V}|}, r \in [n-1]^{|\mathcal{V}|} \tag{14i}$$

Let $\mathcal{EF}$ denote the feasible region of ($\mathbb{EF}$). Finally, we will show that ($\mathbb{MP}2$) and the cycle breaking cuts (13) for all $C \in \mathcal{C}$ are sufficient to prove optimality.

Theorem 1. *If $(\hat{\kappa}, \hat{w}, \hat{y})$ is feasible to ($\mathbb{MP}2$) and (13) for all $C \in \mathcal{C}$, then $(\hat{\kappa}, \hat{w}, \hat{y}) \in proj_{\kappa,w,y}(\mathcal{EF})$, i.e., there exists $\hat{r}$ such that $(\hat{\kappa}, \hat{w}, \hat{y}, \hat{r}) \in \mathcal{EF}$.*

Proof. Proof. By induction on $p \in \mathbb{N}$ with $K \leq p \leq n-1$ denoting the position for which there exists a DVOP order respecting $(\hat{\kappa}, \hat{w})$ in the $[p]$ subset of positions.
Base Case: When $p = K$ there is a DVOP order respecting the $(\hat{\kappa}, \hat{w})$ since we have a $(K+1)$-clique from the $\hat{\kappa}$ which all witness each other.
Hypothesis: Assume we have a DVOP order respecting $(\hat{\kappa}, \hat{w})$ in the $[p]$ subset of positions.
Inductive Step: We show there exists a DVOP order in positions $[p+1]$ respecting $(\hat{\kappa}, \hat{w})$. Fix a DVOP order respecting $(\hat{\kappa}, \hat{w})$ in positions $[p]$. Denote the vertices in the order as $v_1, v_2, \ldots, v_p$, denote, without loss of generality, all other vertices as $q_1, q_2, \ldots, q_{n-p}$.
Case 1: If there is a q such that $\sum_{i=1}^{p} \hat{w}_{qv_i} = K$, we have an order with $p+1$ where q is the $(p+1)^{\text{th}}$ vertex in the order. So we assume no such vertex exists.
Case 2: Since all q need at least K witnesses and we force each vertex to be witnessed by all its neighbours in the initial clique, there is no q with all its witnesses in the initial clique. Thus every q has at least one witness in the set of q vertices or in $v_{K+1}, v_{K+2}, \ldots, v_p$, however not all q can have such witnesses in the q vertices or there will be a cycle. So there exists a $\hat{w}_{q_j v_i} = 1$, that is a q_j that is witnessed by some $v_i \in \{v_{K+2}, v_{K+3}, \ldots, v_p\}$. Without loss of generality, let such a vertex be q_1, so $\hat{w}_{q_1 v_i} = 1$, but it must have $\hat{w}_{q_1 q_j} = 1$ otherwise it would fall into Case 1. Without loss of generality, let q_2 witness q_1, so $\hat{w}_{q_2 q_1} = 0$. If q_2 has all witnesses in v, we have a DVOP where q_2 is the $(p+1)^{\text{th}}$ vertex in the order. So we assume q_2 has a witness in q, say q_3, if q_3 has all witnesses in v, we have a DVOP where q_3 is the $(p+1)^{\text{th}}$ vertex in the order. So we assume q_3 has a witness in q, say q_4, and so on. Otherwise, any $\hat{w}_{q_i q_j} = 0\ \forall\ j < i$, in the worst case $q_i = q_{n-p}$ so it must have all witnesses in v. Thus we have a vertex by which we can extend the DVOP order. □

We provide a valid inequality for ($\mathbb{EF}$) the proof of which is provided in the Appendix E.

Proposition 1. *For any $v \in \mathcal{V}$, the inequality $y_v \leq 1 - \kappa_v$ is valid for ($\mathbb{EF}$).*

As the inequalities (13) for all $C \in \mathcal{C}$ are sufficient for proving optimality, we would like to generate them in a cutting plane fashion using ($\mathbb{SP}2$) instead of adding all the cycle breaking inequalities to ($\mathbb{MP}2$) initially. The procedure used to solve the witnessed-based hybrid decomposition is shown in a cutting plane fashion in Algorithm 3 for ease of presentation, however in practice it is implemented as a branch-and-cut algorithm. We note that if the subproblem is infeasible, we generate a cycle C in $\hat{w}$, but do not specify the method. As previous stated, the cycle generation can be via a combinatorial Benders procedure wherein we solve an IIS to find the cut. However, we may also use the methods in [19] to search for a topological ordering of $\hat{w}$ using depth-first search. If there is no topological order, we can find a cycle in the digraph since there will be some pair of vertices u, v in the order with $\hat{w}_{vu} = 1$ and and such that the search returns $r_v \leq r_u$ which is a contradiction. In this case, (v, u) is in the cycle, with the remaining arcs found on the shortest path from u to v.

5 Computational Results

In this section we present a computational study which compares the mathematical models presented in Sections 3 and 4, as well as the existing models from the literature. We first review the data used in the experiments, then the experimental set up, and finally we outline and discuss the results. Full results tables, with the exception of the pseudo-protein instances, can be found in the Online Supplement F; in this section, we only summarize our main observations from the tables.

5.1 Instances

Our data set consists of three kinds of graph instances which are used to test the performance of the mathematical models; random instances, synthetic instances[2], and pseudo-protein instances[3].

Random Instances. The test data set consisting of randomly generated graphs, with $n \in \{20, 25, 30, 35\}$ and expected edge density (measured as $D = \frac{2|\mathcal{E}|}{n(n-1)}$) in $\{0.3, 0.4, 0.5\}$. The number of doubles for these instances ranged from 1 to 14, with an average of 2 doubles per instance. A summary of the random instances can be seen in Table 1. Of these instances, only three were infeasible for DVOP with $K = 3$, we left these instances in the test set to ensure our algorithms can also detect infeasible DVOP instances and observed that for some algorithms, this time is as fast as the greedy algorithm for DVOP. However, we do not include these instances in the performance profiles.

Synthetic Instances. Synthetic instances for `MIN DOUBLE` were created to simulate the *sparsest* possible graph instances that would still have a DVOP order. Given K, a number of doubles, and a *noise* factor, a synthetic instance is constructed by first building a $(K + 1)$-clique. We then randomly select vertices outside the initial clique to be doubles, next we add edges between vertices ensuring that the doubles and non-doubles have the appropriate number of predecessors in the order. Finally, we add in some noise edges. Notice that the addition of the extra edges may cause there to be less double vertices, however there will always be at least one double vertex by definition.

We generate synthetic instances with $K = 3$ and fix $n \in \{25, 30, 35\}$, the number of doubles in $\in \{\lceil 0.1 \times n \rceil, \lceil 0.1 \times n \rceil + 1, \lceil 0.1 \times n \rceil + 2\}$, and a noise factor in $\{0.1, 0.15, 0.2\}$. For each n, doubles, noise triple we generate a minimal graph as described above, where the number of extra edges added is $\lceil n \times \text{noise} \rceil$. The full algorithm for generating synthetic instances can be found in the Online Supplement G. Table 2 summarizes the synthetic instances. We remark that the synthetic instances are always feasible.

Pseudo-Protein Instances. The pseudo-protein instances come from [16], although they are not the same instances used in their publication. The instances are generated by modifying existing Protein Data Bank instances by first trimming the instance to a desired number of nodes, and then randomly removing edges and checking for feasibility until the desired density is met. In total this test set has 399 instances with nodes in $\{30, 40, \ldots, 100\}$ and expected edge density in $[0.03, 0.12]$. We remark that these pseudo-protein instances are always feasible.

[2] Random and synthetic instances can be found at https://sites.google.com/site/mervebodr/home/MINDOUBLE_Instances.zip.

[3] Pseudo-protein instances can be found at https://gitlab.insa-rennes.fr/Jeremy.Omer/MinDouble.

5.2 Implementation Details

IPs are solved using IBM ILOG CPLEX version 12.8.0 and CPs are solved using IBM ILOG CP Optimizer version 12.8.0, which we implemented in C++. All experiments were run on MacOS with 16GB RAM and a 2.3 GHz Intel Core i5 processor, using a single thread. We used the lazy constraint callback function to implement the branch-and-cut procedures. We used $K = 3$ for all experiments and set the time limit to 1000 seconds. (We also tested the random instances with $K = 4$ and $K = 5$. As the findings were similar to the $K = 3$ case, their results are provided in Tables 12 and 13 of Online Supplement F, respectively.)

The three existing formulations of [16], ($\mathbb{CYCLES}$), ($\mathbb{RANKS}$), and ($\mathbb{CCG}$), are re-implemented using the details provided in their publication. Also, we provided the greedy solutions of [16] to warm start all the algorithms for random and pseudo-protein instances.

5.3 Computational Results and Discussion

We begin by comparing the performance of selected formulations on the random instances. The performance profile in Figure 8 shows that many of the methods have similar performance. Overall, ($\mathbb{CP}^{\mathbb{VERTEX}}$), ($\mathbb{CP}^{\mathbb{COMBINED}}$), and the naive decomposition all with valid inequalities perform the best, although they are only able to solve 30 of the 33 feasible instances within the time limit. We remark that ($\mathbb{CCG}$) is only able to solve one of the feasible random instances, and all the formulations of this work outperform those the literature by both time and number of instances solved.

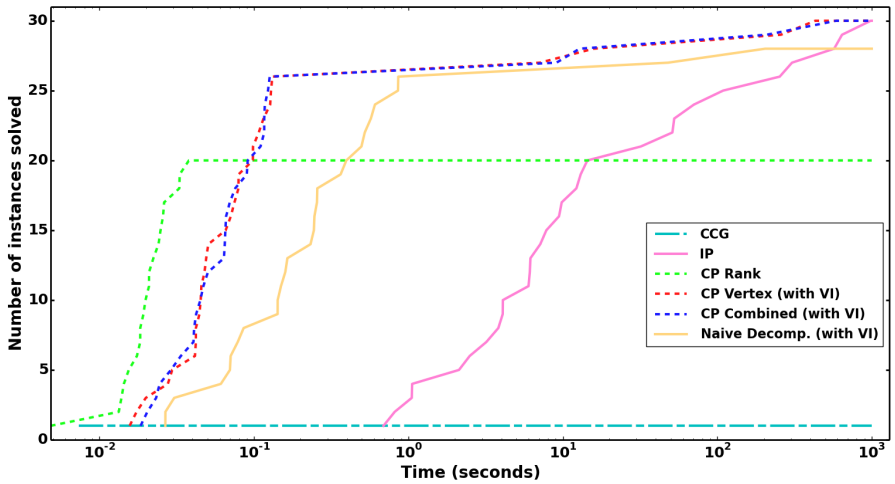


Figure 8: Solution times of the models on random instances.

The three existing formulations from the literature ($\mathbb{CYCLES}$), ($\mathbb{RANKS}$), and ($\mathbb{CCG}$) are all outperformed by ($\mathbb{IP}$) (details can be seen in Table 6 of Online Supplement F). ($\mathbb{IP}$) is able to solve more than half of the instances with $D = 0.4$ and $D = 0.5$ in less than a second. However, solving instances with $D = 0.3$ is more difficult for ($\mathbb{IP}$) as the time limit is reached on 4 of these instances, and the time to a solution exceeds 100 seconds on 4 others. We remark that ($\mathbb{CCG}$) is only able to solve 4 out of 36 instances, three of which are infeasible, and hits the time limit on all others. When ($\mathbb{CCG}$) reaches the time limit on instances with $n \geq 30$ and $D \geq 0.4$ it has applied at least 14,000 cycle separation cuts. Similarly, ($\mathbb{CYCLES}$) is only able to solve 9 instances, three

of which are infeasible, and (RANKS) solves 4 instances, three of which are infeasible. Despite solving fewer instances than (CYCLES), (CCG) has better solution times on the instances that both formulations are able to solve. For this reason, and because [16] conclude that of the existing approaches, (CCG) performs the best, we proceed to compare with (CCG).

The CP formulations have better performance than all of the IP formulations. Their detailed comparison is given in Table 7 of Online Supplement F. ($\mathbb{CP}^{\mathbb{RANK}}$) performs best on instances with $D = 0.4$ or $D = 0.5$, solving most in a fraction of a second. However, if ($\mathbb{CP}^{\mathbb{RANK}}$) cannot solve an instance almost immediately, it cannot solve it in the time limit. As ($\mathbb{CP}^{\mathbb{RANK}}$) solves only 20 instances in total, we deem its performance worse overall than that of ($\mathbb{CP}^{\mathbb{VERTEX}}$) and ($\mathbb{CP}^{\mathbb{COMBINED}}$). When valid inequalities and variable fixing are added to ($\mathbb{CP}^{\mathbb{VERTEX}}$) the time required to solve 25 of the total of 32 solved instances decreases. Similarly, when valid inequalities and variable fixing are added to ($\mathbb{CP}^{\mathbb{COMBINED}}$) the time decreases for 24 of 33 instances. Thus we conclude that the ($\mathbb{CP}^{\mathbb{VERTEX}}$) and ($\mathbb{CP}^{\mathbb{COMBINED}}$) formulations with valid inequalities outperform those without.

As n increases, ($\mathbb{CP}^{\mathbb{VERTEX}}$) with valid inequalities is able to solve instances with $D = 0.4$ and $D = 0.5$ in a fraction of a second, however, it takes significantly longer to solve instances with $D = 0.3$ reaching the time limit for 4 of these lowest density instances. ($\mathbb{CP}^{\mathbb{COMBINED}}$) with valid inequalities reaches the time limit when solving 3 instances all of which have $D = 0.3$, but for all other instances with $n \geq 25$ and $D = 0.4$ or $D = 0.5$ the solution times are competitive. Finally, we compare the extended formulation (EF) which is also a CP, it is outperformed by the other CPs on all but 2 instances and hits the time limit on 12 instances. We conclude that ($\mathbb{CP}^{\mathbb{VERTEX}}$) and ($\mathbb{CP}^{\mathbb{COMBINED}}$) with valid inequalities have the best CP performance.

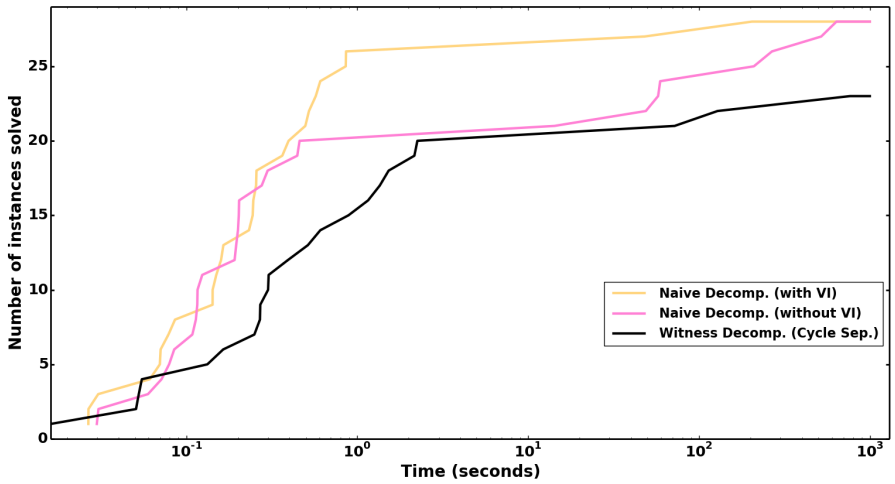


Figure 9: Solution times of the decompositions on random instances.

To show the strength of the valid inequalities and variable fixing for the naive decomposition, we compare the naive decomposition with and without these enhancements and the best performing witness-based decomposition in Figure 9. We were able to detect at least one valid inequality or apply variable fixing for all random instances. The results (provided in detail in Table 8 of Online Supplement F) show that the naive decomposition reaches the time limit on the same 7 instances with or without the addition of the valid inequalities. We also remark that for most instances the naive decomposition with valid inequalities has fewer cuts, and because the majority of the time to find a solution is spent finding an IIS to generate cuts, the naive decomposition with valid

inequalities is faster than without.

We consider a variety of options with which to solve the witness-based decomposition, first we note that because we would like to break cycles in w, we can add cycle breaking constraints to ($\mathbb{EF}$) before beginning the search. In Table 9 of Online Supplement F, we compare the witness-based decomposition using an IIS with no cycle breaking in ($\mathbb{EF}$), breaking only 2-cycles, and breaking both 2 and 3-cycles. Of these the best performance is from breaking both 2 and 3-cycles which we also implement with the cycle separation. We add the valid inequality from Proposition 1 to all formulations. Naturally, finding an IIS is slower than separating a cycle using depth-first search, our results indicate that the cycle separation is several orders of magnitude faster than finding an IIS (these negligible times have been omitted from Table 9). In terms of the number of cuts, neither the IIS nor the cycle separation version dominates, but the cycle separation version is able to solve 3 more instances than the IIS version within the time limit. We also remark that the cycle separation version has the fastest solution time on 20 instances.

Figure 10 gives the solution times for the best performing methods on the synthetic instances, as well as the best method from the literature, ($\mathbb{CCG}$). We observe that for synthetic instances the witness-based decomposition with cycle separation outperforms the other formulations both in speed and in terms of the number of instances solved. Table 11 of Online Supplement F shows the

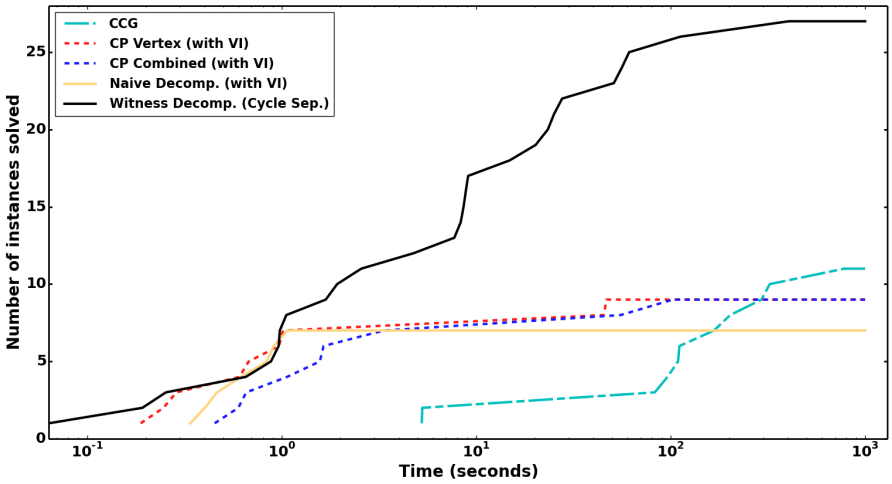


Figure 10: Solution times of the best performing models on synthetic instances.

results for synthetic instances. For this test set, ($\mathbb{CCG}$) reaches the time limit on 16 instances. It struggles in particular when the number of nodes increases, as it only solves 4 instances with $n \geq 30$. The two CP formulations ($\mathbb{CP}^{\text{VERTEX}}$) and ($\mathbb{CP}^{\text{COMBINED}}$) with valid inequalities, have very similar performance, reaching the time limit on 18 instances; those with both a large number of nodes and of double vertices. On synthetic instances, the naive decomposition with valid inequalities reaches the time limit on the most instances, 20 in total. This decomposition is however, faster than the witness-based decomposition on instances that they both solve. The naive decomposition stays at the root node for all instances as they are hitting the time limit while trying to find an IIS to generate a cut. Finally, the witness-based decomposition is the best performing overall, it is the fastest to a solution for all but four instances and only reaches the time limit on a single instance.

Figure 11 gives the solution times for the best performing methods on pseudo-protein instances. Similarly to the synthetic instances, the best performing method is the witness-based decomposition, it is able to solve 222 of the 399 instances within the time limit. ($\mathbb{CCG}$) is able to solve 148

instances within the time limit. Similar to its performance on the random instances, ($\mathbb{CP}^{\mathbb{RANK}}$) dominates all other methods in the beginning as it is able to solve 57 pseudo-protein instances very quickly, but is unable to solve any more instances after that. ($\mathbb{CP}^{\mathbb{VERTEX}}$) and ($\mathbb{CP}^{\mathbb{COMBINED}}$) with valid inequalities have similar performance, solving 58 and 57 instances within the time limit, respectively. Both ($\mathbb{IP}$) and the naive decomposition with valid inequalities were outperformed by all other methods, and were only able to solve 57 and 46 instances within the time limit, respectively.

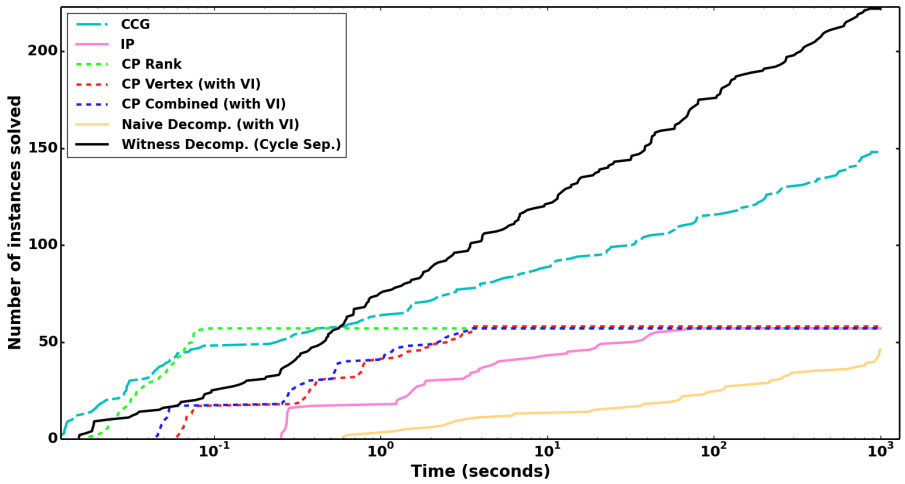


Figure 11: Solution times of the best performing models on pseudo-protein instances.

6 Conclusion

We propose the first CP formulations and hybrid decomposition methods for `MIN DOUBLE` as well as a novel IP. We provide the first valid inequalities for `MIN DOUBLE` and use polyhedral theory to prove the correctness of the witness-based decomposition.

Our computational results show ($\mathbb{CP}^{\mathbb{VERTEX}}$) and ($\mathbb{CP}^{\mathbb{COMBINED}}$) with valid inequalities perform best on random instances, but still struggle with the lowest density instances in this data set. For both synthetic and pseudo-protein instances, we observe that the witness-based decomposition has superior performance to all other formulations. We conclude scalability is still an issue for this problem as the most difficult instances to solve are those in the pseudo-protein dataset with large number of nodes. We also remark that for all instance sets studied there exists a novel formulation in this work that outperforms those of the literature.

Acknowledgments

We are grateful to Leo Liberti for introducing us to the distance geometry area and this particular topic. Also, we would like to thank Jérémy Omer for kindly providing the pseudo-protein instances similar to the ones used in Omer and Gonçalves (2017).

References

[1] A. Cassioli, O. Günlük, C. Lavor, and L. Liberti. Discretization vertex orders in distance geometry. *Discrete Applied Mathematics*, 197:27–41, 2015.

[2] G. Codato and M. Fischetti. Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[3] D. Coudert. A note on Integer Linear Programming formulations for linear ordering problems on graphs. Research report, Inria ; I3S ; Universite Nice Sophia Antipolis ; CNRS, 2016.

[4] D. S. Gonçalves, A. Mucherino, C. Lavor, and L. Liberti. Recent advances on the interval distance geometry problem. *Journal of Global Optimization*, 69(3):525–545, 2017.

[5] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32(6):1195–1220, 1984.

[6] C. Lavor, J. Lee, A. Lee-St. John, L. Liberti, A. Mucherino, and M. Sviridenko. Discretization orders for distance geometry problems. *Optimization Letters*, 6(4):783–796, 2012.

[7] C. Lavor, L. Liberti, W. A. Lodwick, and T. M. da Costa. *An Introduction to Distance Geometry applied to Molecular Geometry*. Springer, 2017.

[8] C. Lavor, L. Liberti, and A. Mucherino. The interval branch-and-prune algorithm for the discretizable molecular distance geometry problem with inexact distances. *Journal of Global Optimization*, 56(3):855–871, 2013.

[9] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.

[10] T. Migot and J. Omer. Vertex order with optimal number of adjacent predecessors. *Discrete Mathematics & Theoretical Computer Science*, 22, 2020.

[11] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.

[12] A. Mucherino. *On the Discretization of Distance Geometry: Theory, Algorithms and Applications*. Habilitation à diriger des recherches, IRISA, 2018.

[13] A. Mucherino, C. Lavor, and L. Liberti. The discretizable distance geometry problem. *Optimization Letters*, 6(8):1671–1686, 2012.

[14] A. Mucherino, C. Lavor, L. Liberti, and N. Maculan. *Distance geometry: Theory, methods, and applications*. Springer Science & Business Media, 2012.

[15] A. Mucherino, J. Omer, L. Hoyet, P. R. Giordano, and F. Multon. An application-based characterization of dynamical distance geometry problems. *Optimization Letters*, pages 1–15, 2018.

[16] J. Omer and D. S. Gonçalves. An integer programming approach for the search of discretization orders in distance geometry problems. *Optimization Letters*, pages 1–14, 2017.

[17] J. Saxe. *Embeddability of Weighted Graphs in K-space is Strongly NP-hard*. CMU-CS-80-102. Carnegie-Mellon University, Department of Computer Science, 1980.

[18] B. M. Smith. Dual models in constraint programming. Technical report, 2001.

[19] R. E. Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–185, 1976.

A Integer Programming Model for MIN NODES

This model is an extension of the model ($\mathbb{IP}$) for MIN DOUBLE, with the addition of constraints (15j)-(15l) and with a different objective function. Define binary variables $y_r = 1$ if the vertex at position $r \in [n-1]$ is a double, 0 otherwise. Define binary variables $x_{vr} = 1$ if the vertex $v \in \mathcal{V}$ is at rank $r \in [n-1]$, 0 otherwise. Finally, define positive integer variables m_r equal to the number of nodes at level r of the BP tree. We also introduce binary indicator variables z_{vr} for all $v \in \mathcal{V},\ r \in [n-1]$ to express logical constraints.

$$\min \sum_{r\in[n-1]} m_r \tag{15a}$$

$$\text{s.t.} \sum_{r\in[n-1]} x_{vr} = 1 \qquad \forall\ v \in \mathcal{V} \tag{15b}$$

$$\sum_{v\in V} x_{vr} = 1 \qquad \forall\ r \in [n-1] \tag{15c}$$

$$\sum_{u\in\mathcal{N}(v)} \sum_{j\in[r-1]} x_{uj} \geq r x_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [1, K] \tag{15d}$$

$$\sum_{u\in\mathcal{N}(v)} \sum_{j\in[r-1]} x_{uj} \geq K x_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K+1, n-1] \tag{15e}$$

$$y_r = 0 \qquad \forall\ r \in [K-1] \tag{15f}$$

$$y_K = 1 \tag{15g}$$

$$\sum_{u\in\mathcal{N}(v)} \sum_{j\in[r-1]} x_{uj} \geq (K+1) z_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K, n-1] \tag{15h}$$

$$x_{vr} - y_r \leq z_{vr} \qquad \forall\ v \in \mathcal{V}, r \in [K, n-1] \tag{15i}$$

$$m_r = 1 \qquad \forall\ r \in [K-1] \tag{15j}$$

$$m_r \geq m_{r-1} \qquad \forall\ r \in [K, n-1] \tag{15k}$$

$$m_r - 2m_{r-1} \geq -2^{r-K}(1 - y_{r-1}) \qquad \forall\ r \in [K, n-1] \tag{15l}$$

$$x \in \{0,1\}^{|\mathcal{V}|\times n}, y \in \{0,1\}^n, z \in \{0,1\}^{|\mathcal{V}|\times n}, m \in \mathbb{Z}^n_+ \tag{15m}$$

Constraints (15b) and (15c) ensure that there is a bijection from the vertices to the ranks, that is each vertex has exactly one rank and vice versa. Constraints (15d) ensure that we have an initial clique of size $K+1$, since each vertex in the clique will be adjacent to all its predecessors. Constraints (15e) ensure that each vertex after the initial clique has at least K adjacent predecessors. Constraints (15f) and (15g) fix the double value of positions $[K]$. Constraints (15h) and (15i) link the x_{vr} and y_v variables, where we want to enforce that if vertex v is a non-double, then for any rank r, either v is not in position r or it has at least $K+1$ adjacent predecessors. Constraints (15j) fix the first $K-1$ layers to have 1 node as follows from the problem definition. Constraints (15k) ensure the number of nodes at each level is non-decreasing. Finally, the constraints (15l) enforce the logical constraint $y_{r-1} = 1 \implies m_r \geq 2m_{r-1}$. Since we are minimizing the sum over the number of nodes at each level, we do not need to also enforce $y_{r-1} = 1 \implies m_r \leq 2m_{r-1}$, as at an optimal solution this constraint is satisfied. The big-M is active in these constraints when $y_{r-1} = 0$, the maximum difference is given when every position before r is a double, thus 2^{r-K}.

B Details of the Existing Models

Prior to this work, [16] present two IP formulations and one branch-and-cut procedure for `MIN DOUBLE`.

B.1 The cycles IP

Define binary precedence variables $p_{ij} = 1$ if and only if vertex $i \in \mathcal{V}$ precedes vertex $j \in \mathcal{V}$ in the order, double variables $y_v = 1$ if and only if vertex $v \in \mathcal{V}$ is double or is in the first clique. Let $\mathcal{K}$ denote the set of all possible K-cliques that can be extended to a $(K+1)$-clique, note these cliques are not ordered. Define binary the initial clique selection variables $\kappa_c = 1$ if and only if clique $c \in \mathcal{K}$, is the initial clique. Let R_i^c be the rank from $[1, K]$ of vertex i in clique c. Then, `MIN DOUBLE` can be formulated as follows:

$$(\mathbb{CYCLES}) : \min \sum_{v \in \mathcal{V}} y_v - K \tag{16a}$$

$$\text{s.t. } p_{ij} + p_{ji} = 1 \qquad \forall\, i, j \in \mathcal{V}, i \neq j \tag{16b}$$

$$p_{ij} + p_{jk} + p_{ki} \leq 2 \qquad \forall\, \{i, j\} \in \mathcal{E}, k \in \mathcal{V}, k \neq i, j \tag{16c}$$

$$\sum_{c \in \mathcal{K}} \kappa_c = 1 \tag{16d}$$

$$\sum_{\{i,j\} \in \mathcal{E}} p_{ji}\text{[4]} + \sum_{c \in \mathcal{K}: i \in c} (K - R_i^c + 1)\kappa_c \geq K + (1 - y_i) \qquad \forall\, i \in \mathcal{V} \tag{16e}$$

$$p_{ij} \in \{0, 1\} \qquad \forall\, i, j \in \mathcal{V}, i \neq j \tag{16f}$$

$$\kappa_c \in \{0, 1\} \qquad \forall\, c \in \mathcal{K} \tag{16g}$$

$$y_v \in \{0, 1\} \qquad \forall\, v \in \mathcal{V} \tag{16h}$$

Objective (16a) minimizes the number of double vertices in the order, the constant term is due to (16e) and will be explained shortly. Constraints (16b), (16c) are the usual linear ordering constraints used to break cycles of size 2 and 3 [5] and sufficient to determine a vertex ordering [3]. Constraint (16d) ensures we choose exactly one initial K-clique in the order. Constraints (16e) are logical constraints which enforce the appropriate number of adjacent predecessors for double or non-double vertices. Fixing a vertex $i \in \mathcal{V}$, $\sum_{\{i,j\} \in \mathcal{E}} p_{ji}$ gives the number of adjacent predecessors of i. If i is not in the chosen initial clique, the second left hand side summation is eliminated, and i has exactly K adjacent predecessors if it is a double and, $y_i = 1$ and otherwise it has at least $K + 1$, as mentioned in the discussion of (2c). If vertex i is inside the selected initial clique, the second left hand side summation becomes $(K - R_i^c + 1)$ and rearranging the terms gives that i has $R_i^c - y_i$ adjacent predecessors. Since each vertex in the initial clique must be adjacent to all the others in the clique and, i is the $(R_i^c)^{\text{th}}$ vertex in the clique, i must have $R_i^c - 1$ adjacent predecessors, meaning $y_i = 1$ for all i in the chosen initial clique. In order to be consistent across all formulations, we subtract these K doubles from the objective (16a), since by definition the vertices in the first clique are not double. Finally, constraints (16f), (16g), (16h) give binary domains for all the decision variables.

For ($\mathbb{CYCLES}$) we have

$$rank(v) = \sum_{\substack{u \in \mathcal{V} \\ u \neq v}} p_{uv}$$

$$double(v) = y_v, \forall\, v \in \mathcal{V}.$$

[4] We believe there is a typo in the original paper [16], which incorrectly writes p_{ij}.

B.2 The rank IP

Define integer variables $r_i \in [n-1]$ for the rank of vertex $i \in V$ in the order.

$$
\begin{aligned}
(\mathbb{RANKS}) : \min \quad & \sum_{v \in \mathcal{V}} y_v - K && && \text{(17a)}\\
\text{s.t. } & n\, p_{ij} + (r_i + 1) - (r_j + 1) \leq n - 1 && \forall\ \{i,j\} \in \mathcal{E} && \text{(17b)}\\
& \sum_{c \in \mathcal{K}} \kappa_c = 1 && && \text{(17c)}\\
& \sum_{\{i,j\} \in \mathcal{E}} p_{ji} + \sum_{c \in \mathcal{K}: i \in c} (K - R_i^c + 1)\kappa_c \geq K + (1 - y_i) && \forall\ i \in \mathcal{V} && \text{(17d)}\\
& p_{ij} \in \{0,1\} && \forall i,j \in \mathcal{V}, i \neq j && \text{(17e)}\\
& \kappa_c \in \{0,1\} && \forall c \in \mathcal{K} && \text{(17f)}\\
& y_v \in \{0,1\} && \forall\ v \in \mathcal{V} && \text{(17g)}\\
& r_v \in [n-1] && \forall\ v \in \mathcal{V} && \text{(17h)}
\end{aligned}
$$

Objective function (17a) and constraints (17c), (17d), (17e), (17f), and (17g) are as in (16). Constraints (17b) ensure that if vertex i precedes vertex j, then the rank of i is strictly less than the rank of j. Constraints (17h) enforce the domain of the rank variables to be one rank for each vertex in the order. Thus, constraints (17b) and (17h) replace the cycle breaking constraints in ($\mathbb{CYCLES}$).

For any feasible solution to ($\mathbb{RANKS}$) we have $rank(v) = r_v$, and $double(v) = y_v$.

B.3 Cycle Constraint Generation ($\mathbb{CCG}$)

The cutting-plane version of ($\mathbb{CCG}$) is given in Algorithm 4 for ease of exposition, but has been implemented in a branch-and-cut fashion. Algorithm 4 takes as input integer $q \geq 2$, it begins by generating all cycles of size $\leq q$, $\mathcal{C}_q$. A master problem MP is then created with the objective (17a) , constraints (17c), (17d), and cycle breaking constraints which will break all cycles in $\mathcal{C}_q$. It then passes the master problem solution to a function which will detect if there is cycle in $\bar{p}$. If the cycle is found, we can add a cycle cut to MP, otherwise we have an optimal solution and are done. We denote the set of cycles of a graph as $\mathcal{C}$, where a cycle $C \in \mathcal{C}$ has vertex set $\mathcal{V}^C$ and edge set $\mathcal{E}^C$, i.e., $C = (\mathcal{V}^C, \mathcal{E}^C)$.
As for ($\mathbb{CYCLES}$),

$$rank(v) = \sum_{\substack{u \in \mathcal{V}: \\ u \neq v}} p_{uv}$$

for ($\mathbb{CCG}$).

[16] show that Cycle Cut Generation outperforms both IP formulations. Nevertheless, we will compare our formulations against all three of their approaches for completeness.

Algorithm 2: Valid inequalities for `MIN DOUBLE`

candidates := $\emptyset$
if $\mathcal{K}_{K+2} = \emptyset$ **then**
  $y_{K+1} = 1$
  found := false
  foreach $\mathcal{V}^C = \mathcal{V}^{C_1} \cup \mathcal{V}^{C_2}$, $C_1, C_2 \in \mathcal{K}_{K+1}$ *s.t.* $|\mathcal{V}^{C_1} \cap \mathcal{V}^{C_2}| = K$ **do**
    if $\exists\, v \in \mathcal{V} \setminus \mathcal{V}^C$ *s.t.* $|\mathcal{N}(v) \cap \mathcal{V}^C| \geq K+1$ **then**
      candidates.add($(\mathcal{V}^C, v)$)
      found := true
    end
  end
  if *not found* **then**
    $y_{K+2} = 1$
  else
    found := false
    foreach $(\mathcal{V}^C, v) \in$ *candidates* **do**
      if $\exists\, v' \in \mathcal{V} \setminus (\mathcal{V}^C \cup \{v\})$ *s.t.* $|\mathcal{N}(v') \cap (\mathcal{V}^C \cup \{v\})| \geq K+1$ **then**
        found := true
        break
      end
    end
    if *not found* **then**
      $y_{K+2} + y_{K+3} \geq 1$
    end
  end
else
  found := false
  foreach $C \in \mathcal{K}_{K+2}$ **do**
    if $\exists\, v \in \mathcal{V} \setminus \mathcal{V}^C$ *s.t.* $|\mathcal{N}(v) \cap \mathcal{V}^C| \geq K+1$ **then**
      candidates.add($(\mathcal{V}^C, v)$)
      found := true
    end
  end
  if *not found* **then**
    $y_{K+1} + y_{K+2} \geq 1$
  else
    found := false
    foreach $(\mathcal{V}^C, v) \in$ *candidates* **do**
      if $\exists\, v' \in \mathcal{V} \setminus (\mathcal{V}^C \cup \{v\})$ *s.t.* $|\mathcal{N}(v') \cap (\mathcal{V}^C \cup \{v\})| \geq K+1$ **then**
        found := true
        break
      end
    end
    if *not found* **then**
      $y_{K+1} + y_{K+2} + y_{K+3} \geq 1$
    end
  end
end

Algorithm 3: Witnessed-based hybrid decomposition.

optimal := false
while *not optimal* **do**
  solve ($\mathbb{MP}2$) and get solution $(\hat{\kappa}, \hat{w}, \hat{y})$, solve ($\mathbb{SP}2$) with $(\hat{\kappa}, \hat{w}, \hat{y})$
  if ($\mathbb{SP}2$) *infeasible* **then**
    generate a cycle C in $\hat{w}$, add (13) to ($\mathbb{MP}2$)
  else
    let $\hat{r}$ be an optimal solution of ($\mathbb{SP}2$)
    accept $(\hat{\kappa}, \hat{w}, \hat{y}, \hat{r})$, optimal := true
  end
end

Table 1: Random instances.

n	D	# Instances	# Feasible	# Infeasible	Mean # Doubles
20	0.3	3	1	2	14
	0.4	3	3	0	3
	0.5	3	3	0	1
25	0.3	3	2	1	6
	0.4	3	3	0	1
	0.5	3	3	0	1
30	0.3	3	3	0	4
	0.4	3	3	0	1
	0.5	3	3	0	1
35	0.3	3	3	0	3
	0.4	3	3	0	1
	0.5	3	3	0	1
Total		36	33	3	

Table 2: Synthetic instances.

n	# Instances	Mean # Doubles
25	9	2
30	9	3
35	9	4
Total	27	

Algorithm 4: Cycle Constraint Generation ($\mathbb{CCG}$) Procedure

Input: $q \in \mathbb{N}$ with $q \geq 2$
$\mathcal{C}_q = \{C \in \mathcal{C} : |\mathcal{V}^C| \leq q\}$
MP := $\left\{ \min \sum_{v \in \mathcal{V}} y_v - K : (17c), (17d), \text{ and } \sum_{\{i,j\} \in \mathcal{E}^C} p_{ji} \leq |\mathcal{V}^C| - 1 \ \forall\, C \in \mathcal{C}_q \right\}$
optimal := false
while *not optimal* **do**
  solve MP and get solution$(\bar{p}, \bar{y}, \bar{\kappa})$
  C := SeparateCycle$(\bar{p})$
  if C *is found* **then**
    add $\sum_{\{i,j\} \in \mathcal{E}^C} p_{ji} \leq |\mathcal{V}^C| - 1$ to MP
  else
    accept $(\bar{p}, \bar{y}, \bar{\kappa})$
    optimal := true
  end
end

C Summary of the paper

Table 3: Summary of the formulations for `MIN DOUBLE`.

`MIN DOUBLE`: Minimize the number of vertices with exactly K adjacent predecessors in the order.						
LITERATURE	Variables	Domain	Number	Constraints	Number	Comments
Cycles ($\mathbb{CYCLES}$)	y_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	linear ordering	$\lvert\mathcal{V}\rvert^2 + \lvert\mathcal{V}\rvert^2 \times \lvert\mathcal{E}\rvert$	
	κ_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	clique selection	1	
	p_{uv}	$\{0,1\}$	$\lvert\mathcal{V}\rvert^2$	linking	$\lvert\mathcal{V}\rvert$	
Ranks ($\mathbb{RANKS}$)	y_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	linear ordering	$\lvert\mathcal{E}\rvert$	
	κ_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	clique selection	1	
	p_{uv}	$\{0,1\}$	$2\lvert\mathcal{E}\rvert$	linking	$\lvert\mathcal{V}\rvert$	
	r_v	$[n-1]$	n			
Cycle cut generation ($\mathbb{CCG}$)	y_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	clique selection	1	Generate cycles in a
	κ_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	linking	$\lvert\mathcal{V}\rvert$	branch-and-cut
	p_{uv}	$\{0,1\}$	$2\lvert\mathcal{E}\rvert$	cycle breaking cuts		procedure
NEW						
Vertex-rank IP ($\mathbb{IP}^{\mathbb{VR}}$)	y_r	$\{0,1\}$	n	1-1 assignment	$2n$	
	x_{vr}	$\{0,1\}$	$\lvert\mathcal{V}\rvert^2$	clique	n^2	
				fixing	$K+1$	
				linking	$2(\lvert\mathcal{V}\rvert \times n - K - 1)$	
CP Rank ($\mathbb{CP}^{\mathbb{RANK}}$)	y_v	$\{0,1\}$	n	AllDifferent	1	
	r_v	$[n-1]$	n	clique	$\lvert\mathcal{V}\rvert(\lvert\mathcal{V}\rvert-1)/2 - \lvert\mathcal{E}\rvert$	
				logical	$\lvert\mathcal{V}\rvert$	
CP Vertex ($\mathbb{CP}^{\mathbb{VERTEX}}$)	y_r	$\{0,1\}$	n	AllDifferent	1	
	v_r	$[\lvert V\rvert-1]$	n	clique	$K(K+1)/2$	
				fixing	$K+1$	
				logical	$n-K-1$	
CP Combined ($\mathbb{CP}^{\mathbb{COMBINED}}$)	y_r	$\{0,1\}$	n	inverse	1	Combines CP Rank
	r_v	$[n-1]$	n	clique	$(\lvert\mathcal{V}\rvert(\lvert\mathcal{V}\rvert-1) + K(K+1))/2 - \lvert\mathcal{E}\rvert$	
	v_r	$[\lvert V\rvert-1]$	n	logical	$\lvert\mathcal{V}\rvert + n - K - 1$	
				fixing	$K+1$	
Naive Decomposition	y_r	$\{0,1\}$	n	fixing	$K+1$	MP (IP): fixes doubles
	v_r	$[\lvert V\rvert-1]$	n	AllDifferent	1	SP (CP): finds an order
				clique	$K(K+1)/2$	
				fixing and logical	n	
Witness-based Decomposition	y_v	$\{0,1\}$	n	clique selection	1	MP (IP): fixes doubles,
	κ_v	$\{0,1\}$	$\lvert\mathcal{V}\rvert$	clique witness	$\lvert\mathcal{V}\rvert(\lvert\mathcal{V}\rvert-1)/2 + \lvert\mathcal{E}\rvert$	first clique, witnesses
	w_{uv}	$\{0,1\}$	$2\lvert\mathcal{E}\rvert$	witness	$\lvert\mathcal{V}\rvert$	SP (CP): finds an order
	r_v	$[n-1]$	n	AllDifferent	1	($\mathbb{EF}$) combines MP and
				rank	$\lvert\mathcal{V}\rvert + 2\lvert\mathcal{E}\rvert$	SP into one CP

D Numerical Comparison of ($\mathbb{CCG}$) and Witness-Based Decomposition

In this section, we present a brief numerical comparison of ($\mathbb{CCG}$) and the witness-based decomposition on select random instances with more than one double vertex in the optimal solution. As mentioned in Section 4.3, the formulations are similar in that they both incorporate ideas from linear ordering formulations by defining variables on the graph edges and adding cycle breaking cuts in an iterative manner. The key difference in the formulations is in their treatment of the initial clique, while ($\mathbb{CCG}$) enumerates all possible ordered K-cliques that can be extended to a $(K+1)$-clique and creates a variable for each clique, the witness-based decomposition defines indicator variables for the initial clique and constrains the vertices in the first $(K+1)$ positions to form a $(K+1)$-clique. In particular, the exact position of these vertices within the clique is not assigned. Table 4 presents the results of running both formulations when the initial K-clique has been fixed to an optimal one.

The columns of Table 4 are:

- "Time": The solution time in seconds if the instance is solved in the given time limit, "TL" if the instance hit the time limit.
- "BB Nodes": The number of branch-and-bound nodes explored (for the IP formulations); exact if it is less than one million, lower bound rounded to the closest million otherwise where a single decimal point is used up to ten million for better accuracy.
- " # Cuts": The number of cuts added in the branch-and-cut procedure.
- "n" $\in \{20, 25, 30, 35\}$: The number of vertices in the input graph.
- "D"$\in \{0.3, 0.4, 0.5\}$: The edge density of the input graph.
- "Inst.": The assigned instance number from $\{1, 2, 3\}$ for each (n, D) combination.
- "Obj": Optimal objective value of the instance; $+\infty$ if it is not solved by any method.

As expected, the witness-based decomposition is able to solve four of the six instances faster than without the clique fixing, ($\mathbb{CCG}$) however is unable to solve any instance within the time limit.

Table 4: Results of ($\mathbb{CCG}$) and the witness-based decomposition on selected random instances with an optimal first clique fixed.

				($\mathbb{CCG}$)			Witness-based Decomp.		
n	D	Inst.	Obj.	Time	BB Nodes	# Cuts	Time	BB Nodes	# Cuts
20	0.4	1	2	TL	>1.5M	827	**1.53**	6951	245
		2	3	TL	>1.1M	1337	**34.43**	80141	533
		3	4	TL	>1M	1605	**270.00**	370348	1522
25	0.3	2	5	TL	>1.7M	485	**10.51**	28141	210
		3	7	TL	>1.1M	1015	TL	>1.2M	985
	0.4	2	2	TL	184450	10813	TL	344436	9140

We also compare the number of initial cliques considered by the master problem during the

witness-based decomposition and the number of cliques enumerated for (ℂℂ𝔾). That is, the unique initial clique candidates the master problem of the witness-based decomposition passes to the subproblem and the number of clique variables, κ_c, created in (ℂℂ𝔾). We focus on the three instances for which the witness-based decomposition was able to find an optimal solution within the time limit. The results are presented in Table 5, which has similar columns to Table 4 except gives the number of cliques generated or enumerated in each method.

Table 5: Comparison of the number of cliques considered by (ℂℂ𝔾) and the witness-based decomposition.

n	D	Inst.	Obj.	Cliques (ℂℂ𝔾)	Cliques Witness-based Decomp.
20	0.4	1	2	56	18
		2	3	38	10
25	0.3	2	5	27	8

We observe that the witness-based decomposition considers far fewer cliques than (ℂℂ𝔾). This may help to explain the success of witness-based decomposition over (ℂℂ𝔾).

E Proof of Proposition 1

We prove the validity of the inequality $y_v \leq 1 - \kappa_v$ for (𝔼𝔽).

Without loss of generality, we fix $v \in \mathcal{V}$. If $\kappa_v = 0$, the inequality is redundant, since y_v is binary. If $\kappa_v = 1$, we would like $y_v = 0$, since a vertex in the initial clique is not a double. By constraint (14e), when $\kappa_v = 1$, v will have $y_v + K$ witnesses. Thus, since we are minimizing the number of doubles, the objective function will force $y_v = 0$. Meaning we have $y_v = 0$ when $\kappa_v = 1$, and so the inequality holds.

F `MIN DOUBLE` Results

We provide the following tables[5]:

- Table 6 compares the IP formulations, including the newly proposed formulation (𝕀ℙ), as well as the existing formulations from the literature the cycle cut generation procedure (ℂℂ𝔾), the cycles formulation (ℂ𝕐ℂ𝕃𝔼𝕊), and the ranks formulation (ℝ𝔸ℕ𝕂𝕊) on random instances.
- Table 7 compares the four novel CP formulations ($\mathbb{CP}^{\mathbb{VERTEX}}$) with and without valid inequalities, ($\mathbb{CP}^{\mathbb{RANK}}$), ($\mathbb{CP}^{\mathbb{COMBINED}}$) with and without valid inequalities, and (𝔼𝔽) on random instances.
- Table 8 compares the naive decomposition with and without the valid inequalities.
- Table 9 compares the witness-based decomposition under different options, initializing (𝕄ℙ2) with 2- and 3-cycle breaking, and generating cuts using an IIS or the cycle separator.

[5]The results tables for the pseudo-protein instances are available upon request.

- Table 10 compares the best performing formulations from the previous tables, namely ($\mathbb{CP}^{\mathbb{VERTEX}}$) with valid inequities, ($\mathbb{CP}^{\mathbb{COMBINED}}$) with valid inequities, the naive decomposition with valid inequalities, and the witness-based decomposition with 2-cycle breaking and 3-cycle breaking added to ($\mathbb{MP}2$) and using cycle separator.
- Table 11 compares the best performing formulations on synthetic instances.
- Table 12 compares the best performing formulations on feasible random instances with $K = 4$.
- Table 13 compares the best performing formulations on feasible random instances with $K = 5$.

For the random instances, we have:

- "n" $\in \{20, 25, 30, 35\}$: The number of vertices in the input graph.
- "D"$\in \{0.3, 0.4, 0.5\}$: The edge density of the input graph.
- "Inst.": The assigned instance number from $\{1, 2, 3\}$ for each (n, D) combination.
- "Obj": Optimal objective value of the instance; $+\infty$ if it is not solved by any method.

For the synthetic instances, we have:

- "n" $\in \{20, 25, 30, 35\}$: The number of vertices in the input graph.
- "Doubles" $\in \{\lceil 0.1 \times n \rceil, \lceil 0.1 \times n \rceil + 1, \lceil 0.1 \times n \rceil + 2\}$: The upper bound on the number of doubles in the input graph, there may be less depending on the noise edges.
- "Noise" : The number of extra edges added to the input graph.
- "Obj": Optimal objective value of the instance; $+\infty$ if it is not solved by any method.

We also have:

- "Time": Solution time in seconds if the instance is solved in the given time limit, "TL" if the instance hit the time limit.
- "IIS Time": The time require for the conflict refiner in CP Optimizer to find an IIS.
- "BB Nodes": The number of branch-and-bound nodes explored (for the IP formulations); exact if it is less than one million, lower bound rounded to the closest million otherwise where a single decimal point is used up to ten million for a better accuracy.
- "Ch.Pts.": The number of choice points (for the CP formulations); the number convention is the same as the "BB Nodes".
- " # Cuts": The number of cuts added in the branch-and-cut procedure.

Table 6: Results for IP formulations on random instances.

				(IP)		(CCG)			(CYCLES)		(RANKS)	
n	D	Inst.	Obj.	Time	BB Nodes	Time	BB Nodes	# cuts	Time	BB Nodes	Time	BB Nodes
20	0.3	1	14	TL	453417	**0.01**	0	0	0.03	0	0.01	0
		2	$+\infty$	0.06	0	**0.00**	0	0	0.00	0	0.00	0
		3	$+\infty$	43.16	15086	**0.08**	258	43	0.14	20	0.15	398
	0.4	1	2	**5.74**	3028	TL	934500	921	89.22	20984	TL	>1.1M
		2	3	**25.53**	10890	TL	787149	1172	160.21	43842	TL	>1.1M
		3	4	**30.28**	12493	TL	764550	1885	279.12	91610	TL	>1M
	0.5	1	1	**0.09**	0	TL	385745	4450	TL	217051	TL	>1.3M
		2	1	**0.09**	0	TL	402470	4108	TL	205948	TL	>1.5M
		3	1	**0.09**	0	TL	426618	4108	TL	208035	TL	>1.2M
25	0.3	1	$+\infty$	0.11	0	0.00	0	0	0.00	0	**0.00**	0
		2	5	**80.57**	11650	TL	>1.2M	629	549.19	57399	TL	735923
		3	7	581.19	95937	TL	792761	1307	**363.11**	36138	TL	690563
	0.4	1	1	**0.17**	0	TL	328277	5389	TL	74699	TL	998600
		2	2	**35.83**	8983	TL	292200	6502	TL	75560	TL	953026
		3	1	**0.18**	0	TL	342172	5299	TL	69395	TL	878376
	0.5	1	1	**0.21**	0	TL	143260	16695	TL	80320	TL	>1.1M
		2	1	**0.21**	0	TL	127055	17654	TL	81223	TL	>1.1M
		3	1	**0.21**	0	TL	142009	17796	TL	87235	TL	993935
30	0.3	1	3	**160.00**	20239	TL	369653	4186	TL	31633	TL	895019
		2	4	**164.67**	20843	TL	327500	4050	TL	26437	TL	756744
		3	4	TL	74413	TL	262484	4662	TL	24985	TL	745650
	0.4	1	1	**0.49**	0	TL	149648	14657	TL	26089	TL	>1M
		2	1	**0.45**	0	TL	113121	19578	TL	29055	TL	824766
		3	1	**0.51**	0	TL	117574	17796	TL	42971	TL	871401
	0.5	1	1	**0.65**	0	TL	66740	31222	TL	29502	TL	936367
		2	1	**0.57**	0	TL	55952	32607	TL	42067	TL	>1M
		3	1	**0.59**	0	TL	68639	34761	TL	29835	TL	>1M
35	0.3	1	3	TL	70945	TL	133890	13011	TL	12328	TL	652591
		2	3	TL	52051	TL	149991	11774	TL	11495	TL	713065
		3	3	**655.51**	46969	TL	137145	12610	TL	9883	TL	724207
	0.4	1	1	**1.00**	0	TL	50158	36286	TL	12080	TL	797966
		2	1	**0.93**	0	TL	69639	34443	TL	16788	TL	943108
		3	1	**0.92**	0	TL	61730	35345	TL	11998	TL	902507
	0.5	1	1	**1.16**	0	TL	47893	43789	TL	19223	TL	>1M
		2	1	**1.13**	0	TL	48441	43845	TL	17275	TL	761910
		3	1	**1.12**	0	TL	48151	43090	TL	18154	TL	745675

Table 7: Results for the CP formulations on random instances.

				($\mathbb{CP}^{\text{VERTEX}}$)		($\mathbb{CP}^{\text{VERTEX}}$) (with VI)		($\mathbb{CP}^{\text{RANK}}$)		($\mathbb{CP}^{\text{COMBINED}}$)		($\mathbb{CP}^{\text{COMBINED}}$) (with VI)		($\mathbb{EF}$)	
n	D	Inst.	Obj.	Time	Ch.Pts.	Time	Ch.Pts.	Time	Ch.Pts.	Time	Ch.Pts.	Time	Ch.Pts.	Time	Ch.Pts.
20	0.3	1	14	TL	>19M	TL	>17M	TL	>34M	TL	>13M	TL	>14M	**1.80**	114233
		2	$+\infty$	1.16	26095	1.39	28232	**0.00**	0	0.00	0	**0.00**	0	0.00	0
		3	$+\infty$	39.82	856311	34.94	739690	TL	>23M	8.46	200036	8.27	209521	**2.88**	0
	0.4	1	2	1.92	37797	0.03	451	TL	>30M	2.36	50296	**0.02**	455	TL	>35M
		2	3	8.95	161665	0.03	436	TL	>39M	3.55	77317	**0.02**	446	TL	>42M
		3	4	15.78	332463	**7.12**	146090	TL	>45M	10.08	223677	9.10	201655	TL	>39M
	0.5	1	1	0.02	186	**0.02**	186	0.02	1028	0.02	453	0.02	453	0.61	30411
		2	1	0.02	160	0.02	160	**0.01**	978	0.03	463	0.03	463	0.16	8222
		3	1	**0.02**	228	0.02	228	0.02	1543	0.02	445	0.02	445	0.14	7412
25	0.3	1	$+\infty$	TL	>13M	TL	>13M	**0.00**	0	0.02	6	0.01	12	0.00	0
		2	5	TL	>13M	TL	>12M	TL	>38M	TL	>12M	TL	>12M	TL	>34M
		3	7	TL	>13M	TL	>13M	TL	>42M	TL	>13M	TL	>12M	TL	>37M
	0.4	1	1	0.09	1192	0.05	447	**0.01**	1122	0.04	470	0.05	470	0.54	27543
		2	2	3.39	49464	0.05	465	993.79	>45M	6.93	100309	**0.03**	458	TL	>35M
		3	1	0.24	3814	0.04	451	**0.02**	1613	0.04	450	0.04	450	5.17	213898
	0.5	1	1	0.08	1198	0.04	467	**0.00**	238	0.04	471	0.05	471	0.37	16623
		2	1	0.05	529	0.04	459	**0.01**	1123	0.05	469	0.04	469	0.87	37055
		3	1	0.04	511	0.04	462	**0.02**	1502	0.05	460	0.04	460	0.37	15621
30	0.3	1	3	31.34	358648	0.07	485	TL	>51M	25.72	296148	**0.06**	469	TL	>30M
		2	4	35.81	423625	15.80	138956	TL	>60M	48.10	643116	**12.84**	164305	TL	>34M
		3	4	353.13	>3.7M	430.65	>4.5M	TL	>56M	**258.87**	>3M	588.08	>6.9M	TL	>27M
	0.4	1	1	0.32	3185	0.08	478	**0.02**	1138	0.07	478	0.07	478	1.05	49638
		2	1	0.75	7566	0.08	468	**0.02**	1719	0.06	458	0.06	458	0.55	25189
		3	1	0.15	1680	0.07	467	**0.02**	1720	0.06	476	0.07	476	4.43	177270
	0.5	1	1	0.05	169	0.05	169	**0.02**	1138	0.06	475	0.08	475	0.36	16984
		2	1	0.06	207	0.05	207	**0.02**	1148	0.04	181	0.04	181	0.64	26581
		3	1	0.06	203	0.05	203	**0.02**	1740	0.08	471	0.07	471	1.23	57488
35	0.3	1	3	388.29	>3M	258.45	>1.6M	TL	>58M	985.83	>9.3M	**209.41**	>2.1M	TL	>35M
		2	3	100.39	748258	**0.12**	492	TL	>62M	145.04	>1.4M	0.12	503	TL	>32M
		3	3	118.75	934246	0.13	480	TL	>58M	175.48	>1.5M	**0.12**	490	TL	>27M
	0.4	1	1	0.13	616	0.13	487	**0.03**	1852	0.12	489	0.12	489	1.38	54887
		2	1	0.27	2208	0.13	477	**0.03**	1233	0.12	492	0.11	492	1.38	57876
		3	1	0.09	208	0.11	208	**0.03**	1855	0.12	499	0.12	499	3.99	138625
	0.5	1	1	0.10	205	0.10	205	**0.02**	1123	0.09	171	0.09	171	0.53	21071
		2	1	0.08	88	0.07	88	**0.04**	1842	0.08	168	0.09	168	0.42	18364
		3	1	0.09	170	0.10	170	**0.03**	1776	0.11	501	0.13	501	0.72	31121

Table 8: Results for the naive decomposition with and without valid inequalities (VI) on random instances.

				Naive Decomp. (without VI)				Naive Decomp. (with VI)				
n	D	Inst.	Obj.	Time	IIS time	BB Nodes	# Cuts	Time	IIS time	BB Nodes	# Cuts	VI added
20	0.3	1	14	TL	992.91	0	6	TL	991.49	0	6	$y_{K+1} + y_{K+2} \geq 1$
		2	$+\infty$	**302.18**	285.99	1	56	597.65	549.67	1	106	$y_{K+1} = 1$, $y_{K+2} = 1$
		3	$+\infty$	TL	994.63	11	23	TL	970.16	0	22	$y_{K+1} = 1$, $y_{K+2} = 1$
	0.4	1	2	14.28	12.87	0	1	**0.23**	0	0	0	$y_{K+1} + y_{K+2} \geq 1$
		2	3	48.87	45.35	0	2	**0.03**	0	0	0	$y_{K+1} = 1$, $y_{K+2} = 1$
		3	4	57.65	55.11	0	3	**48.10**	44.18	0	1	$y_{K+1} = 1$, $y_{K+2} = 1$
	0.5	1	1	0.08	0	0	0	**0.08**	0	0	0	
		2	1	0.03	0	0	0	**0.03**	0	0	0	
		3	1	0.03	0	0	0	**0.03**	0	0	0	
25	0.3	1	$+\infty$	TL	983.07	0	1	TL	980.37	0	1	$y_{K+1} + y_{K+2} \geq 1$
		2	5	TL	734.242	0	4	TL	655.09	0	2	$y_{K+1} = 1$, $y_{K+2} + y_{K+3} \geq 1$
		3	7	TL	930.99	0	5	TL	927.70	0	4	$y_{K+1} = 1$, $y_{K+2} = 1$
	0.4	1	1	0.19	0	0	0	**0.16**	0	0	0	
		2	2	59.29	56.01	0	1	**0.50**	0	0	0	$y_{K+1} + y_{K+2} \geq 1$
		3	1	0.11	0	0	0	**0.09**	0	0	0	
	0.5	1	1	0.08	0	0	0	**0.07**	0	0	0	
		2	1	0.07	0	0	0	**0.07**	0	0	0	
		3	1	**0.06**	0	0	0	0.06	0	0	0	
30	0.3	1	3	209.49	206.41	0	2	**0.86**	0	0	0	$y_{K+1} = 1$, $y_{K+2} + y_{K+3} \geq 1$
		2	4	267.44	260.88	0	3	**204.33**	187.15	0	1	$y_{K+1} = 1$, $y_{K+2} = 1$
		3	4	TL	967.79	0	5	TL	937.77	0	2	$y_{K+1} + y_{K+2} \geq 1$
	0.4	1	1	**0.45**	0	0	0	0.57	0	0	0	
		2	1	**0.30**	0	0	0	0.40	0	0	0	
		3	1	**0.12**	0	0	0	0.14	0	0	0	
	0.5	1	1	**0.12**	0	0	0	0.16	0	0	0	
		2	1	**0.11**	0	0	0	0.15	0	0	0	
		3	1	**0.12**	0	0	0	0.14	0	0	0	
35	0.3	1	3	TL	910.55	0	2	TL	944.54	0	1	$y_{K+1} + y_{K+2} \geq 1$
		2	3	637.82	596.94	0	2	**0.61**	0	0	0	$y_{K+1} = 1$, $y_{K+2} + y_{K+3} \geq 1$
		3	3	519.48	469.57	0	2	**0.86**	0	0	0	$y_{K+1} = 1$, $y_{K+2} = 1$
	0.4	1	1	**0.46**	0	0	0	0.52	0	0	0	
		2	1	**0.20**	0	0	0	0.24	0	0	0	
		3	1	**0.28**	0	0	0	0.36	0	0	0	
	0.5	1	1	**0.20**	0	0	0	0.26	0	0	0	
		2	1	**0.20**	0	0	0	0.26	0	0	0	
		3	1	**0.20**	0	0	0	0.25	0	0	0	

Table 9: Results for the witness-based decomposition on random instances, the column headers define the options used to solve each instance. We include only the times for those methods which are clearly outperformed.

				IIS no cycle breaking	IIS 2-cycle breaking	IIS 2-cycle & 3-cycle breaking				cycle separation 2-cycle & 3-cycle breaking		
n	D	Inst.	Obj.	Time	Time	Time	IIS time	BB Nodes	# Cuts	Time	BB Nodes	# Cuts
20	0.3	1	14	0.82	0.22	0.11	0.04	29	2	**0.05**	30	1
		2	$+\infty$	0.00	0.00	**0.00**	0.00	0	0	0.00	0	0
		3	$+\infty$	1.27	1.11	0.49	0.39	393	18	**0.09**	250	30
	0.4	1	2	503.49	322.40	188.39	23.55	241864	764	**72.06**	136729	1086
		2	3	TL	TL	826.28	44.20	868805	1309	**765.68**	>1M	1858
		3	4	TL	TL	TL	56.34	819500	1557	TL	850346	2865
	0.5	1	1	TL	19.35	0.02	0.00	0	0	**0.02**	0	0
		2	1	0.86	2.51	**0.46**	0.38	9	12	0.51	534	290
		3	1	67.42	TL	899.36	167.57	483400	4726	**0.30**	250	191
25	0.3	1	$+\infty$	0.00	**0.00**	0.00	0.00	0	0	0.00	0	0
		2	5	265.95	182.43	**102.67**	15.28	147275	448	128.44	214519	428
		3	7	TL	TL	TL	17.55	>1.2M	653	TL	814100	1561
	0.4	1	1	49.47	22.69	1.32	1.18	19	28	**0.05**	13	12
		2	2	TL	TL	TL	390.42	76940	10783	TL	435200	13778
		3	1	TL	822.20	74.26	63.96	12997	1917	**2.25**	3508	1213
	0.5	1	1	14.83	5.61	33.98	31.50	1285	997	**1.53**	1128	900
		2	1	144.43	**0.02**	0.10	0.05	0	2	0.05	0	2
		3	1	1.69	74.92	7.33	6.78	140	231	**0.13**	0	3
30	0.3	1	3	TL	TL	TL	395.67	96929	8766	TL	110687	16086
		2	4	TL	TL	TL	227.83	164700	5215	TL	120326	14070
		3	4	TL	TL	TL	129.15	412610	3121	TL	271708	6599
	0.4	1	1	TL	2.44	TL	431.60	60001	10376	**1.16**	983	874
		2	1	TL	TL	111.46	101.08	3962	2389	**2.16**	1076	1398
		3	1	47.21	7.55	0.48	0.34	3	10	**0.30**	162	226
	0.5	1	1	11.14	1.97	TL	784.87	37527	16374	**0.25**	30	57
		2	1	4.93	2.63	0.88	0.65	7	19	**0.27**	19	41
		3	1	**0.19**	40.20	5.71	5.23	105	130	0.27	63	69
35	0.3	1	3	TL	TL	TL	389.40	81254	8553	TL	200157	10093
		2	3	TL	TL	TL	330.50	102679	6448	TL	144100	12350
		3	3	TL	TL	TL	384.89	102852	8311	TL	106642	23345
	0.4	1	1	TL	TL	TL	670.87	48007	13694	TL	104470	26293
		2	1	TL	TL	1.24	0.95	14	21	**0.39**	173	227
		3	1	TL	25.52	TL	734.19	56599	13849	**0.61**	230	339
	0.5	1	1	1.10	19.22	6.28	5.64	116	133	**0.16**	0	3
		2	1	11.84	20.52	159.28	144.22	3981	2899	**0.89**	233	389
		3	1	2.51	25.54	3.20	2.74	41	61	**1.36**	389	671

Table 10: Results for the best performing formulations on random instances.

				($\mathbb{IP}$)		($\mathbb{CP}^{\text{VERTEX}}$) (with VI)		($\mathbb{CP}^{\text{COMBINED}}$) (with VI)		Naive Decomp. (with VI)			Witness-based Decomp. cycle separation 2-cycle & 3-cycle breaking		
n	D	Inst.	Obj.	Time	BB Nodes	Time	Ch.Pts.	Time	Ch.Pts.	Time	BB Nodes	#Cuts	Time	BB Nodes	#Cuts
20	0.3	1	14	956.68	453417	932.80	>17M	TL	>14M	TL	0	6	**0.05**	30	1
		2	$+\infty$	0.06	0	1.39	28232	**0.00**	0	597.65	1	106	0.00	0	0
		3	$+\infty$	43.16	15086	34.94	739690	8.27	209521	TL	0	22	**0.09**	250	30
	0.4	1	2	5.74	3028	0.03	451	**0.02**	455	0.23	0	0	72.06	136729	1086
		2	3	25.53	10890	0.03	436	**0.02**	446	0.03	0	0	765.68	>1M	1858
		3	4	30.28	12493	**7.12**	146090	9.10	201655	48.10	0	1	TL	850346	2865
	0.5	1	1	0.09	0	**0.02**	186	0.02	453	0.08	0	0	0.02	0	0
		2	1	0.09	0	**0.02**	160	0.03	463	0.03	0	0	0.51	534	290
		3	1	0.09	0	**0.02**	228	0.02	445	0.03	0	0	0.30	250	191
25	0.3	1	$+\infty$	0.11	0	TL	>13M	0.01	12	TL	0	1	**0.00**	0	0
		2	5	**80.57**	11650	TL	>12M	TL	>12M	TL	0	2	128.44	214519	428
		3	7	**581.19**	95937	TL	>13M	TL	>12M	TL	0	4	TL	814100	1561
	0.4	1	1	0.17	0	**0.05**	447	0.05	470	0.16	0	0	0.05	13	12
		2	2	35.83	8983	0.05	465	**0.03**	458	0.50	0	0	TL	435200	13778
		3	1	0.18	0	0.04	451	**0.04**	450	0.09	0	0	2.25	3508	1213
	0.5	1	1	0.21	0	**0.04**	467	0.05	471	0.07	0	0	1.53	1128	900
		2	1	0.21	0	0.04	459	**0.04**	469	0.07	0	0	0.05	0	2
		3	1	0.21	0	**0.04**	462	0.04	460	0.06	0	0	0.13	0	3
30	0.3	1	3	160.00	20239	0.07	485	**0.06**	469	0.86	0	0	TL	110687	16086
		2	4	164.67	20843	15.80	138956	**12.84**	164305	204.33	0	1	TL	120326	14070
		3	4	TL	74413	**430.65**	>4.5M	588.08	>6.9M	TL	0	2	TL	271708	6599
	0.4	1	1	0.49	0	0.08	478	**0.07**	478	0.57	0	0	1.16	983	874
		2	1	0.45	0	0.08	468	**0.06**	458	0.40	0	0	2.16	1076	1398
		3	1	0.51	0	0.07	467	**0.07**	476	0.14	0	0	0.30	162	226
	0.5	1	1	0.65	0	**0.05**	169	0.08	475	0.16	0	0	0.25	30	57
		2	1	0.57	0	0.05	207	**0.04**	181	0.15	0	0	0.27	19	41
		3	1	0.59	0	**0.05**	203	0.07	471	0.14	0	0	0.27	63	69
35	0.3	1	3	TL	70945	258.45	>1.6M	**209.41**	>2.1M	TL	0	1	TL	200157	10093
		2	3	TL	52051	**0.12**	492	0.12	503	0.61	0	0	TL	144100	12350
		3	3	655.51	46969	0.13	480	**0.12**	490	0.86	0	0	TL	106642	23345
	0.4	1	1	1.00	0	0.13	487	**0.12**	489	0.52	0	0	TL	104470	26293
		2	1	0.93	0	0.13	477	**0.11**	492	0.24	0	0	0.39	173	227
		3	1	0.92	0	**0.11**	208	0.12	499	0.36	0	0	0.61	230	339
	0.5	1	1	1.16	0	0.10	205	**0.09**	171	0.26	0	0	0.16	0	3
		2	1	1.13	0	**0.07**	88	0.09	168	0.26	0	0	0.89	233	389
		3	1	1.12	0	**0.10**	170	0.13	501	0.25	0	0	1.36	389	671

Table 11: Results for the best performing formulations on synthetic instances.

				($\mathbb{CCG}$)			($\mathbb{CP}^{\text{VERTEX}}$) (with VI)		($\mathbb{CP}^{\text{COMBINED}}$) (with VI)		Naive Decomp. (with VI)			Witness-based Decomp. cycle separation 2-cycle & 3-cycle breaking		
n	Doubles	Noise	Obj.	Time	BB Nodes	# Cuts	Time	Ch.Pts.	Total Time	Ch.Pts.	Time	BB Nodes	# Cuts	Time	BB Nodes	# Cuts
25	3	3	2	82.87	132209	292	**0.25**	5989	1.57	27179	0.47	0	0	1.01	484	20
		4	2	781.13	>1M	560	**0.29**	6676	0.45	9189	0.34	0	0	2.08	1298	40
		5	2	TL	>1M	977	**0.19**	4492	0.66	17568	0.40	0	0	7.42	3750	131
	4	3	2	5.28	7007	180	45.37	966391	55.34	>1M	TL	0	1	**1.09**	257	22
		4	2	96.35	125717	423	46.76	973021	102.71	>1.7M	TL	0	1	**0.59**	344	7
		5	1	111.11	152250	250	0.61	12744	**0.60**	16244	0.62	0	0	1.92	667	38
	5	3	3	5.26	6841	70	TL	>20M	TL	>14M	TL	0	0	**0.94**	555	19
		4	3	202.42	282382	298	TL	>21M	TL	>13M	TL	0	0	**9.17**	5014	170
		5	3	TL	>1M	479	TL	>19M	TL	>18M	TL	0	0	**108.01**	34378	1494
30	3	3	2	293.07	226800	1014	**0.68**	10973	1.64	26819	0.91	0	0	2.64	1372	31
		5	2	TL	703581	1534	0.97	13645	3.36	56883	**0.84**	0	0	29.56	25076	174
		6	2	TL	727815	974	**1.02**	15450	1.07	21795	1.07	0	0	96.23	43878	1012
	4	3	3	166.98	192681	314	TL	>12M	TL	>9.5M	TL	0	0	**11.52**	1873	235
		5	3	TL	>1M	541	TL	>12M	TL	>9.4M	TL	0	0	**21.73**	13673	234
		6	3	TL	>1M	641	TL	>12M	TL	>13M	TL	0	0	**49.62**	45561	376
	5	3	4	323.10	382120	261	TL	>14M	TL	>12M	TL	0	0	**9.71**	4323	157
		5	4	TL	>1M	579	TL	>14M	TL	>12M	TL	0	0	**59.32**	70357	263
		6	3	TL	982200	569	TL	>13M	TL	>14M	TL	0	0	**23.30**	19367	187
35	4	4	4	TL	919450	441	TL	>10M	TL	>9.5M	TL	0	0	**17.53**	12017	121
		6	4	TL	511962	1028	TL	>10M	TL	>9.2M	TL	0	0	**428.53**	343809	286
		7	3	TL	459000	1277	TL	>10M	TL	>13M	TL	0	0	TL	773333	477
	5	4	4	TL	756700	402	TL	>9.9M	TL	>9M	TL	0	0	**11.77**	9761	48
		6	4	TL	741600	895	TL	>9.8M	TL	>9.2M	TL	0	0	**134.45**	117720	247
		7	4	TL	479861	1098	TL	>10M	TL	>11M	TL	0	0	**836.95**	767184	387
	6	4	4	109.11	102481	298	TL	>9.9M	TL	>10M	TL	0	0	**2.88**	1348	17
		6	4	TL	794092	599	TL	>10M	TL	>9.7M	TL	0	0	**42.49**	31154	144
		7	3	TL	482000	1333	TL	>10M	TL	>13M	TL	0	0	**35.25**	33400	172

Table 12: Results for the best performing formulations on random instances with $K = 4$.

				(IP)		(CCG)			(CP^{VERTEX}) (with VI)		($CP^{COMBINED}$) (with VI)		Naive Decomp. (with VI)			Witness-based Decomp. cycle separation 2-cycle & 3-cycle breaking		
n	D	Inst.	Obj.	Time	BB Nodes	Time	BB Nodes	#Cuts	Time	Ch.Pts.	Time	Ch.Pts.	Time	BB Nodes	#Cuts	Time	BB Nodes	#Cuts
20	0.5	1	3	198.54	83515	TL	323383	4152	97.08	>2.2M	**46.34**	>1.3M	TL	0	4	TL	698737	4096
		2	2	24.41	12257	TL	429900	3618	**0.13**	4468	0.26	9707	28.15	0	2	TL	>1M	3622
		3	4	294.51	99787	TL	443716	3565	123.19	>2.8M	**82.99**	>2.1M	TL	0	4	TL	610254	4891
	0.7	1	2	0.38	0	TL	87819	18949	**0.01**	28	0.01	36	0.04	0	0	0.06	1	3
		2	2	0.50	100	TL	90067	17007	**0.01**	102	0.01	104	0.02	0	0	32.17	23946	8097
		3	2	0.57	20	TL	90497	18369	**0.01**	32	0.01	30	0.02	0	0	0.11	69	58
25	0.4	1	2	100.53	23942	TL	255230	5083	3.23	60795	**0.65**	18463	222.89	0	2	TL	944585	1953
		3	4	191.34	39444	TL	183556	7808	**37.57**	778990	115.23	>2.5M	659.51	0	4	TL	754647	1544
	0.5	1	2	9.37	2171	TL	103080	17686	0.22	5032	**0.10**	2482	0.11	0	0	TL	452598	10506
		2	2	3.06	544	TL	98861	17595	**0.04**	518	0.15	3844	0.11	0	0	2.47	2679	1290
		3	2	117.14	27346	TL	103104	15080	**0.07**	1565	0.34	9083	43.63			TL	335431	18925
	0.7	1	2	0.71	0	TL	43425	34277	**0.01**	31	0.02	57	0.05	0	0	0.09	4	0
		2	2	1.41	30	TL	46617	35453	**0.01**	31	0.02	52	0.05	0	0	0.19	14	23
		3	2	1.09	0	TL	50974	36328	**0.01**	33	0.02	48	0.05	0	0	0.03	0	0
30	0.4	1	2	TL	117419	TL	111696	14955	**3.19**	48203	6.74	119001	TL	0	2	TL	259546	9562
		2	3	840.23	91914	TL	110217	15295	249.28	>4M	**56.69**	994726	TL	0	2	TL	128032	16278
		3	3	TL	99842	TL	96684	13686	252.12	>3.5M	**77.23**	>1.3M	TL	0	1	TL	193379	12298
	0.5	1	2	25.32	2743	TL	45286	32276	**0.04**	71	0.09	1323	0.52	0	0	27.12	13384	5019
		2	2	11.98	604	TL	47654	32034	**0.05**	626	0.47	9244	0.81	0	0	TL	115100	24688
		3	2	800.83	103399	TL	45780	28561	0.12	1596	**0.08**	1396	TL	0	1	TL	172140	25786
	0.7	1	2	1.74	0	TL	36983	46019	**0.03**	38	0.04	284	0.13	0	0	0.11	0	1
		2	2	1.55	0	TL	34086	45394	**0.03**	37	0.03	38	0.15	0	0	0.19	4	6
		3	2	1.18	0	TL	31620	49277	**0.03**	38	0.03	37	0.11	0	0	0.14	2	5
35	0.4	2	2	40.95	1573	TL	56761	29474	0.49	7500	1.33	18297	**0.27**	0	0	0.65	287	308
		3	2	811.71	56844	TL	52878	28871	0.43	4930	**0.07**	125	TL	0	1	TL	142021	21304
	0.5	1	2	15.03	731	TL	35316	41620	**0.10**	562	0.97	14074	0.23	0	0	0.37	22	35
		2	2	12.19	164	TL	34960	45237	**0.06**	115	0.40	5611	1.44	0	0	8.25	3354	3285
		3	2	10.40	150	TL	34937	44867	**0.08**	199	0.13	1308	0.80	0	0	TL	214703	22183
	0.7	1	2	5.30	40	TL	20362	47286	**0.06**	55	0.06	62	0.20	0	0	0.15	0	0
		2	2	2.87	0	TL	19120	47919	**0.06**	57	0.06	59	0.21	0	0	0.24	43	54
		3	2	4.17	0	TL	18270	47295	**0.06**	55	0.06	60	0.22	0	0	0.11	3	3

Table 13: Results for the best performing formulations on random instances with $K = 5$.

				($\mathbb{IP}$)		($\mathbb{CCG}$)			($\mathbb{CP}^{\text{VERTEX}}$) (with VI)		($\mathbb{CP}^{\text{COMBINED}}$) (with VI)		Naive Decomp. (with VI)			Witness-based Decomp. cycle separation 2-cycle & 3-cycle breaking		
n	D	Inst.	Obj.	Time	BB Nodes	Time	BB Nodes	#Cuts	Time	Ch.Pts.	Time	Ch.Pts.	Time	BB Nodes	#Cuts	Time	BB Nodes	#Cuts
20	0.7	1	2	0.52	0	TL	127066	13270	**0.01**	28	0.02	541	0.03	0	0	0.23	44	7
		2	2	209.33	117075	TL	120642	13778	0.05	1211	**0.05**	1609	969.40	0	1	TL	310055	13348
		3	2	0.86	260	TL	171232	10873	**0.01**	28	0.02	484	0.03	0	0	0.13	3	2
25	0.5	2	4	TL	183338	TL	133379	10894	**314.66**	>4.5M	559.60	>9M	TL	0	3	TL	329012	5492
	0.7	1	2	2.30	0	TL	58130	23335	0.02	147	**0.02**	56	TL	0	3	0.31	81	103
		2	2	1.65	80	TL	57317	28280	**0.02**	158	0.07	1536	0.05	0	0	0.26	4	13
		3	2	1.12	0	TL	48025	26474	**0.01**	35	0.03	533	0.05	0	0	0.42	32	33
30	0.5	1	3	TL	132475	TL	57854	21568	553.77	>5.8M	450.33	>6.6M	**0.05**	0	0	TL	158584	1445
		2	3	TL	123439	TL	66413	20646	894.57	>8.2M	**601.74**	>8.1M	TL	0	2	TL	125635	16910
	0.7	1	2	4.74	310	TL	32065	40585	**0.03**	43	0.04	44	TL:	0	2	0.17	2	3
		2	2	2.43	10	TL	30354	42925	**0.03**	79	0.08	950	0.13	0	0	36.70	9460	6970
		3	2	1.57	0	TL	27713	39782	**0.03**	42	0.04	188	0.16	0	0	0.19	7	23
35	0.5	1	2	TL	64824	TL	39674	35014	0.97	11610	0.41	5392	**0.15**	0	0	TL	86717	22700
		2	2	TL	61939	TL	43800	34937	0.92	11223	**0.20**	2673	TL	0	0	TL	148236	29015
		3	2	T;L	61253	TL	35592	37923	**1.35**	15605	1.58	22016	TL	0	0	TL	113575	17829
	0.7	1	2	8.49	14	TL	18249	45102	**0.06**	42	0.06	55	0.26	0	0	0.35	11	30
		2	2	4.30	0	TL	19194	44076	**0.06**	44	0.06	61	0.25	0	0	0.29	8	24
		3	2	3.28	0	TL	21799	42927	0.06	42	**0.06**	57	0.26	0	0	0.55	19	21

G Synthetic Instance Generation

Algorithm 5 gives the procedure to generate synthetic instances.

Algorithm 5: Synthetic Instance Generation Procedure

Input: K, *numDoubles*, *noise*, n

$y := \{0 \mid \forall r \in [n-1]\}$
$y_K := 1$
while $\sum_{r \in [n-1]} y_r < numDoubles$ **do**
  $r :=$ random integer in $[K+1, n-1]$
  if $y_r = 0$ **then**
    $y_r := 1$
  end
end
$\mathcal{E} = \emptyset$
for *all* i, j *pairs* $i, j \in [K]$ **do**
  add an edge (i, j) to $\mathcal{E}$
end
foreach $v \in [K+1, n-1]$ **do**
  if $y_v = 1$ **then**
    randomly select a subset of vertices of size K, $U \subseteq [v-1]$, and add edges (u, v) for all $u \in U$ to $\mathcal{E}$
  else
    randomly select a subset of vertices of size $K+1$, $U \subseteq [v-1]$, and add edges (u, v) for all $u \in U$ to $\mathcal{E}$
  end
end
$noiseCount := 0$
while $noiseCount < \lceil noise \times n \rceil$ **do**
  randomly select $u, v \in [K+1, n-1]$, $u \neq v$, with $y_u = 0$ and $y_v = 0$
  if $(u, v) \notin \mathcal{E}$ **then**
    add edge (u, v) to $\mathcal{E}$
    $noiseCount++$
  end
end
return $G := ([n-1], \mathcal{E})$