A Compact Mixed Integer Linear Formulation for Safe Set Problems

Pierre Hosteins

A Compact Mixed Integer Linear Formulation for Safe Set Problems

Pierre Hosteins
Univ Gustave Eiffel, IFSTTAR, COSYS, ESTAS,
F-59650 Villeneuve d'Ascq, Lille, France
e-mail: pierre.hosteins@ifsttar.fr

February 5, 2020

Abstract

The Safe Set Problem is a type of partitioning problem with particular constraints on the weight of adjacent partition components. Several theoretical results exist in the literature along with a mixed integer linear formulation. We propose a new, more compact, Mixed Integer Linear Program for the (Weighted) Safe Set Problem and Connected Safe Set Problem with a polynomial number of variables and constraints, based on a flow from an additional fictitious node. The model is enriched by symmetry-breaking considerations and a variable reduction subproblem. We test the model on a benchmark of small instances which underline the difficulty of solving the Safe Set Problem through mathematical programming approaches. We also compare it with the only existing formulation in the literature and find that our approach is usually more efficient on a set of benchmark instances.

Keywords: Graph partitioning, network control, Safe Set, Connected Safe Set, Mixed Integer Linear Programming, symmetry-breaking, variable reduction.

1 Introduction

Consider an undirected and connected graph $G = (V, E)$ with node weights $w_i \geq 0$ for $i \in V$. We call $N_i \subset V$ the set of neighbours of node $i \in V$ and define a non-empty subset $S \subset V$ which induces a subgraph $G[S]$. The function $w : 2^V \to \mathbb{R}_+$ is defined as $w(S) = \sum_{i \in S} w_i$, i.e. the total weight of a subset of nodes $S \subseteq V$. S is said to be a safe set if $S \neq \emptyset$ and for any maximally connected component C in $G[S]$ and C' in $G[V \setminus S]$ we have $w(C) \geq w(C')$ whenever an edge exists between C and C'. The Weighted Safe Set problem (WSSP) consists in finding the safe set with minimum total weight, while the simpler Safe Set Problem (SSP) considers only unit node weights. The Connected Safe Set Problem (CSSP) is a Safe Set Problem where the safe set is connected (i.e. $G[S]$ has only one component). An optimal solution of any Safe Set variant is denoted as S^*. We observe that if G is not connected, the problem decomposes completely on each of the connected components of G. Thus, we assume without loss of generality that G is connected. An example of a feasible safe set on a small graph is illustrated in Figure 1.

The SSP was first defined in [7] as a way to determine important influential communities in a social network. Another application is invoked in [13] where one looks for a subspace inside a building that can shelter the people from nearby spaces in a case of a catastrophe, i.e. a building design problem. From a more general point of view, the SSP is designed to determine the smallest subset of a network which we need to control

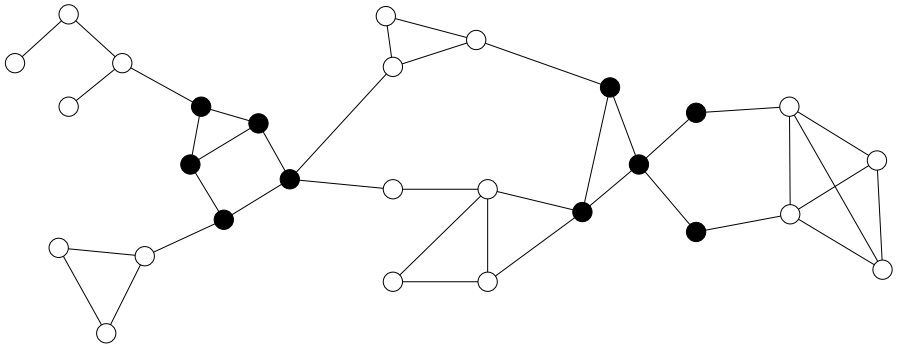

Figure 1: Example of a feasible safe set for a small instance of the SSP. The nodes in the safe set are displayed in black.

(or equip) in order to control the behaviour of (or assist) the whole network. One can see from its definition above that it can be classified as a type of graph partitioning problem, though only node weights are considered. This differentiates greatly the SSP from partitioning problems based on edge cuts. Other node-based partitioning problems exist, such as the Vertex Separator Problem [6] and the k-separator problem [9] where a subset of nodes of minimum cardinality (or weight) is removed from the graph in order to obtain (possibly balanced) disconnected components of the original graph. Another class of interdiction problem, called the Critical Node Problem [5, 15], is also linked to node-based graph partitioning as some of its versions aim at fragmenting the graph into disconnected subgraphs by deleting a subset S of nodes which is not necessarily a minimal node cut-set. Contrary to such node-cut based graph partitioning problems, no subset of nodes is removed from the graph in a Safe Set solution. Other social network problems study the influence of certain parts of the network on the whole network. A prominent example is the positive influence dominating set problem [12, 21], where we seek a minimal set of nodes S such that all other nodes in the network have at least a fraction $\rho > 0$ of their neighbours in S. Such models try to identify a minimal set of participants in the network to solicit so that we can guarantee a positive influence over the rest of the network. Such models, however, are defined based on local properties, such as a subset of neighbours for each node belonging to a certain set. The SSP is defined using more global properties based on whole connected components of a graph partition, which makes it less straightforward to cast as a Mixed Integer Linear Program (MILP) to solve it. The problem is quite recent and has not been studied extensively in the literature. Moreover mainly theoretical contributions on the subject can be found. For example, the weighted problem over trees has been shown to be NP-hard in [7, 8] while the unweighted problem is shown to be solvable in polynomial time on graphs with bounded treewidth and interval graphs in [1]. Bounds on the size of safe sets and connected safe sets are also studied in [13]. While the present manuscript was being revised, the work of [16] was published, which introduces a MILP formulation and a greedy heuristic to solve the WSSP on general graphs. The MILP has an exponential number of constraints, which are introduced at each node of the branching tree using a separation algorithm, and can be easily adapted to the connected version of the problem. The MILP and the heuristic are tested on a benchmark of instances with number of nodes ranging from 10 to 30 and different edge densities.

We propose in the following a relatively compact MILP with a polynomial number of variables and constraints to tackle the problem on general graphs. The model is quite different in its approach from the one in [16]. In Section 2 we detail our MILP formulation for the WSSP and its connected version. In Section 3 we propose a way to break some problem symmetries and solve a simpler MILP in a preprocessing step

to reduce the number of variables of our Safe Set model. In Section 4 we present the results of numerical experiments made on a benchmark of small random graphs and in Section 5 we briefly conclude. Additional results on small world graphs are provided in Appendix A. In Appendix Appendix B, we compare the results of our approach with that of [16].

2 A compact MILP formulation

In the following we define the necessary variables and constraints to provide a MILP for the (Connected) WSSP. We define a model that identifies the connected components of a given solution by using a flow circulation coming from a fictitious node 0, which will be partially absorbed by each node in the graph. Recall that a feasible safe set is a subset $S \subset V$, and we define $n \coloneqq |V|$, $m \coloneqq |E| \geq n-1$, and $W \coloneqq \sum_{i \in V} w_i$. We also define E' to be the set of directed edges corresponding to E, i.e. $E' = \{(i,j) : i \in V, j \in V \text{ and } \{i,j\} \in E\}$. Since for a general graph, it is impossible to know in advance the number of maximally connected components in the optimal partition, we have to define variables with indices ranging up to the maximal theoretical number of connected components $n-1$[1]. As a result, the variables related to potential components which are unused in the optimal solution will be 0. In order to formulate our model, we will define the following variables, corresponding to a feasible safe set S:

- x_i: binary variable equal to 1 if node $i \in S$ (i.e. is in the safe set), 0 otherwise.
- y_{ij}: binary variable equal to 1 for edge $\{i,j\} \in E$ if $i \in S$ and $j \in S$, 0 otherwise.
- y'_{ij}: binary variable equal to 1 for edge $\{i,j\} \in E$ if $i \in V \setminus S$ and $j \in V \setminus S$, 0 otherwise.
- a_{ic}: binary variable equal to 1 if node $i \in S$ is part of component $c \in \{1, \ldots, n-1\}$.
- a'_{ic}: binary variable equal to 1 if node $i \in V \setminus S$ is part of component $c \in \{1, \ldots, n-1\}$.
- t_{ic}: binary variable equal to 1 if node $i \in S$ is the unique node in $c \in \{1, \ldots, n-1\}$ which receives a non-zero flow from node 0.
- t'_{ic}: binary variable equal to 1 if node $i \in V \setminus S$ is the unique node in $c \in \{1, \ldots, n-1\}$ which receives a non-zero flow from node 0.
- f_{ij}: continuous flow variables from $i \in V \cup \{0\}$ to $j \in V$ defined for couples (i,j) such that $i = 0$ or $(i,j) \in E'$.
- ω_i: total weight of the component in $G[S]$ to which i belongs.
- ω'_i: total weight of the component in $G[V \setminus S]$ to which i belongs.
- ν_c: total weight of the component $c \in \{1, \ldots, n-1\}$.
- ν'_c: total weight of the component $c \in \{1, \ldots, n-1\}$.

Variables with a prime refer to quantities linked to the set $V \setminus S$. The approach advocated is inspired by a MILP formulation proposed for a Critical Node Problem in [17] which aims at deleting a subset of nodes in the graph in order to maximise the number of maximally connected components of the resulting subgraph. The rationale behind this strategy is to make sure that the structure of the connected components inside the

[1]Such a value for the number of connected components of $G[V \setminus S^*]$ can be obtained when G is a star graph.

partitioned graph is correctly taken into account by the MILP. In the modified graph, node 0 is the source of n units of flow while all the other nodes in V are sinks which absorb one unit of net flow each. Moreover, each node is assigned to a connected component inside $G[S]$ or $G[V \setminus S]$ and only one node per connected component can receive a non-zero amount of flow from node 0. Finally, the flow received (and partially absorbed) by such a node is dispatched to the other nodes in the component until it is completely absorbed. Since a flow can pass only between two nodes connected by an edge $\{i, j\}$ for which $y_{ij} = 1$ or $y'_{ij} = 1$, the flow conservation conditions guarantee that we cannot assign two nodes to the same connected component if they are not indeed part of a unique connected component inside $G[S]$ or $G[V \setminus S]$. We illustrate this process in Figure 2 on a small graph where the optimal safe set is made up of the two grey nodes 1 and 2 (with weight $w_1 = w_2 = 2$) while the nodes in $V \setminus S$ (with unit weights) are white. A total flow of 8 is created at node 0 and flows along the arcs displayed as dashed lines towards one node in the unique component of the safe set and the two nodes 3 and 6 belonging each to a component of $G[V \setminus S]$ (the other arcs from node 0 to the remaining nodes in V are omitted for reasons of clarity). The arcs where a non-zero flow circulates are displayed as directed arcs in order to represent the direction of the flow.

We stress that instead of using a multi-commodity flow model as in [17], we only use a single type of flow. Indeed, the model of [17] is a bi-level MILP which needs an integrality property for the inner optimisation problem in order to dualise it and obtain a single level reformulation (see Proposition 1 in [17]). Restricting to a single commodity in the framework of [17] would nullify this property, while it is perfectly admissible in our case. This certainly introduces a less tight formulation on general grounds. However, it greatly reduces the number of flow variables and accelerates notably the resolution of the linear relaxation of the MILP presented below. Moreover, our tests with a multi-commodity formulation tend to hint at the fact that such a formulation does not in practice improve the lower bound of the model detailed below.

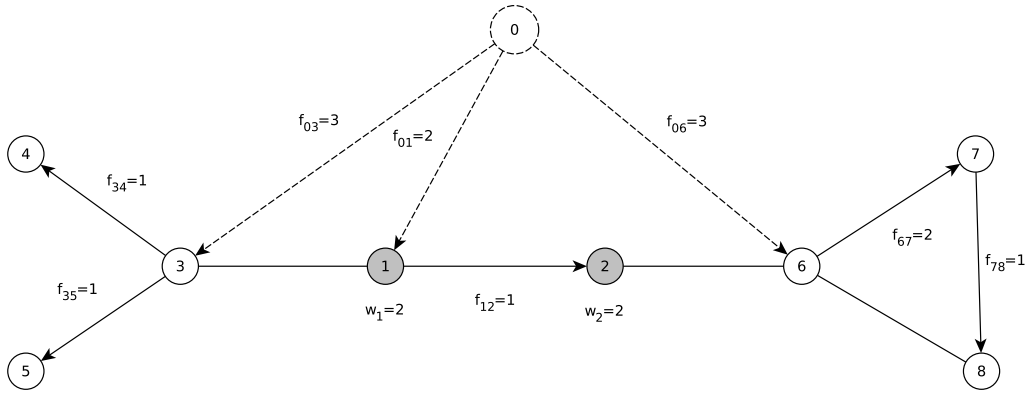

Figure 2: Example of a feasible flow configuration for a small instance of the WSSP. The nodes in the safe set are displayed in grey.

Armed with the above variables, we propose the following formulation of the WSSP, called $\mathcal{M}_{SS}$:

$$(\mathcal{M}_{SS}) \text{ minimise } \sum_{i \in V} w_i x_i \tag{1}$$

$$\text{subject to } y_{ij} \geq x_i + x_j - 1 \qquad \{i,j\} \in E, \tag{2}$$

$$y'_{ij} \geq 1 - x_i - x_j \qquad \{i,j\} \in E, \tag{3}$$

$$y_{ij} \leq x_i \qquad \{i,j\} \in E, \tag{4}$$

$$y_{ij} \leq x_j \qquad \{i,j\} \in E, \tag{5}$$

$$y'_{ij} \leq 1 - x_i \qquad \{i,j\} \in E, \tag{6}$$

$$y'_{ij} \leq 1 - x_j \qquad \{i,j\} \in E, \tag{7}$$

$$\sum_{j \in N_i} f_{ij} - \sum_{j \in N_i \cup \{0\}} f_{ji} = -1 \qquad i \in V, \tag{8}$$

$$\sum_{i \in V} f_{0i} = n \tag{9}$$

$$f_{0i} \leq (n-1) \sum_{c=1}^{n-1} (t_{ic} + t'_{ic}) \qquad i \in V, \tag{10}$$

$$f_{ij} \leq (n-1)(y_{ij} + y'_{ij}) \qquad (i,j) \in E', \tag{11}$$

$$\sum_{c=1}^{n-1} a_{ic} = x_i \qquad i \in V, \tag{12}$$

$$\sum_{c=1}^{n-1} a'_{ic} = 1 - x_i \qquad i \in V, \tag{13}$$

$$a_{ic} \geq a_{jc} + y_{ij} - 1 \qquad \{i,j\} \in E,\ c = 1, \ldots, n-1, \tag{14}$$

$$a_{jc} \geq a_{ic} + y_{ij} - 1 \qquad \{i,j\} \in E,\ c = 1, \ldots, n-1, \tag{15}$$

$$a'_{ic} \geq a'_{jc} + y'_{ij} - 1 \qquad \{i,j\} \in E,\ c = 1, \ldots, n-1, \tag{16}$$

$$a'_{jc} \geq a'_{ic} + y'_{ij} - 1 \qquad \{i,j\} \in E,\ c = 1, \ldots, n-1, \tag{17}$$

$$t_{ic} \leq a_{ic} \qquad i \in V,\ c = 1, \ldots, n-1, \tag{18}$$

$$t'_{ic} \leq a'_{ic} \qquad i \in V,\ c = 1, \ldots, n-1, \tag{19}$$

$$\sum_{i \in V} t_{ic} \leq 1 \qquad c = 1, \ldots, n-1, \tag{20}$$

$$\sum_{i \in V} t'_{ic} \leq 1 \qquad c = 1, \ldots, n-1, \tag{21}$$

$$\nu_c = \sum_{i \in V} w_i a_{ic} \qquad c = 1, \ldots, n-1, \tag{22}$$

$$\nu'_c = \sum_{i \in V} w_i a'_{ic} \qquad c = 1, \ldots, n-1, \tag{23}$$

$$\omega_i \geq \nu_c - W(1 - a_{ic}) \qquad i \in V,\ c = 1, \ldots, n-1, \tag{24}$$
$$\omega_i \leq \nu_c + W(1 - a_{ic}) \qquad i \in V,\ c = 1, \ldots, n-1, \tag{25}$$
$$\omega_i \leq W x_i \qquad i \in V, \tag{26}$$
$$\omega'_i \geq \nu'_c - W(1 - a'_{ic}) \qquad i \in V,\ c = 1, \ldots, n-1, \tag{27}$$
$$\omega'_i \leq \nu'_c + W(1 - a'_{ic}) \qquad i \in V,\ c = 1, \ldots, n-1, \tag{28}$$
$$\omega'_i \leq W(1 - x_i) \qquad i \in V, \tag{29}$$
$$\omega_i \geq \omega'_j - W y'_{ij} \qquad (i,j) \in E', \tag{30}$$
$$\sum_{i \in V} x_i \geq 1 \tag{31}$$
$$y_{ij}, y'_{ij} \in \{0,1\} : \{i,j\} \in E, \tag{32}$$
$$a_{ic}, a'_{ic} \in \{0,1\} : i \in V,\ c = 1, \ldots, n-1, \tag{33}$$
$$\omega_i, \omega'_i \geq 0 : i \in V, \tag{34}$$
$$\nu_c, \nu'_c \geq 0 : c = 1, \ldots, n-1, \tag{35}$$
$$f_{ij} \in [0, n-1] : i \in V \cup \{0\},\ j \in V, \tag{36}$$
$$x_i \in \{0,1\} : i \in V, \tag{37}$$
$$t_{ic}, t'_{ic} \in \{0,1\} : i \in V,\ c = 1, \ldots, n-1. \tag{38}$$

Should we want to extend the problem to the connected version, we must define the variables t_{ic}, a_{ic} and ν_c only for $c = 1$, or else add the following constraint:

$$\sum_{i \in V} \sum_{c=2}^{n-1} a_{ic} = 0. \tag{39}$$

Let us now explain briefly each constraint family used to model the WSSP. Constraints (2) to (7) guarantee the consistency of y variables with respect to the decision variables x. They are a linearisation of constraints $y_{ij} = x_i x_j$ and $y'_{ij} = (1 - x_i)(1 - x_j)$ for $\{i,j\} \in E$. Constraint (8) guarantees that a unit of flow coming from node 0 is absorbed by each node in the graph. Constraint (9) guarantees that exactly the right amount of flow is provided by fictitious node 0. Constraints (10) ensure that flow can only go from node 0 to another node in the graph if the associated t or t' variable is non-zero. Constraints (11) then make sure that the flow can only circulate on edges inside a given component of $G[S]$ or $G[V \setminus S]$. Constraints (12) and (13) assign a node $i \in V$ to only one component, be it in $G[S]$ or $G[V \setminus S]$. Constraints (14) to (17) then ensure that two nodes inside the same component are indeed assigned to the same component number c. Indeed, when $y_{ij} = 1$, the two inequalities (14) and (15) will transform into the equality $a_{ic} = a_{jc}$ for all values of c (the same is true for (16) and (17) when $y'_{ij} = 1$). We then define the consistency relation of the t variables with respect to variables a in constraints (18) and (19) and also that only one node inside component c of $G[S]$ or $G[V \setminus S]$ can have a non-zero t value. This node will represent the bridge through which some flow will enter component c and then be dispatched to all nodes of c, each of which needs to absorb one unit of net flow. The combination of all previous constraints impose that for each partition c of $G[S]$ (respectively $G[V \setminus S]$), the necessary quantity of flow will be dispatched to c through a single bridge node, and will then be absorbed by all the nodes in the component. Then, the link between t and a variables ensures that the nodes in this component will be assigned to the corresponding component number.

Constraints (22) and (23) are used to define the ν variables according to which node is assigned to each component label c and constraints (24) to (29) ensure that the ω variables are defined correctly with respect

to the ν variables. Recalling that ω_i is the total weight of the component that node $i \in S$ belongs to, constraints (30) are the Safe Set constraints which guarantee that $\omega_i \geq \omega'_j$ if $\{i,j\} \in E$ with $i \in S$ and $j \in V \setminus S$. Finally, since constraint (30) allows for the model to choose to assign every node to $V \setminus S$ (and therefore $S = \emptyset$), we force the model to have at least one node in S with Equation (31).

Proposition 1. *Model $\mathcal{M}_{SS}$ provides the optimal solution for the (Connected) WSSP.*

Proof. Consider first an optimal solution S^* to the WSSP. We can construct a feasible solution to model $\mathcal{M}_{SS}$ as follows. First define $x^*_i = 1$ for all nodes $i \in S^*$ and $x^*_i = 0$ for $i \in V \setminus S^*$. It follows that $y^*_{ij} = 1$ for $i \in S^*$ and $j \in S^*$ with $\{i,j\} \in E$, while $y^*_{ij} = 0$ if either $i \notin S^*$ or $j \notin S^*$ (and conversely $y'^*_{ij} = 1$ if $i \in V \setminus S^*$ and $j \in V \setminus S^*$ with $\{i,j\} \in E$ and 0 otherwise), so that constraints (2) to (7) are satisfied. Consider the sets $\mathcal{C}_S$ and $\mathcal{C}'_S$ defined respectively as the connected components of graphs $G[S^*]$ and $G[V \setminus S^*]$. For each component $C \in \mathcal{C}_S$, pick one node $i \in C$ and a yet unused index $c \in \{1, \ldots, n-1\}$ and set $a^*_{ic} = 1$, as well as $t^*_{ic} = 1$. For nodes j of the same component set $a^*_{jc} = 1$ for the same index c and $t^*_{jc} = 0$, then set all other variables a^*, a'^*, t^* and t'^* related to the nodes of C at value 0. The same thing can be done for the nodes of each component $C \in \mathcal{C}'_S$ by changing the roles of a and t variables with a' and t' variables. Given the values of the y^*_{ij} and y'^*_{ij} variables for arcs $\{i,j\}$ inside each connected component, the values assigned clearly respect the constraints (12) to (21). For each index $c \in \{1, \ldots, n-1\}$ attributed to a connected component $C \in \mathcal{C}_S$, we set variable ν^*_c to the total weight of the nodes of that component and for each node $i \in S$ we set ω^*_i to the total weight of the component that node i belongs to (and we perform the same actions for the components of $\mathcal{C}'_S$ and the variables ν'^*_c and ω'^*_i). Finally, we can set the flow variables as follows. For each node $i \in V$ and $c \in \{1, \ldots, n-1\}$ such that $t^{(\prime)*}_{ic} = 1$, we set f^*_{0i} to be equal to the number of nodes in the connected component of node i. After which, we increase f^*_{jk} by one unit over each edge $\{j,k\}$ on a path from node i to each node of its connected component. It is easy to verify that constraints (8) to (11) are satisfied by such assignments. We therefore have constructed a solution to model $\mathcal{M}_{SS}$ with an objective function value which is equal to $w(S^*)$ and therefore is not larger than the one of optimal solution S^* to the WSSP.

Consider now the optimal solution to model $\mathcal{M}_{SS}$, with binary optimal values for x^*_i for $i \in V$. We define the safe set $S^* = \{i \in V : x^*_i = 1\}$. By construction, $w(S^*)$ is the same as the optimal objective function value of $\mathcal{M}_{SS}$ so we have constructed a solution to the WSSP with an objective function value which is not larger than the one of the optimal solution to model $\mathcal{M}_{SS}$, which completes the proof. □

Proposition 2. *The y, y', a and a' variables can be relaxed as continuous and still achieve integrality when the x and t are integers.*

Proof. Suppose the variables $y^{(\prime)}$ and $a^{(\prime)}$ were defined as continuous variables bounded by 0 and 1. Consider a feasible solution to $\mathcal{M}_{SS}$ with integer x and $t^{(\prime)}$ variables. We have a partition of V into $S = \{i \in V : x_i = 1\}$ and its complement $V \setminus S = \{i \in V : x_i = 0\}$. Consequently, the integrality of y and y' is straightforward to check. Indeed, $y_{ij} = 0$ when either $x_i = 0$ or $x_j = 0$ because of constraints (4) and (5). In contrast, by (2), $y_{ij} = 1$ when $x_i = x_j = 1$ (while the same reasoning applies to the y').

We now prove the integrality of the a variables; the integrality of the a' variables follows by a similar argument. For two nodes $i \in S$ and $j \in S$ with $\{i,j\} \in E$, constraints (14) and (15) will ensure that $a_{ic} = a_{jc}$ for all $c = 1, \ldots, n-1$ and by propagation the same equalities will hold between all pairs of nodes inside the same connected component. By constraint (9) and flow conservation constraints (8), a total flow of n units originates from node 0 and each node has to absorb one unit, so that some non-zero flow $f_{0i} > 0$

is required for a subset of nodes $i \in V$. Therefore, by constraints (10) and (20), for each component index c with $\sum_{i \in V} a_{ic} > 0$ there will be one (and only one) node i with $t_{ic} = 1$ and $f_{0i} > 0$. For that node, we see from (18) that $a_{ic} = 1$ and $a_{id} = 0$ for all $d \neq c$, so that i is assigned to component label c. By the reasoning applied on the a variables earlier, we can extend this result to all the nodes inside the same connected component as i, which guarantees that the a variables are binary whenever the x and t variables are. This completes the proof of integrality for $y^{(\prime)}$ and $a^{(\prime)}$ variables. □

When implementing the model for numerical experiments, the y and y' variables are kept continuous but we noticed that the model is solved faster if the a and a' variables are kept binary. This unexpected behaviour might be due to the important limitations imposed by setting an a variable to 0 or 1, through a propagation effect of constraints (14) to (17). In such a case, branching early on a and a' might allow for a larger increase of the optimal objective value of the linear relaxation when $\mathcal{M}_{SS}$ is solved by a linear solver and speed up the overall convergence of the branching procedure. Moreover, we observe that CPLEX is able to eliminate some $a^{(\prime)}$ variables during its presolving step, which might also explain why the reduced model is then easier to solve. In the above MILP we have in total $\mathcal{O}(n^2)$ binary variables and $\mathcal{O}(m)$ continuous variables (or $\mathcal{O}(n^2)$ continuous variables if the a and a' variables are treated as continuous) and we have a maximum of $\mathcal{O}(mn)$ constraints, which overall makes the model relatively compact for sparse graphs.

3 MILP improvements

In this section we provide some improvements which will help solving the $\mathcal{M}_{SS}$ faster when using a numerical MILP solver.

3.1 Symmetry breaking

The model defined in Section 2 suffers from several symmetries which might hamper its performance when we try to solve it with a MILP solver. For example, any node inside a given component c can be used equivalently to dispatch the flow from node 0 to the rest of the component. We can break such symmetry with the following constraints:

$$\sum_{j>i} t_{jc} \leq t_{ic} + 1 - a_{ic}, \qquad i \in V, c = 1, \ldots, n, \tag{40a}$$

$$\sum_{j>i} t'_{jc} \leq t'_{ic} + 1 - a'_{ic}, \qquad i \in V, c = 1, \ldots, n. \tag{40b}$$

Such constraints ensure that the node with the smallest index will be the one chosen to receive a non-zero flow from node 0.

We may also observe that for a given solution, a component in the graph can be given a priori any label $c = 1, ..., n-1$. We can try to break (partially) that symmetry through the following constraints:

$$\nu_c \leq \nu_{c-1}, \qquad 2 \leq c \leq n-1, \tag{41a}$$

$$\nu'_c \leq \nu'_{c-1}, \qquad 2 \leq c \leq n-1. \tag{41b}$$

3.2 Reducing the number of variables

Given the large initial range for the c index of a and a', $\mathcal{M}_{SS}$ has a large number of binary variables, most of which are superfluous in many instances. Moreover, for the linear relaxation of $\mathcal{M}_{SS}$, the initial large number of c indices for a' allows for many small fractional values of these variables, suggesting that the nodes are partially "spread" inside many components and can satisfy easily the Safe Set constraint (30). Limiting the range of available c indices for the nodes in a generic safe set or its complement could help increase the lower bound of the linear relaxation. In the following, we introduce a model whose optimal solution allows us to find an upper bound on the c index for all the variables of $\mathcal{M}_{SS}$.

In order to do so, we consider the "worst case", for any given safe set S, in terms of the number of connected components of $G[V \setminus S]$. We prove below that an upper bound on the number of connected components in both $G[S]$ and $G[V \setminus S]$ can be obtained by finding the subset $\Sigma^* \subset V$ of smallest cardinality $|\Sigma^*| = k^*$ with at least one edge towards each node of $V \setminus \Sigma^*$, which is the definition of the *dominating set* of minimum cardinality (which is an NP-hard problem [14]). A classic MILP formulation for the Minimum Dominating Set Problem (see e.g., [18]) uses the following variables:

- $x_i = 1$ if node $i \in \Sigma$ and 0 otherwise.

We also define the set $\Gamma_i = N_i \cup \{i\}$ which contains node $i \in V$ and its neighbours. k^* is the optimal objective value of the following optimisation problem, called Δ_{SS}:

$$(\Delta_{SS}) \text{ minimise} \quad k \tag{42a}$$

$$\text{subject to} \sum_{i \in V} x_i = k \tag{42b}$$

$$\sum_{j \in \Gamma_i} x_j \geq 1 \qquad i \in V, \tag{42c}$$

$$k \in \mathbb{N}, \quad x_i \in \{0,1\} : i \in V. \tag{42d}$$

Proposition 3. *The optimal value k^* for the model (42) is such that $n - k^*$ defines an upper bound on the number of maximally connected components in $G[S]$ and $G[V \setminus S]$ respectively for any solution set S of the SSP.*

Proof. Consider the graph $G[V \setminus S]$ and its connected components. If one such component has more than one node, it is possible to move some of these nodes to the safe set and obtain more (in the worst case, not less) connected components in $G[V \setminus S]$. By induction, a solution with the largest possible number of components in $G[V \setminus S]$ is obtained when each such component has cardinality one. We obtain an upper bound on the number of connected components by ignoring the edges inside $G[V \setminus S]$, so that each node in $V \setminus S$ defines an independent connected component, and by relaxing the safe set constraints (30), supposing it is satisfied by any solution we select. However, since each node in $G[V \setminus S]$ has to define a possible independent component, it has to be connected by an edge to a node in the safe set S. Consequently, the problem is akin to finding a dominating set Σ^* of minimum cardinality. Moreover, since k^* represents the number of nodes attributed to the safe set and each node in $G[V \setminus \Sigma^*]$ is supposed to form a unique component, $n - k^*$ is the number of components in $G[V \setminus \Sigma^*]$. We can apply the same reasoning symmetrically to the components of $G[S]$, which completes the proof. □

Although the Minimum Dominating Set Problem was proved to be NP-hard, it is solved very quickly for the benchmark instances presented in Section 4 and Appendix A, which makes it an acceptable pre-processing

step for the SSP. Using the optimal solution k^* of Δ_{SS} allows us to define variables $a_{ic}^{(\prime)}$, $t_{ic}^{(\prime)}$ for each $i \in V$ and $\nu_c^{(\prime)}$ for the restricted set of indices $c = 1, \ldots, n - k^*$.

Alternative ways to obtain an upper bound on the number of components of the safe set or its complement can be devised. For example, the Critical Node Problem based on removing a subset of nodes of maximum cardinality in order to maximise the number of connected components in the induced subgraph, see e.g. [17, 20], can provide the maximum number of connected components in any induced subgraph if one removes the cardinality constraint on the set of deleted nodes. By extension, and since we are interested in the number of components and not their cardinality, we can extend the solution of such a modified Critical Node Problem by deleting nodes further until all remaining components have cardinality one, in which case we would obtain a vertex cover of the graph. Ergo, the solution to the Minimum Vertex Cover Problem would also provide a valid bound on the number of connected components of the safe set and its complement. Such problems, however, tend to be slower to solve than the Minimum Dominating Set Problem and therefore subtract more computational time to model $\mathcal{M}_{SS}$, since the total time budget of our experiments is used for the preprocessing step and the resolution of model $\mathcal{M}_{SS}$ together. Preliminary results on select instances, using a relaxed version of the Critical Node Problem, did not indicate a significant improvement with respect to using the Minimum Dominating Set Problem.

4 Numerical experiments

We now propose to test the model $\mathcal{M}_{SS}$ with the improvements of Sections 3.1 and 3.2 on a set of small instances in order to test its viability.

4.1 Instance generation and experimental conditions

In order to test our MILP, we generate a first set of benchmark instances, with $n \in \{20, 25, 30, 35, 40, 50\}$ with different possible values for the edge density $\delta_e = 2m/(n(n-1)) \in \{0.1, 0.2, 0.3, 0.4\}$. For each value of n and δ_e, we generate five instances randomly as follows. First we randomly add edges to create a tree, to ensure that the graph is connected, then we add edges randomly inside the graph until we reach the desired edge density. The graphs share similar properties with the random graphs of the Erdös-Renyi type, except that the constraint of building a connected graph can influence the graph characteristics for small sizes, e.g. the graphs with 10 nodes and density 0.1 have to be trees. The average characteristics of the graph subsets for each value of n and δ_e are displayed in Table 1. They are respectively the clustering coefficient C, the average length of all shortest paths $\bar{l}$, the diameter D, the average degree $\bar{d}$, the dispersion of the nodes degree σ_d and finally the ratio of the last two quantities which gives an idea whether the nodes have similar degree or whether there is a large degree dispersion in the graph. The measure of dispersion used the standard deviation. These same instances can be considered either as unweighted or with node weights randomly generated as integers between 1 and 10. Additional results on instances of the *small world* type are provided and discussed in Appendix A, while a comparison of $\mathcal{M}_{SS}$ with the branch-and-cut approach of [16] on a different set of benchmark instances is provided in Appendix B.

We add the symmetry-breaking constraints (40) to model $\mathcal{M}_{SS}$. The symmetry-breaking constraints (41) make the linear relaxation much slower and do not allow for any speedup when solving $\mathcal{M}_{SS}$, therefore we do not include them in our numerical experiments. The indices for the a and a' variables are reduced using the variable reduction technique introduced in Section 3.2 by solving Δ_{SS} as a preprocessing step, which is

n	δ_e	C	$\bar{l}$	D	$\bar{d}$	σ_d	$\sigma_d/\bar{d}$	n	δ_e	C	$\bar{l}$	D	$\bar{d}$	σ_d	$\sigma_d/\bar{d}$
20	0.1	0.00	7.00	19.00	1.90	1.09	0.58	20	0.2	0.15	2.24	4.40	3.80	2.48	0.65
25	0.1	0.08	4.53	11.00	2.40	1.49	0.62	25	0.2	0.14	2.10	4.00	4.80	2.98	0.62
30	0.1	0.05	3.25	6.80	2.87	1.88	0.66	30	0.2	0.19	2.06	3.80	5.80	3.71	0.64
35	0.1	0.06	3.07	6.60	3.37	2.24	0.66	35	0.2	0.18	1.99	3.60	6.80	4.25	0.62
40	0.1	0.10	2.79	5.40	3.90	2.54	0.65	40	0.2	0.20	1.98	3.60	7.80	4.89	0.63
50	0.1	0.08	2.59	4.80	4.88	3.22	0.66	50	0.2	0.19	1.91	3.00	9.80	6.10	0.62
20	0.3	0.27	1.81	3.00	5.70	3.58	0.63	20	0.4	0.39	1.62	3.00	7.60	4.69	0.62
25	0.3	0.29	1.77	3.00	7.20	4.53	0.63	25	0.4	0.39	1.60	2.60	9.60	5.73	0.60
30	0.3	0.29	1.74	3.00	8.67	5.28	0.61	30	0.4	0.39	1.60	2.60	11.60	6.95	0.60
35	0.3	0.30	1.73	3.00	10.17	6.23	0.61	35	0.4	0.39	1.60	2.40	13.60	8.00	0.59
40	0.3	0.30	1.72	3.00	11.70	7.15	0.61	40	0.4	0.40	1.60	2.40	15.60	9.28	0.60
50	0.3	0.30	1.71	3.00	14.68	8.94	0.61	50	0.4	0.40	1.60	2.20	19.60	11.65	0.59

Table 1: Average characteristics (explained in the text) for the sets of five instances, for each edge density δ_e and graph size n.

always very fast. For the connected version of the problem, we impose that variables a and t are defined only for index $c = 1$. We stress that in such a case, the variable reduction technique can still be applied to the complement of the safe set.

The code is implemented using the C++ library of CPLEX 12.9. Results are obtained on a computer with an Intel Core i7-6600U CPU at 2.60GHz (4 threads) and 32GB of memory. We set CPLEX's absolute gap to 0 and impose a time limit of 3600 seconds (minus the time needed to solve the variable reduction problem, which in practice is almost negligible). We also disable CPLEX's cuts, because they tend to slow down the solver and do not substantially improve the lower bound on the objective value.

4.2 Numerical results

We start this section with a numerical study of the effect of the MILP improvements of Section 3. The results are displayed in Table 2 for the instances of the first benchmark where each line represents an instance size and we limit ourselves to the smallest and highest density values $\delta_e = 0.1$ and $\delta_e = 0.4$ to assess the efficiency of the improvements both for our least and most dense instances. For each instance size we present the average time (in seconds) to solve the problem over the five corresponding instances, the average relative gap (in percent) normalised by the upper bound (i.e. comprised between 0 and 100%) and the number of instances (out of five) that were solved to optimality within the time limit. We include the instances which reached the time limit in the average solution time. The results were computed using the unweighted version of the SSP, respectively by solving $\mathcal{M}_{SS}$ alone, by solving $\mathcal{M}_{SS}$ with the symmetry constraints (40) and finally by solving $\mathcal{M}_{SS}$ with the variable reduction technique of model Δ_{SS}. From the table, for $\delta_e = 0.1$, it appears that each improvement method brings by itself a reduction in the average time needed to solve an instance of the SSP, however the symmetry-breaking technique manages to close more instances than the variable reduction preprocessing. The results when using both the symmetry breaking constraints and the variable reduction technique are instead displayed in the upper left part of Table 3: the added benefit of using both techniques simultaneously is a further increase in the number of solved instances. For denser instances with $\delta_e = 0.4$ however, the benefit of the improvements is less clear. More precisely, the addition

of the symmetry-breaking constraints seems to be beneficial but the variable reduction mechanism does not provide a clear help. As a consequence, the addition of both MILP improvements does not improve as clearly over the results of $\mathcal{M}_{SS}$ without any improvement, though the number of closed instances is larger. This is not necessarily surprising since model $\mathcal{M}_{SS}$ already shows better performances on denser instances without the improvements.

		$\mathcal{M}_{SS}$			$\mathcal{M}_{SS}$+(40)			$\mathcal{M}_{SS} + \Delta_{SS}$		
n	δ_e	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	0.1	453.6	0.00	5	273.6	0.00	5	198.2	0.00	5
25	0.1	3347.0	40.69	1	2291.8	24.18	3	2598.8	25.78	2
30	0.1	3117.2	30.11	2	1932.2	11.60	4	3037.2	33.98	2
35	0.1	3600.0	74.57	0	3600.0	48.55	0	3600.0	52.14	0
40	0.1	3600.0	80.29	0	3600.0	72.76	0	3600.0	76.40	0
50	0.1	3600.0	92.94	0	3600.0	91.81	0	3600.0	90.84	0
20	0.4	15.4	0.00	5	18.0	0.00	5	14.0	0.00	5
25	0.4	103.6	0.00	5	95.6	0.00	5	65.6	0.00	5
30	0.4	327.2	0.00	5	334.2	0.00	5	237.6	0.00	5
35	0.4	2782.0	10.16	2	1632.4	0.00	5	2546.8	6.72	2
40	0.4	3322.4	8.33	3	2672.0	0.00	5	3380.8	16.12	3
50	0.4	3600.0	95.97	0	3600.0	95.15	0	3600.0	93.77	0

Table 2: Average results for the unweighted SSP over sets of five instances for edge density $\delta_e = 0.1$ or 0.4 and graph size n. We display results without improvements ($\mathcal{M}_{SS}$), with the help of symmetry-breaking constraints ($\mathcal{M}_{SS}$+(40)) and of the variable reduction technique ($\mathcal{M}_{SS} + \Delta_{SS}$). For each graph size, we display the average running time of the model, the average relative gap and the number of instances out of five that are solved within the one hour time limit. The time is displayed in seconds.

We display the results for the unweighted and weighted version of the SSP, with the help of both MILP improvements, in Table 3. The quantities reported in the table are the same as in Table 2. The model is generally able to find the optimal solution for instances of up to 30 nodes. Additionally, instances with a higher density of edges seem to be easier to solve as the average time is generally smaller for each instance size and a larger number of instances are solved within the time limit. This is in sharp contrast to some other types of node partitioning problems, see e.g. [4]. Therefore, it seems to be a particular feature of our model to be fit for graphs with a somewhat large density, which are generally found in social networks applications (see e.g. some examples of real social networks intances and their characteristics in [3]). We observe that it is hard to conclude from our results whether it is harder to solve the weighted version of the problem, although it generally provides a larger average relative gap for larger instances, which is not a surprise in the light of complexity results over trees [7, 8]. Though our model can hope to solve instances of up to 40 nodes depending on the graph density, it is clear that solving instances with 50 nodes is out of reach within a limited computational time of one hour.

The results for the Connected Safe Set are displayed in Table 4 and generally confirm our previous findings. The dependance of the solve time on the edge density is harder to establish though, even if we still solve more instances with $\delta_e = 0.4$ than in other cases. It is generally faster to solve the connected version of the problem, which is probably due to the reduction of a and t variables in the model. We can solve more instances than the unconnected case within the one hour time limit. Nevertheless no instance of 50 nodes can be solved either for the C(W)SSP.

		Unweighted			Weighted		
n	δ_e	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	0.1	94.2	0.00	5	72.2	0.00	5
25	0.1	532.2	0.00	5	746.4	0.00	5
30	0.1	1743.8	4.44	4	2554.6	16.65	3
35	0.1	3538.0	33.84	1	3600.0	39.53	0
40	0.1	3600.0	67.73	0	3600.0	74.89	0
50	0.1	3600.0	87.97	0	3600.0	94.55	0
20	0.2	15.4	0.00	5	28.0	0.00	5
25	0.2	234.4	0.00	5	140.2	0.00	5
30	0.2	856.8	0.00	5	1472.0	7.52	4
35	0.2	3437.0	18.01	2	3522.6	34.49	1
40	0.2	3600.0	71.13	0	3600.0	72.21	0
50	0.2	3600.0	93.11	0	3600.0	93.72	0
20	0.3	14.4	0.00	5	11.8	0.00	5
25	0.3	77.6	0.00	5	52.0	0.00	5
30	0.3	362.6	0.00	5	562.2	0.00	5
35	0.3	2798.2	10.64	3	1956.8	4.31	4
40	0.3	3600.0	38.39	0	3384.0	43.19	1
50	0.3	3600.0	91.05	0	3600.0	96.36	0
20	0.4	14.2	0.00	5	12.8	0.00	5
25	0.4	98.8	0.00	5	66.4	0.00	5
30	0.4	267.8	0.00	5	375.0	0.00	5
35	0.4	1709.4	2.40	4	905.4	0.00	5
40	0.4	2678.4	11.58	4	2788.0	16.05	3
50	0.4	3600.0	95.27	0	3600.0	99.41	0

Table 3: Average results of model $\mathcal{M}_{SS}$ with symmetry breaking constraints (40) and the variable reduction preprocessing for the SSP, over sets of five instances for each edge density δ_e and graph size n. For both unweighted and weighted instances, we display the average running time of the model, the average relative gap and the number of instances out of five that are solved within the one hour time limit. The time is displayed in seconds.

5 Conclusion

We have provided a compact MILP for the (Connected) Weighted Safe Set problem, a somewhat new partitioning problem in the literature with interesting applications in network majority control. The model is relatively compact for sparse graphs. We also provided symmetry-breaking constraints and a variable reduction technique which greatly help the convergence of a numerical solver on small instances. Our results show that instances with up to 35 nodes can be solved within one hour of computational time (depending on the density), which makes it faster than the branch-and-cut algorithm from [16].

A first possible research direction to tackle the SSP through Mathematical Programming would be to study

		Unweighted			Weighted		
n	δ_e	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	0.1	4.4	0.00	5	4.6	0.00	5
25	0.1	41.4	0.00	5	55.0	0.00	5
30	0.1	258.4	0.00	5	284.4	0.00	5
35	0.1	2423.0	29.30	2	1614.2	11.05	4
40	0.1	3492.8	38.60	1	3529.6	41.95	1
50	0.1	3600.0	66.30	0	3600.0	74.76	0
20	0.2	7.8	0.00	5	6.8	0.00	5
25	0.2	47.2	0.00	5	35.4	0.00	5
30	0.2	232.2	0.00	5	193.8	0.00	5
35	0.2	3206.2	17.22	2	1687.4	5.62	4
40	0.2	3600.0	42.36	0	3600.0	41.44	0
50	0.2	3600.0	67.66	0	3600.0	76.88	0
20	0.3	7.2	0.00	5	6.4	0.00	5
25	0.3	48.6	0.00	5	51.2	0.00	5
30	0.3	256.4	0.00	5	217.6	0.00	5
35	0.3	2265.2	0.00	5	1300.6	0.00	5
40	0.3	3600.0	45.32	0	3600.0	45.44	0
50	0.3	3600.0	92.72	0	3600.0	83.52	0
20	0.4	9.6	0.00	5	9.8	0.00	5
25	0.4	71.8	0.00	5	74.8	0.00	5
30	0.4	236.0	0.00	5	302.4	0.00	5
35	0.4	2485.0	2.57	4	1002.4	0.00	5
40	0.4	3297.0	18.94	2	2915.0	0.00	5
50	0.4	3600.0	92.60	0	3600.0	95.21	0

Table 4: Results of model $\mathcal{M}_{SS}$ with symmetry breaking constraints (40) and the variable reduction preprocessing for the Connected SSP, on both unweighted and weighted instances. The quantities reported are the same as in Table 3.

alternative improvements to our model, such as tightening further the number of variables or reducing further the symmetries. Given the number of continuous variables introduced to define correctly the connected components of a given solution, it would be interesting to pursue the exploration of our model through Benders decomposition and check whether it can help increasing the size of instances which can be solved. Since the continuous variables are easy to determine once the integer variables are fixed, it could be possible to solve the dual of the slave problem analytically to speed up the convergence of the Benders approach. Since several inequalities in $\mathcal{M}_{SS}$ involve big-M coefficients, e.g. equations (24) to (30), alternative decomposition methods tailored for such cases, as in [10], are also interesting future research directions. Another type of decomposition known to be successful for some partitioning problem is column generation. An approach along the lines of [11] could provide interesting results on instances larger than those tackled in this work. Given the difficulty of solving even small instances of the Safe Set problem, designing tailored efficient heuristic approaches would be a very welcome avenue of research. For example, carefully designed

metaheuristics have had great success for solving difficult node partitioning problems in the literature where exact approaches have failed to obtain good quality solutions [2, 3, 4, 23].

Acknowledgements

I would like to thank Rosario Scatamacchia for sharing C++ code useful for generating the benchmark instances and for fruitful discussions, as well as the constructive and detailed comments of three anonymous referees which greatly helped enhance the manuscript.

References

[1] R. Águeda, N. Cohen, S. Fujita, S. Legay, Y. Manoussakis, Y. Matsui, L. Montero, R. Naserasr, H. Ono, Y. Otachi, T. Sakuma, Z. Tuza, and R. Xu. Safe sets in graphs: Graph classes and structural parameters. *Journal of Combinatorial Optimization*, 36(4):1221–1242, 2018.

[2] R. Aringhieri, A. Grosso, and P. Hosteins. A Genetic Algorithm for a class of Critical Node Problems. In *The 7th International Network Optimization Conference (INOC'15)*, volume 52 of *Electronic Notes in Discrete Mathematics*, pages 359–366, 2016.

[3] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. A general Evolutionary Framework for different classes of Critical Node Problems. *Engineering Applications of Artificial Intelligence*, 55:128–145, 2016.

[4] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. Local Search Metaheuristics for the Critical Node Problem. *Networks*, 67(3):209–221, 2016.

[5] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[6] E. Balas and C.C. de Souza. The vertex separator problem: a polyhedral investigation. *Mathematical Programming Series A*, 103(6):583–608, 2005.

[7] R.B. Bapat, S. Fujita, S. Legay, Y. Manoussakis, Y. Matsui, T. Sakuma, and Z. Tuza. Network majority on tree topological network. *Electronic Notes in Discrete Mathematics*, 54:79–84, 2016.

[8] R.B. Bapat, S. Fujita, S. Legay, Y. Manoussakis, Y. Matsui, T. Sakuma, and Z. Tuza. Safe sets, network majority on weighted trees. *Networks*, 71(1):81–92, 2018.

[9] W. Ben-Ameur, M.-A. Mohamed-Sidi, and J. Neto. The k-separator problem: polyhedra, complexity and approximation results. *Journal of Combinatorial Optimization*, 29(1):276–307, 2015.

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

[11] D. Cornaz, F. Furini, M. Lacroix, E. Malaguti, A. R. Mahjoub, and S. Martin. The vertex k-cut problem. *Discrete Optimization*, 31:8–28, 2019.

[12] T.N. Dinh, Y. Shen, D.T. Nguyen, and M.T. Thai. On the approximability of positive influence dominating set in social networks. *Journal of Combinatorial Optimization*, 27(3):487–503, 2014.

[13] S. Fujita, G. MacGillivray, and T. Sakuma. Safe set problem on graphs. *Discrete Applied Mathematics*, 215:106–111, 2016.

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

[15] M. Lalou, M.A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018.

[16] A.F.U. Macambira, L. Simonetti, H. Barbalho, P.H. Gonzalez, and N. Maculan. A new formulation for the safe set problem on graphs. *Computers & Operations Research*, 111:346–356, 2019.

[17] S. Shen, J.C. Smith, and R. Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9(3):172–188, 2012.

[18] L. Simonetti, A. Salles da Cunha, and A. Lucena. The minimum connected dominating set problem: Formulation, valid inequalities and a branch-and-cut algorithm. In J. Pahl, T. Reiners, and S. Voß, editors, *Network Optimization*, pages 162–169, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[19] M. Ventresca. Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Computers & Operations Research*, 39(11):2763–2775, 2012.

[20] A. Veremyev, O.A. Prokopyev, and E.L. Pasiliao. An integer programming framework for critical elements detection in graphs. *Journal of Combinatorial Optimization*, 28:233–273, 2014.

[21] F. Wang, E. Camacho, and K. Xu. Positive influence dominating set in online social networks. In D.-Z. Du, X. Hu, and P. M. Pardalos, editors, *Combinatorial Optimization and Applications*, pages 313–321, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[22] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:400–442, 1998.

[23] Y. Zhou, J.-K. Hao, and F. Glover. Memetic search for identifying critical nodes in sparse graphs. *IEEE Transactions on Cybernetics*, 49(10):3699–3712, 2019.

Appendix A Results over small world networks

In this appendix we present detailed results over small networks generated through the mechanism described in [22]. These networks typically have a small diameter and represent social networks fairly well. They are known to be notoriously difficult to solve for some partitioning problems, such as [4, 23]. We generated instances with a number of nodes in the set $n \in \{20, 25, 30, 35, 40\}$ as in Section 4. The parameters needed to generate small world graphs are the average number of neighbours k and a rewiring probability of edges which we choose as $p = 0.05$, similar to [19]. We vary parameter $k \in \{4, 6, 8, 10\}$ to obtain instances with different characteristics and generate five instances for each value of n and k. The different average coefficients over those sets of five instances are displayed in Table 5 in a manner similar to Table 1, except for the additional column $\bar{\delta}_e$ which represents the average density of the instances. We can clearly see that the clustering coefficient is usually much larger for small world graphs while the ratio of the dispersion (standard deviation) in the nodes degree over the average degree (last column) is much smaller, denoting the fact that two different nodes tend to have similar degree in a small world graph.

n	k	$\bar{\delta}_e$	C	$\bar{l}$	D	$\bar{d}$	σ_d	$\sigma_d/\bar{d}$	n	k	$\bar{\delta}_e$	C	$\bar{l}$	D	$\bar{d}$	σ_d	$\sigma_d/\bar{d}$
20	4	0.21	0.46	2.72	5.00	4.00	0.37	0.09	20	6	0.32	0.53	1.98	3.80	6.00	0.48	0.08
25	4	0.17	0.45	3.15	6.00	4.00	0.35	0.09	25	6	0.25	0.55	2.33	4.20	6.00	0.52	0.09
30	4	0.14	0.45	3.54	7.60	4.00	0.39	0.10	30	6	0.21	0.54	2.63	5.00	6.00	0.44	0.07
35	4	0.12	0.44	3.77	7.80	4.00	0.42	0.11	35	6	0.18	0.54	2.85	5.80	6.00	0.52	0.09
40	4	0.10	0.44	4.25	8.60	4.00	0.38	0.09	40	6	0.15	0.51	2.82	5.40	6.00	0.59	0.10
20	8	0.42	0.60	1.68	3.00	8.00	0.56	0.07	20	10	0.53	0.62	1.47	2.00	10.00	0.66	0.07
25	8	0.33	0.59	1.91	3.20	8.00	0.54	0.07	25	10	0.42	0.60	1.67	3.00	10.00	0.64	0.06
30	8	0.28	0.57	2.11	4.00	8.00	0.53	0.07	30	10	0.34	0.59	1.84	3.00	10.00	0.70	0.07
35	8	0.24	0.56	2.27	4.40	8.00	0.61	0.08	35	10	0.29	0.60	2.01	3.60	10.00	0.62	0.06
40	8	0.21	0.57	2.41	4.00	8.00	0.60	0.08	40	10	0.26	0.59	2.13	4.00	10.00	0.66	0.07

Table 5: Average characteristics for the sets of five instances, for the small world graphs. The columns represent the same quantities as in Table 1 except for the additional column $\bar{\delta}_e$ which represents the average edge density over the five instances.

As for the results in Section 4, we first display the effect of the MILP improvements for small world graphs in Table 6, for unweighted instances with parameter $k = 4$ and $k = 10$. The results are similar to those of Section 4, in the sense that instances which are less dense tend to profit from the MILP improvements while denser instances are not solved faster on average, while there even are instance sizes where the improvements slightly worsen the performance of the model. This underlines the dependency of the performance of said improvements on the graph topology.

		$\mathcal{M}_{SS}$			$\mathcal{M}_{SS}$+(40)			$\mathcal{M}_{SS} + \Delta_{SS}$		
n	k	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	4	239.2	0.00	5	140.2	0.00	5	103.6	0.00	5
25	4	2479.6	18.04	3	2153.2	19.24	3	1936.2	0.00	5
30	4	3600.0	40.96	0	3600.0	32.73	0	3600.0	44.01	0
35	4	3600.0	79.16	0	3600.0	62.24	0	3600.0	79.46	0
40	4	3600.0	83.82	0	3600.0	79.15	0	3600.0	71.42	0
20	10	27.4	0.00	5	24.2	0.00	5	22.2	0.00	5
25	10	138.2	0.00	5	110.4	0.00	5	88.2	0.00	5
30	10	461.2	0.00	5	838.0	0.00	5	373.4	0.00	5
35	10	2763.8	20.99	2	3264.2	12.03	2	2402.4	11.34	3
40	10	3600.0	57.59	0	3600.0	68.88	0	3600.0	55.99	0

Table 6: Average results for the unweighted SSP over sets of five instances of small world graphs for parameter $k = 4$ and $k = 10$ and number of nodes n. We display results without improvements ($\mathcal{M}_{SS}$), with the help of symmetry-breaking constraints ($\mathcal{M}_{SS}$+(40)) and with the help of the variable reduction technique ($\mathcal{M}_{SS} + \Delta_{SS}$). For each graph size, we display the average running time of the model, the average relative gap and the number of instances out of five that are solved within the one hour time limit. The time is displayed in seconds.

We display in Tables 7 and 8 the results of model $\mathcal{M}_{SS}$ with symmetry breaking constraints (40) and the variable reduction preprocessing on small world graphs, similar to Section 4. The SSP is still very hard to solve for such instances, even for very small graphs. As for random graphs, we can see that instances with a

larger k parameter (and hence with a larger edge density) are usually easier to solve. Contrary to Section 4, however, no instance with 40 nodes is ever solved to optimality since the density for such instances do not even reach density 0.3 or 0.4, even with $k = 10$.

		Unweighted			Weighted		
n	k	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	4	105.4	0.00	5	82.4	0.00	5
25	4	1430.8	3.42	4	727.2	0.00	5
30	4	2860.8	32.78	2	2975.0	20.23	2
35	4	3600.0	53.20	0	3600.0	65.42	0
40	4	3600.0	80.38	0	3600.0	81.06	0
20	6	22.2	0.00	5	19.0	0.00	5
25	6	379.6	0.00	5	429.0	0.00	5
30	6	2420.2	15.61	3	2398.0	2.94	4
35	6	3600.0	52.28	0	3600.0	53.56	0
40	6	3600.0	61.21	0	3600.0	80.44	0
20	8	20.2	0.00	5	17.4	0.00	5
25	8	94.8	0.00	5	123.2	0.00	5
30	8	1016.2	0.00	5	2087.8	0.00	5
35	8	3600.0	39.95	0	3422.6	41.44	1
40	8	3600.0	73.21	0	3600.0	68.62	0
20	10	25.8	0.00	5	19.8	0.00	5
25	10	113.2	0.00	5	116.8	0.00	5
30	10	525.8	0.00	5	442.0	0.00	5
35	10	3540.8	21.50	1	3461.2	23.33	2
40	10	3600.0	74.27	0	3600.0	68.01	0

Table 7: Average results of model $\mathcal{M}_{SS}$ with symmetry breaking constraints (40) and the variable reduction preprocessing for the SSP, over sets of five instances for small world graphs, for each average number of neighbours k and graph size n. For both unweighted and weighted instances, we display the average running time of the model, the average relative gap and the number of instances out of five that are solved within the one hour time limit. The time is displayed in seconds.

Appendix B Comparison with the model of [16]

We compare in the following results of our model $\mathcal{M}_{SS}$ with those of the MILP approach in [16] on a set of benchmark instances they introduced. As discussed in Section 1, the algorithm advocated in [16] is a branch-and-cut algorithm based on a model with an exponential number of constraints, implemented as lazy constraints. The results of both models are reported in Table 9 for unweighted instances of SSP and in Table 10 for weighted instances. The instances range from 10 to 30 nodes with densities 0.3, 0.5 and 0.7. We report the upper bound (UB) and running time of both models. For the model of [16], when the running time is 7200 seconds, the algorithm reached its time limit and the instance is not solved at optimality,

		Unweighted			Weighted		
n	k	time (s)	gap (%)	# solved	time (s)	gap (%)	# solved
20	4	44.4	0.00	5	29.4	0.00	5
25	4	457.6	0.00	5	410.2	0.00	5
30	4	952.2	0.00	5	1060.0	0.00	5
35	4	3526.6	21.70	2	3552.0	38.68	1
40	4	3600.0	47.06	0	3600.0	39.16	0
20	6	11.8	0.00	5	9.2	0.00	5
25	6	72.8	0.00	5	68.2	0.00	5
30	6	1785.8	6.00	4	1397.0	10.49	4
35	6	3600.0	40.59	0	3180.6	15.97	3
40	6	3600.0	49.11	0	3600.0	53.85	0
20	8	10.2	0.00	5	10.0	0.00	5
25	8	60.6	0.00	5	61.4	0.00	5
30	8	283.2	0.00	5	301.2	0.00	5
35	8	2121.6	13.57	3	1953.8	9.29	3
40	8	3600.0	46.95	0	3600.0	39.27	0
20	10	16.8	0.00	5	12.0	0.00	5
25	10	76.4	0.00	5	74.8	0.00	5
30	10	237.6	0.00	5	203.4	0.00	5
35	10	787.0	0.00	5	2102.2	6.65	3
40	10	3600.0	39.81	0	2922.8	12.01	3

Table 8: Results of model $\mathcal{M}_{SS}$ with symmetry breaking constraints (40) and the variable reduction preprocessing for the Connected SSP on small world graphs, both unweighted and weighted. The quantities reported are the same as in Table 7.

however the LB was not reported.

The results for the model of [16] are those reported in the tables of [16]. Even though those results were obtained on a different computer and using a somewhat different framework (e.g., we make use of the heuristics of CPLEX, while [16] does not), we believe the important quantitative differences in the running times of both models is enough to assert the relative computational efficiency of our approach. Indeed, in most instances with 20 nodes or more, model $\mathcal{M}_{SS}$ closes the instances at least ten times faster. On the smallest instances, and contrary to the numerical analysis in Section 4 and Appendix A, $\mathcal{M}_{SS}$ does not always solve denser instances faster for a given graph size. The same pattern, however, can be seen again on the largest unweighted instances. The results confirm that model $\mathcal{M}_{SS}$ (with MILP improvements) is able to solve instances of up to 30 nodes without problem.

n	$\delta_e = 0.3$				$\delta_e = 0.5$				$\delta_e = 0.7$			
	$\mathcal{M}_{SS}$		MILP of [16]		$\mathcal{M}_{SS}$		MILP of [16]		$\mathcal{M}_{SS}$		MILP of [16]	
	UB	time	UB	time	UB	time	UB	time	UB	time	UB	time
10	4	1	4	1	5	0	5	1	5	1	5	1
11	4	0	4	1	4	1	4	4	5	0	5	1
12	4	1	4	2	6	1	6	3	6	1	6	4
13	5	1	5	2	6	1	6	4	6	1	6	4
14	5	2	5	8	6	2	6	5	7	2	7	7
15	6	2	6	10	7	3	7	15	7	4	7	15
16	6	2	6	17	7	4	7	25	8	6	8	33
17	6	5	6	24	8	6	8	48	8	4	8	49
18	6	7	6	32	8	9	8	39	9	6	9	80
19	7	10	7	55	9	10	9	84	9	12	9	101
20	8	11	8	232	9	15	9	223	10	22	10	194
21	8	16	8	298	10	27	10	409	10	37	10	190
22	8	18	8	379	10	15	10	715	11	27	11	750
23	9	34	9	614	11	41	11	467	11	94	11	546
24	10	38	10	381	11	58	11	658	12	110	12	2434
25	11	59	11	1091	12	44	12	908	12	123	12	1768
26	11	78	11	1561	13	97	13	3307	13	148	13	2389
27	11	150	11	1854	13	134	13	2462	13	168	13	3899
28	12	284	12	4557	13	248	13	3891	14	194	14	2709
29	12	182	12	4277	14	130	14	4089	15	178	15	4680
30	13	472	13	4963	15	359	15	7200	15	332	15	7200

Table 9: UB and running time for the benchmark instances of [16] for unweighted instances with densities 0.3, 0.5 and 0.7, for model $\mathcal{M}_{SS}$ with improvements and the branch-and-cut algorithm of [16].

n	$\delta_e = 0.3$				$\delta_e = 0.5$				$\delta_e = 0.7$			
	$\mathcal{M}_{SS}$		MILP of [16]		$\mathcal{M}_{SS}$		MILP of [16]		$\mathcal{M}_{SS}$		MILP of [16]	
	UB	time	UB	time	UB	time	UB	time	UB	time	UB	time
10	249	0	249	1	254	0	254	2	312	1	312	1
11	214	1	214	1	214	0	214	1	260	1	260	3
12	197	1	197	3	251	0	251	4	300	1	300	4
13	116	1	116	2	149	1	149	4	173	1	173	5
14	235	2	235	7	299	2	299	8	320	3	320	14
15	251	2	251	8	301	3	301	21	348	4	348	27
16	216	4	216	31	279	4	279	27	323	6	323	45
17	257	4	257	25	336	4	336	45	377	4	377	59
18	208	4	208	35	325	7	325	43	378	10	378	96
19	281	13	281	72	378	10	378	69	420	14	420	158
20	433	11	433	131	489	13	489	126	533	19	533	318
21	374	11	374	222	545	20	545	326	606	33	606	304
22	421	22	421	864	524	17	524	515	571	51	571	1423
23	297	19	297	244	410	26	410	428	410	44	410	661
24	427	69	427	455	536	59	536	810	624	76	624	2297
25	425	61	425	940	539	49	539	2209	583	50	583	3066
26	510	203	510	2893	638	80	638	3042	681	113	681	3167
27	587	71	587	2648	725	83	725	4399	756	262	756	2626
28	444	221	444	3081	572	282	572	5962	636	251	636	6762
29	595	138	595	5559	735	211	735	5575	775	254	775	7200
30	562	219	562	3907	710	267	719	7200	759	305	759	7200

Table 10: UB and running time for the benchmark instances of [16] for weighted instances with densities 0.3, 0.5 and 0.7, for model $\mathcal{M}_{SS}$ with improvements and the branch-and-cut algorithm of [16].