Discrete Optimization 30 (2018) 144–167

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

An integer linear programming formulation for removing nodes in a network to minimize the spread of influenza virus infections

Hadi Charkhgard*, Vignesh Subramanian, Walter Silva, Tapas K. Das

Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL, 33620, USA

H I G H L I G H T S

- We study the spread of influenza virus infections on networks of people.
- We develop a novel integer programming formulation to minimize the influenza spread.
- We develop several enhancement techniques to improve the proposed formulation.

A R T I C L E   I N F O

Article history:
Received 5 February 2018
Received in revised form 16 June 2018
Accepted 18 June 2018
Available online 2 July 2018

Keywords:
Influenza
Node removal
Mixed integer linear programming

A B S T R A C T

We study the problem of minimizing the spread of influenza virus infections in (dynamic) networks of people by isolating sick nodes (or vaccinating susceptible nodes) over time. This can be viewed as removing nodes with certain characteristics from networks of people over time. We present a novel integer linear programming formulation for this problem that incorporates several practical aspects of the spread of influenza virus infections. A comprehensive computational study demonstrates the efficacy of the proposed formulation and several enhancement techniques.

1. Introduction

Influenza outbreaks raise serious concerns for public health decision makers due to their impact on health and economy. Each year, a significant number of people die from periodic influenza outbreaks and businesses incur financial losses due to the absence (or poor performance) of the infected employees. Influenza outbreaks often turn into *epidemics*, i.e. the spread is not worldwide but can infect hundreds of thousands of people. An outbreak becomes even more serious when it turns into a *pandemic*, i.e. the spread is worldwide and it can infect millions of people [1]. For example, Centers for Disease Controls and Prevention reports that an H1N1 influenza outbreak infected approximately 60.8 million people and caused 12,469 deaths just in the

* Corresponding author.
E-mail address: hcharkhgard@usf.edu (H. Charkhgard).

https://doi.org/10.1016/j.disopt.2018.06.005
1572-5286/© 2018 Elsevier B.V. All rights reserved.

United States from April 12, 2009 to April 10, 2010 [2]. So, not surprisingly, public health decision makers are constantly in search for better solutions to further reduce/minimize the spread of influenza outbreaks and their impact on the affected population.

The spread of an influenza virus infection depends highly on the connectivity of the network of people, where nodes represent individuals and links represent their connections/interactions. So, minimizing the spread of an outbreak can be attained by disconnecting the healthy and susceptible nodes of the network from the sick nodes. This can be done by removing some of the nodes or links of the network. This approach has been a common practice during the past influenza epidemics and pandemics. It is worth mentioning that the option of removing links from a network of people is not very attractive since removing the connection between two individuals is not usually easy. However, in many other real life networks, removing links makes more sense than removing nodes. For example, if nodes represent airports, and links represent flights between them then shutting down an entire airport may not be possible. However, to reduce the spread of an infection, some flights may be canceled from/to origins/destinations where the spread of the infection is highly prevalent [3]. A recent study conducted by Nandi and Medal [3] presents new integer linear programs and heuristic solution approaches in the context of node removal. Other related papers on this topic are Enns et al. [4], Kimura et al. [5], Marcelino and Kaiser [6,7], and Nandi et al. [8].

The focus of this study is on networks of people and hence a node removal strategy is of interest. Note that removing a node does not imply physically destroying a node. It only means that we assure that there is no way that the infection to be passed from that node to other nodes anymore. By this definition, two common approaches for removing a node in a network of people are vaccination (if the node is healthy and susceptible) and isolation (if the node is sick). Our assumption, in this study, is that both vaccination and isolation are 100% effective. In other words, if we vaccinate a healthy and susceptible node then it cannot become infected and cannot make other nodes infected anymore. Also, if we isolate a node then it cannot make other nodes infected anymore. It is worth mentioning that removing nodes to minimize the connectivity in a network has been studied for a long time in the fields of graph theory and/or network optimization [9–15]. However, the only study, to our knowledge, that explores node removal in the context of the spread of infections (in a broad sense and not influenza virus infections) is conducted by He et al. [16]. In that study, the authors develop a novel approach for deleting both nodes and links at the same time. They also propose new exact and approximate polynomial time algorithms for special cases of the problem.

In light of the above, we believe that, our study is the first attempt to apply mathematical optimization techniques for node removal in the context of influenza virus infections. The main contributions of our paper are as follows:

- We present an integer linear programming formulation to minimize the spread of an influenza virus infection by removing a subset of nodes over time in a (dynamic) network of people. It is also worth mentioning that the idea of using integer programming to control the spread of an infection is not new (see for instance [17]). However, the methods by which influenza virus infections spread in practice are quite complex. For example, it is quite possible that a susceptible individual who comes in contact with one infected does not become infected. Also, there is a transition time between the moment that an influenza virus enters to the body of a susceptible and the moment that the person becomes sick. Consequently, modeling the spread of an influenza virus infection is quite challenging. In fact, in this study, we attempt to develop a model that incorporates several practical aspects of the spread of influenza virus infections. We are not aware of any network model with link removal and/or node removal that considers the level of detail that we consider in this study for the virus transmission process.
- We develop several potential enhancement techniques including variable fixing, moderating big-M coefficients, and symmetry breaking for the proposed formulation. We conduct a computational study

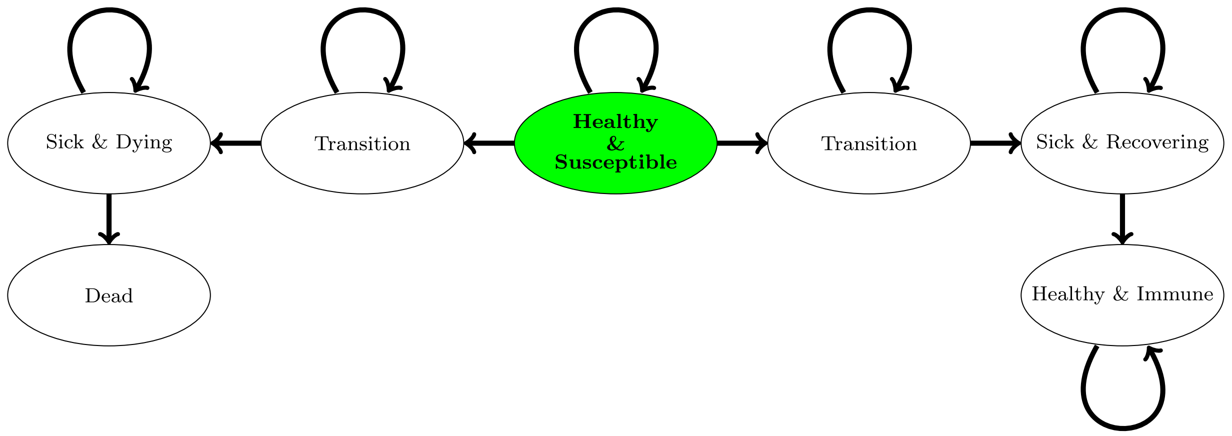


Fig. 1. Change of status of a node in consecutive time periods.

(though for relatively small networks) to demonstrate the value of our proposed formulation and the potential enhancements on the run time and the optimality gap. We show that most of our proposed techniques are useful for practical applications.

In the computational study, we impose a run time limit of 30 min for all our experiments and try to solve instances with up to 200 nodes for a planning horizon of 60 days. Although our instances are reasonably sized, our approach is not yet scalable for analyzing a real life epidemic or pandemic outbreaks which may easily include millions of nodes. However, our proposed formulation can be used for smaller communities such as rural communities which may include hundreds of people.

The rest of paper is organized as follows. In Section 2, we provide a detailed description of the problem. In Section 3, a non-linear integer programming formulation for the problem is introduced. In Section 4, we show how the non-linear formulation can be linearized. In Section 5, we explain the key concepts of the proposed formulation on a small example. In Section 6, we introduce a few potential enhancement techniques to improve our proposed formulation. In Section 7, we present the result of a computational study. Finally, in Section 8, we give some concluding remarks.

2. Problem description

Given a (dynamic) network of people, i.e. the nodes representing individuals and the links representing their interactions, we remove a subset of nodes with some certain characteristics over time to minimize the spread of influenza virus infections (with large penalties for fatality). Specifically, the mathematical formulation, that we develop, chooses a node to be removed only if it is sick. Note that selecting and removing a sick node can be interpreted as isolating that node. Later we show that our formulation can be easily modified to select a node to be removed only if it is healthy and susceptible. It is worth mentioning that selecting and removing a healthy and susceptible node can be interpreted as vaccinating that node.

We denote by $\mathcal{N} := \{1, \dots, N\}$ the index set of nodes in the network and by $\mathcal{T} := \{1, \dots, T\}$ the index set of time periods in the planning horizon. The main assumption of this study is that there is only one strain of virus circulating, and there is no mutation or reassortment of the virus during the entire planning horizon. The status of each node $i \in \mathcal{N}$ can be changed in consecutive time periods, i.e. $t, t+1 \in \mathcal{T}$, as shown in Fig. 1. Next, we explain the key components of this figure in detail.

A healthy node can be either *immune* (a node that has recovered after infection) or *not-immune* (susceptible). A healthy and immune node cannot become infected and consequently cannot infect other people. However, a healthy and susceptible node may become infected depending on the *total risk of the network* (this concept will be formally defined later in this section) for that particular node. Specifically, if the total risk of the network for a healthy and susceptible node is small at $t \in \mathcal{T}$ then that node remains healthy and susceptible in the next time period, i.e. $t+1$. However, if the total risk of the network is medium or large for a healthy and susceptible node at $t \in \mathcal{T}$ then that node will be considered as an infected node in the next time period, i.e. $t+1$. An infected node will be considered a sick node only after completing its transition state, i.e. a number of time periods that the infection is evolving without having the capability of infecting other nodes. The transition state is sometimes called as the *latency period* in the literature [18]. We denote the number of time periods in the transition state by $t_w \in \mathbb{N}$ (a natural number). It is worth mentioning that t_w varies because it is estimated to follow a Weibull distribution with the power parameter of 2.21 and the scale parameter of 1.10 [19]. However, since in this study, our goal is to develop a deterministic model, we simply assume that $t_w = 3$ for our computational experiments.

Sick and recovering nodes are those that initially were infected because the total risk of the network (for each of them) was medium. These nodes will remain sick for a few time periods but they will finally become healthy and immune. It is clear that as soon as we realize that a node is healthy and immune, we can assume that it does not exist in the network for the subsequent time periods since that node cannot become infected again and it cannot make other people infected. However, sick and dying nodes are those that initially were infected because the total risk of the network (for each of them) was large. These nodes will remain sick for a few time periods but they will finally die. Again, as soon as we realize that a node is dead, we can assume that it does not exist in the network for the subsequent time periods.

We now formally introduce the concept of 'the total risk of the network' for a given node. In general, based on the findings of Ferguson et al. [18,20], two important factors that can indicate whether a healthy and susceptible node $i \in \mathcal{N}$ has become infected are (1) its *level of interactions* with the sick nodes and (2) the *contagious degree* of the sick nodes that have some interactions with node i. The latter, i.e. the contagious degree, quantifies the amount of the virus that can be spread by a node.

It is evident that at time period $t \in \mathcal{T}$, the level of interactions of node $i \in \mathcal{N}$ with node $j \in \mathcal{N} \setminus \{i\}$, denoted by $w_{ijt} \in \mathbb{Z}_{\geq}$ (where $\mathbb{Z}_{\geq} := \{s \in \mathbb{Z} : s \geq 0\}$), can be simply viewed as the weight of the link/arc between them. Similarly, at time period $t \in \mathcal{T}$, the contagious degree of node $i \in \mathcal{N}$, denoted by $c_{it} \in \mathbb{Z}_{\geq}$, can be viewed as the weight of node i. So, the total risk of the network for node $i \in \mathcal{N}$ at time period $t \in \mathcal{T}$ is defined as follows:

$$z_{it} := \begin{cases} \sum\limits_{j \in \mathcal{N} \setminus \{i\}} w_{jit} c_{jt} & \text{if node } i \text{ is healthy and susceptible,} \\ 0 & \text{otherwise.} \end{cases}$$

It is worth mentioning that based on the findings of Ferguson et al. [18,20], the contagious degree has a dynamic behavior. More specifically, the contagious degree of a sick node in the first time period of its illness is different from the second time period and so on. Therefore, $c_{it} = 0$ if node i is isolated or not sick at time period t. Otherwise, $c_{it} \in \{d_1, \ldots, d_{t_s}\}$ where d_k is the contagious degree of a (not-isolated) sick node at time period k of its illness and $t_s \in \mathbb{N}$ is the maximum number of time periods that a node remains sick.

In practice, the contagious degree can be estimated by using the lognormal distribution with parameters -0.72 and 1.18 [18,20]. Accordingly, t_s varies but one can simply assume that $t_s = 3$. Therefore, in practice, $d_1 \approx 2$, $d_2 \approx 0.25$ and $d_3 \approx 0.125$. Given that the formulation that we will develop is an integer linear program, it is computationally advantageous if we change these number to integers. Consequently, for our computational experiments, we simply set $d_1 = 2 \times 8 = 16$, $d_2 = 0.25 \times 8 = 2$, and $d_3 = 0.125 \times 8 = 1$.

Next, we explain how we can determine whether the total risk of the network for a given node is small, medium or large. Let $\beta_1 = 0$ and $\beta_2, \beta_3 \in \mathbb{N}$ be some user-defined parameters such that,

$$0 = \beta_1 < \beta_2 < \beta_3.$$

Let M be a sufficiently large value, i.e. an upper bound for the maximum possible value of z_{it} for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$. For any given $i \in \mathcal{N}$ and $t \in \mathcal{T}$, we say that z_{it} is small, medium, and large if $z_{it} \in [\beta_1, \beta_2 - 1]$, $z_{it} \in [\beta_2, \beta_3 - 1]$, and $z_{it} \in [\beta_3, M]$, respectively (we assume that $\beta_3 \leq M$).

3. A nonlinear formulation

In this section, we develop a nonlinear formulation for the problem. We assume that all nodes are healthy and susceptible at the beginning, i.e. $t = 1$. The following decision variables are used in the nonlinear formulation:

- x_{it} is a binary decision variable for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$. If a sick node i is selected for isolation at time period t then $x_{it} = 1$, and $x_{it} = 0$ otherwise.
- z_{it} is an integer decision variable for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$. This variable captures the value of the total risk of the network for node i at time period t. Note that we have previously defined this notation in Section 2 but here, we treat it as a decision variable since its value has to be determined by the formulation.
- $y_{it}^{(1)}$ is a binary decision variable for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$. If $z_{it} \in [\beta_1, \beta_2 - 1]$ then $y_{it}^{(1)} = 1$, and $y_{it}^{(1)} = 0$ otherwise.
- $y_{it}^{(2)}$ is a binary decision variable for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$. If $z_{it} \in [\beta_2, \beta_3 - 1]$ then $y_{it}^{(2)} = 1$, and $y_{it}^{(2)} = 0$ otherwise.
- $y_{it}^{(3)}$ is a binary decision variable for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$. If $z_{it} \in [\beta_3, M]$ then $y_{it}^{(3)} = 1$, and $y_{it}^{(3)} = 0$ otherwise.

Let $\lambda \in \mathbb{N}$ be a user-defined parameter. The objective function of the problem can be defined as follows:

$$\min \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} (y_{it}^{(2)} + \lambda y_{it}^{(3)}). \tag{1}$$

The objective function simply minimizes the weighted summation of the total number of infected nodes. It is worth mentioning that based on the reports of World Health Organization (WHO), see for instance [21], the influenza mortality rate can be even up to around 4% of the infected nodes. So, in our computational experiments in this paper, we simply assume that from every 26 infected nodes, one may die. Consequently, we simply set $\lambda = 25$ for all our computational experiments. This implies that from the view point of the objective function, the cost of one sick and dying node is equivalent to the cost of 25 recovering and sick nodes.

Next, we introduce the constraints of the problem. For each $i \in \mathcal{N}$, we have,

$$\sum_{t \in \mathcal{T}} x_{it} \leq 1. \tag{2}$$

This constraint ensures that node i can be selected for isolation at most once in the entire planning horizon. Note that after choosing a node for isolation that node does not technically exist in the network until the end of the planning horizon since it cannot threat any other node. For each $i \in \mathcal{N}$ and $t \in \mathcal{T}$, we have,

$$\sum_{l=1}^{3} y_{it}^{(l)} = 1. \tag{3}$$

This constraint ensures that the total risk of the network for node i at time period t, i.e. z_{it}, is either small or medium or large. For each $i \in \mathcal{N}$, we have,

$$\sum_{t \in \mathcal{T}} y_{it}^{(2)} + y_{it}^{(3)} \leq 1. \tag{4}$$

This constraint guarantees that each person can be infected at most once in the entire planning horizon. Next, we introduce a notation to facilitate the presentation of some of the remaining constraints. For each $i \in \mathcal{N}$ and $t \in \mathcal{T}$, we define,

$$s_{it} := \sum_{t'=\max\{t-t_w-t_s,1\}}^{t-t_w-1} y_{it'}^{(2)} + \sum_{t'=\max\{t-t_w-t_s,1\}}^{t-t_w-1} y_{it'}^{(3)}.$$

It is not hard to see that $s_{it} \in \{0,1\}$. More specifically if node i is sick at time period t then $s_{it} = 1$ and $s_{it} = 0$ otherwise. Note that node i is sick at time period t only if it has been infected at time periods $t-t_w-t_s, \dots, t-t_w-1$. As an aside, by definition, t' cannot be non-positive, and that is the reason that the term '$\max\{t-t_w-t_s,1\}$' is used in the equation.

By using the new notation, we can now introduce two of the remaining constraints. For each $i \in \mathcal{N}$ and $t \in \mathcal{T}$, we have,

$$x_{it} \le s_{it}. \tag{5}$$

This constraint guarantees that node i can only be selected for isolation at time period t if it is sick at that time period. For each $t \in \mathcal{T}$, we have,

$$\sum_{i\in\mathcal{N}} x_{it} \le \alpha \sum_{i\in\mathcal{N}} s_{it}, \tag{6}$$

where $\alpha \in [0,1]$ is a user-defined parameter. In practice, isolating all sick nodes may not be possible due to the lack of resources or even lack of detecting all sick nodes. However, decision makers may attempt to provide more resources if they see that the number of sick nodes is increasing. One way to capture these observations is to use Constraint (6). This constraint ensures that at time period $t \in \mathcal{T}$, the ratio of sick nodes selected for isolation to the total number of sick nodes cannot be more than α. Note that Constraint (6) can also be viewed differently. We know that, in practice, not all sick nodes may be discovered immediately (for example because the symptoms may have not been appeared), and so α can be viewed as a cap for the ratio of the sick nodes that can be discovered in each time period. Nevertheless, one may use a different budget constraint instead of Constraint (6).

We assume that the total risk of the network for node $i \in \mathcal{N}$ at time period $t = 1$ is given as a parameter, denoted by $\textsc{Data}_i \in \mathbb{Z}_{\geq}$. So, for each $i \in \mathcal{N}$, we add the following constraint,

$$z_{i1} = \textsc{Data}_i. \tag{7}$$

We previously introduced the notation c_{jt} in Section 2 for each $j \in \mathcal{N}$ and $t \in \mathcal{T}$. This notation simply captures the contagious degree of node j at time period t as follows,

$$c_{jt} := (1 - \sum_{t'=1}^{t} x_{jt'})(\sum_{k=1}^{\min\{t_s,t-t_w-1\}} d_k y_{j,t-t_w-k}^{(2)} + \sum_{k=1}^{\min\{t_s,t-t_w-1\}} d_k y_{j,t-t_w-k}^{(3)}).$$

Note that, by Constraint (2), we know that $1 - \sum_{t'=1}^{t} x_{jt'} \in \{0,1\}$. However, if $1 - \sum_{t'=1}^{t} x_{jt'} = 0$ then node j has been isolated and so it cannot spread the infection at time period t. Also, if node j is sick at time period t, and this node is at time period $k \in \{1,\dots,t_s\}$ of its illness then it must have been infected at time period $y_{j,t-t_w-k}^{(2)}$. By these observations, it is not hard to see that $c_{jt} = 0$ if node j has been isolated or it is not sick, and otherwise $c_{jt} = d_k$ if it is at time period $k \in \{1,\dots,t_s\}$ of its illness. Note that we have used the term '$\min\{t_s, t-t_w-1\}$' in the equation since, by definition, we must have that $t - t_w - k > 0$. By using this notation, for each $i \in \mathcal{N}$ and $t \in \mathcal{T} \setminus \{1\}$, we add the following constraint,

$$z_{it} = (1 - \sum_{t'=1}^{t-1} y_{it'}^{(2)} - \sum_{t'=1}^{t-1} y_{it'}^{(3)}) \sum_{j\in\mathcal{N}\setminus\{i\}} w_{jit} c_{jt}. \tag{8}$$

Note that by Constraint (4), we know that $(1-\sum_{t'=1}^{t-1} y_{it'}^{(2)} - \sum_{t'=1}^{t-1} y_{it'}^{(3)}) \in \{0,1\}$. So, if $(1-\sum_{t'=1}^{t-1} y_{it'}^{(2)} - \sum_{t'=1}^{t-1} y_{it'}^{(3)}) = 0$ then node i is not healthy and susceptible at time period t, and so by definition, we must have $z_{it} = 0$. Finally, for each $i \in \mathcal{N}$ and $t \in \mathcal{T} \setminus \{1\}$, we add the following constraints,

$$y_{it}^{(l)} \beta_l \le z_{it} \qquad \forall l \in \{1,2,3\}, \tag{9}$$

$$z_{it} \le (\beta_{l+1} - 1 - M) y_{it}^{(l)} + M \qquad \forall l \in \{1,2\}. \tag{10}$$

These constraints determine whether z_{it} is small or medium or large. Note that Constraint (3) implies that for each $i \in \mathcal{N}$ and $t \in \mathcal{T}$, exactly one of $y_{it}^{(1)}$, $y_{it}^{(2)}$ and $y_{it}^{(3)}$ is one. So, it is easy to see from Constraints (9) and (10) that if $y_{it}^{(1)} = 1$ then $z_{it} \in [\beta_1, \beta_2 - 1]$. Also, if $y_{it}^{(2)} = 1$ then $z_{it} \in [\beta_2, \beta_3 - 1]$. Finally, if $y_{it}^{(3)} = 1$ then $z_{it} \in [\beta_3, M]$. Note that by definition, we always have $z_{it} \le M$.

4. Linearizing the formulation

In the proposed nonlinear formulation, the only non-linear constraint is (8). So, in this section, in order to obtain an integer linear programming formulation, Constraint (8) is replaced with some linear constraints after introducing some new binary decision variables. We first note that the notation c_{jt} captures the value of a nonlinear function for each $j \in \mathcal{N}$ and $t \in \mathcal{T}$. So, we first linearize that function.

Let $\bar{y}_{jtk}^{(2)}$ be a new binary decision variable for each $j \in \mathcal{N}$, $t \in \{t_w + k + 1, \dots, T\}$ and $k \in \{1, \dots, t_s\}$. We define $\bar{y}_{jtk}^{(2)} := (1 - \sum_{t'=1}^{t} x_{jt'}) y_{j,t-t_w-k}^{(2)}$ and this non-linear equation is equivalent to the following linear constraints:

$$\bar{y}_{jtk}^{(2)} \le y_{j,t-t_w-k}^{(2)}, \tag{11}$$

$$\bar{y}_{jtk}^{(2)} \le 1 - \sum_{t'=1}^{t} x_{jt'}, \tag{12}$$

$$\bar{y}_{jtk}^{(2)} \ge y_{j,t-t_w-k}^{(2)} - \sum_{t'=1}^{t} x_{jt'}. \tag{13}$$

Obviously, based on these constraints, if $y_{j,t-t_w-k}^{(2)} = 0$ or $\sum_{t'=1}^{t} x_{jt'} = 1$ then $\bar{y}_{jtk}^{(2)} = 0$, and $\bar{y}_{jtk}^{(2)} = 1$ otherwise. Similarly, let $\bar{y}_{jtk}^{(3)}$ be a new binary decision variable for each $j \in \mathcal{N}$, $t \in \{t_w + k + 1, \dots, T\}$ and $k \in \{1, \dots, t_s\}$. We define $\bar{y}_{jtk}^{(3)} := (1 - \sum_{t'=1}^{t} x_{jt'}) y_{j,t-t_w-k}^{(3)}$ and this non-linear equation is equivalent to the following linear constraints:

$$\bar{y}_{jtk}^{(3)} \le y_{j,t-t_w-k}^{(3)}, \tag{14}$$

$$\bar{y}_{jtk}^{(3)} \le 1 - \sum_{t'=1}^{t} x_{jt'}, \tag{15}$$

$$\bar{y}_{jtk}^{(3)} \ge y_{j,t-t_w-k}^{(3)} - \sum_{t'=1}^{t} x_{jt'}. \tag{16}$$

In the remaining of this section, we assume that these linear constraints are added to the formulation. So, for each $j \in \mathcal{N}$ and $t \in \mathcal{T}$, we can now redefine the notation c_{jt} as follows:

$$c_{jt} = \sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k \bar{y}_{jtk}^{(2)} + \sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k \bar{y}_{jtk}^{(3)}.$$

Consequently, the notation c_{jt} now captures the value of a linear function for each $j \in \mathcal{N}$ and $t \in \mathcal{T}$. Hence, in the remaining of this section, whenever we use the notation c_{jt}, readers should simply replace it with the linear function presented above.

Next, we linearize Constraint (8). This can be done by simply replacing Constraint (8) by the following set of linear constraints:

$$z_{it} \leq \sum_{j\in\mathcal{N}\setminus\{i\}} w_{jit}c_{jt}, \tag{17}$$

$$z_{it} \leq M(1-\sum_{t'=1}^{t-1} y^{(2)}_{it'} - \sum_{t'=1}^{t-1} y^{(3)}_{it'}), \tag{18}$$

$$z_{it} \geq \sum_{j\in\mathcal{N}\setminus\{i\}} w_{jit}c_{jt} - M(\sum_{t'=1}^{t-1} y^{(2)}_{it'} + \sum_{t'=1}^{t-1} y^{(3)}_{it'}). \tag{19}$$

Obviously, if $(1-\sum_{t'=1}^{t-1}y^{(2)}_{it'} - \sum_{t'=1}^{t-1}y^{(3)}_{it'}) = 0$ then $z_{it}=0$, and $z_{it} = \sum_{j\in\mathcal{N}\setminus\{i\}}w_{jit}c_{jt}$ otherwise. So, we were able to reformulate the problem as an integer linear program. Next we make a few comments:

- The proposed formulation is written to optimally choose a subset of sick nodes for isolation. However, this formulation can be easily modified for choosing a subset of healthy and susceptible nodes for vaccination. In order to do so, for each $i\in\mathcal{N}$ and $t\in\mathcal{T}$, Constraint (5) should be replaced by the following constraint,

$$x_{it} \leq 1-(\sum_{t'=1}^{t} y^{(2)}_{it'} + \sum_{t'=1}^{t} y^{(3)}_{it'}).$$

  Also, in this case, one may want to replace Constraint (6) by some other budget constraint.
- Let $u := \max\{d_1,\dots,d_{t_s}\}$. Note that, in practice, based on our discussion in Section 2, we have $u=16$. So, for each $i\in\mathcal{N}$, we define $z^{max}_{i1} := \textsc{Data}_i$ and $z^{max}_{it} := \sum_{j\in\mathcal{N}\setminus\{i\}}w_{jit}u$ for $t\in\mathcal{T}\setminus\{1\}$. It is evident that, by definition, M can be safely set to $\max\{z^{max}_{it} : i\in\mathcal{N},\ t\in\mathcal{T}\}$. It is also worth mentioning that, in Constraints (18) and (19), M can be simply replaced by z^{max}_{it}.

5. An example

In this section, we present a simple example with eight nodes, i.e. $N=8$. We assume that the number of time periods in the planing horizon is twenty, i.e. $T=20$. Let

$$A = \begin{bmatrix} 0&1&1&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&0&1&1&1\\ 1&0&0&0&1&0&0&0\\ 0&1&0&1&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0 \end{bmatrix}.$$

We assume that $w_{ji1}=\cdots=w_{jiT}=A_{ji}$ for each $i,j\in\mathcal{N}$. Based on this assumption, Fig. 2 shows the level of interaction between nodes at each time period. We also assume that $t_w=3$, $t_s=3$, $d_1=16$, $d_2=2$, $d_3=1$, $\textsc{Data}_1=16$, $\textsc{Data}_2=\cdots=\textsc{Data}_8=0$, $\beta_1=0$, $\beta_2=10$, $\beta_3=30$, and $\alpha=0.5$.

Fig. 3 illustrates the changes of the network from $t=1$ to $t=13$ for a non-optimal solution. We use dashed circles for the nodes that become infected. It is worth mentioning that at $t=1$, we only assume that Node 1 becomes infected as shown in Fig. 2. When a node is in its transition state, it is shaded (with green color). Furthermore, if a node is sick then its contagious degree is given. A node is crossed if it is isolated,

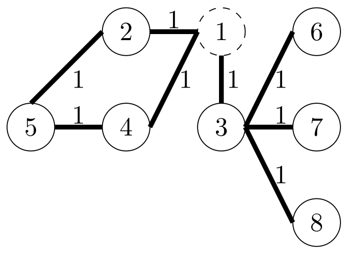


Fig. 2. The level of interactions between nodes at each time period.

and also filled by black color if it is healthy and immune. Note that since $\alpha = 0.5$, no more than 50% of sick nodes in each time period can be selected for isolation. So, in this solution only Node 3 is selected for isolation at $t = 9$. We note that this solution is not optimal since the total risk of the network for Node 5 at $t = 9$ is large, i.e. 32, and so this node will die. Note too that our assumption is that the penalty of any death incident is significant, i.e. $\lambda = 25$.

Fig. 4 illustrates the changes of the network from $t = 8$ to $t = 13$ for an optimal solution. Note that the changes of the network from $t = 2$ to $t = 7$ are as same as those of Fig. 3. In this solution, only Node 4 is selected for isolation at $t = 9$. It is clear that no node dies by using this solution, but all nodes are infected.

6. Potential enhancements

In this section, we introduce a few techniques that may result in reducing the solution time of integer linear programming solvers for the proposed formulation.

6.1. Variable fixing

Since we have assumed that all nodes are healthy and susceptible at $t = 1$, the first sick node can be observed no earlier than $t = t_w + 2$. Consequently, the following constraints can be added,

$$x_{it} = 0 \qquad \forall i \in \mathcal{N},\ t \in \{1, \dots, t_w + 1\}, \tag{20}$$

$$y_{it}^{(1)} = 1 \qquad \forall i \in \mathcal{N},\ t \in \{2, \dots, t_w + 1\}. \tag{21}$$

Also, let $\mathcal{N}^1 := \{i \in \mathcal{N} : \textsc{Data}_i \in [\beta_1, \beta_2 - 1]\}$, $\mathcal{N}^2 := \{i \in \mathcal{N} : \textsc{Data}_i \in [\beta_2, \beta_3 - 1]\}$ and $\mathcal{N}^3 := \{i \in \mathcal{N} : \textsc{Data}_i \in [\beta_3, M]\}$. So, the following constraints can be added:

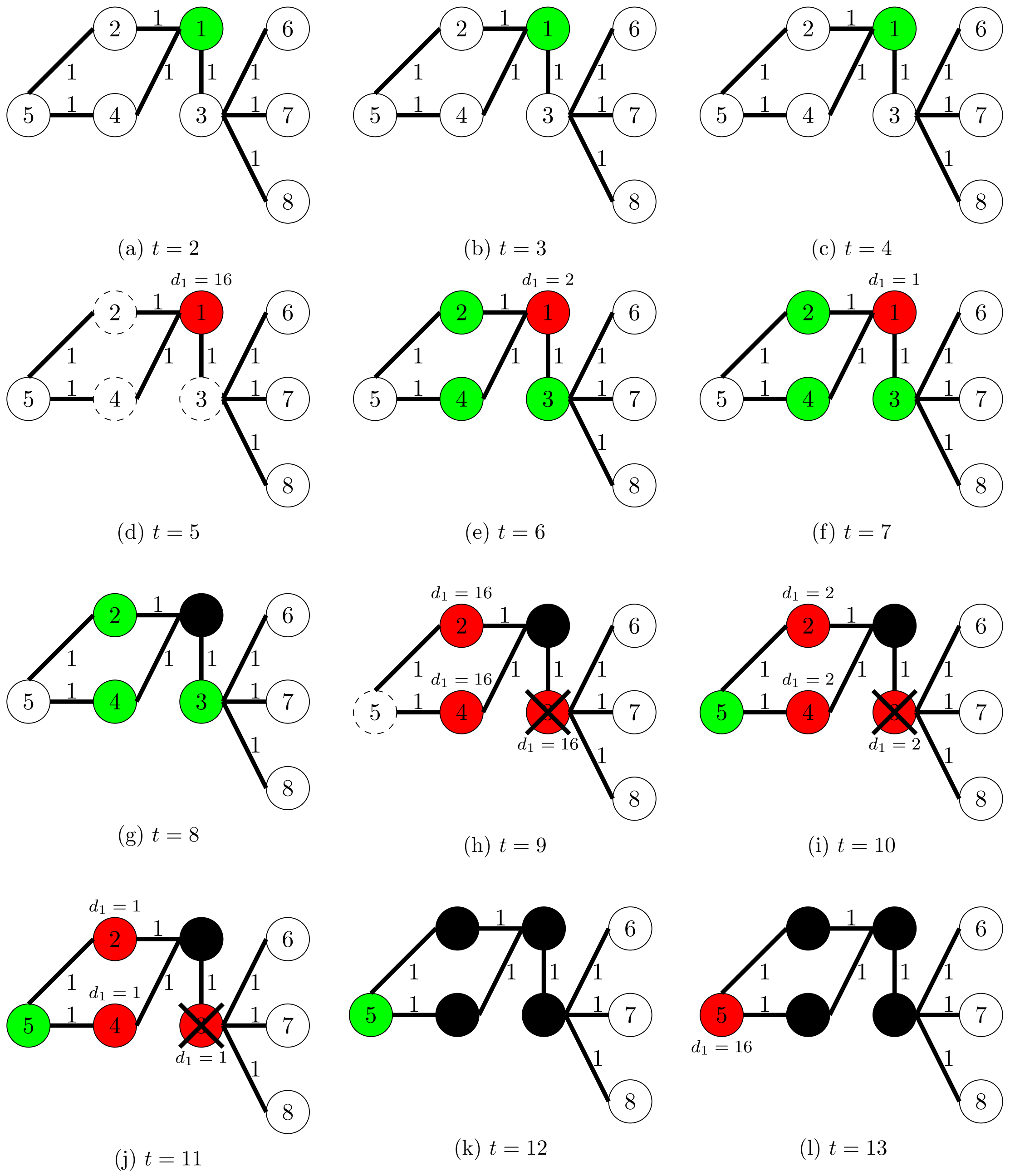
$$y_{i1}^{(1)} = 1 \qquad \forall i \in \mathcal{N}^1, \tag{22}$$

$$y_{i1}^{(2)} = 1 \qquad \forall i \in \mathcal{N}^2, \tag{23}$$

$$y_{i1}^{(3)} = 1 \qquad \forall i \in \mathcal{N}^3. \tag{24}$$

Let S^t be the set of all nodes that may become infected at time period $t \in \mathcal{T}$ in the worst case scenario, i.e. when we do not isolate anyone. We now show that S^t can be constructed recursively. Obviously, $S^1 = \mathcal{N} \setminus \mathcal{N}^1$ and $S^2 = \cdots = S^{t_w+1} = \emptyset$. For each $t \in \{t_w + 2, \dots, T\}$, we have,

$$S^t = \bigcup_{k=1:\ t-t_w-k \geq 1}^{t_s} \{i \in \mathcal{N} : \exists j \in S^{t-t_w-k} \text{ with } w_{jit} > 0\}.$$



(a) $t = 2$ (b) $t = 3$ (c) $t = 4$

(d) $t = 5$ (e) $t = 6$ (f) $t = 7$

(g) $t = 8$ (h) $t = 9$ (i) $t = 10$

(j) $t = 11$ (k) $t = 12$ (l) $t = 13$

Fig. 3. A non-optimal solution in which Node 5 dies and 5 nodes are infected in total. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

More specifically, in the worst case scenario, any node that has a positive interaction with any sick node will become infected. Note that we used the union symbol since sick nodes can be in different time periods of their illness, i.e. $\{1, \ldots, t_s\}$. Now, for each $t \in \{t_w + 2, \ldots, T\}$ and $i \in \mathcal{N} \setminus S^t$, the following constraints can be added,

$$z_{it} = 0, \tag{25}$$

$$y_{it}^{(1)} = 1. \tag{26}$$

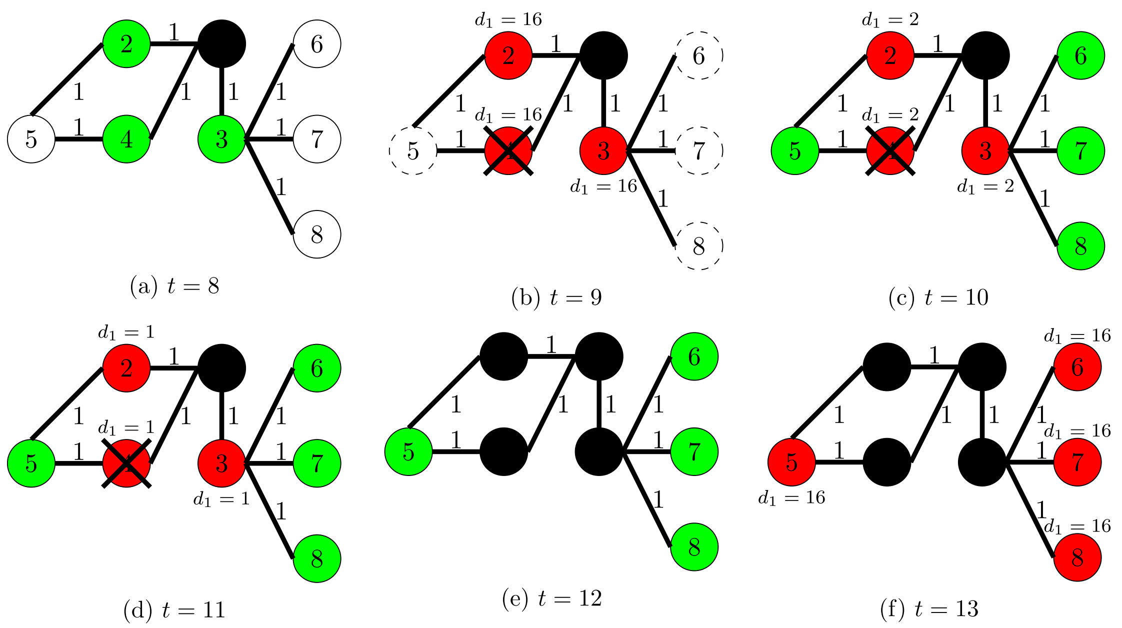


Fig. 4. An optimal solution in which all nodes are infected.

6.2. Moderating big-M coefficients

Linear programming relaxation of an integer linear programming formulation with disjunctive sets may be improved by moderating its big-M parameters. We first show that the value of z_{it}^{max} can be improved. For each $i \in \mathcal{N}$, we denote the improved value by,

$$\widehat{z}_{it}^{max} := \textsc{Data}_i \qquad \text{if } t = 1, \tag{27}$$

$$\widehat{z}_{it}^{max} := \sum_{j \in \widehat{S}^t \setminus \{i\}} w_{jit} u \qquad \text{if } t \in \mathcal{T} \setminus \{1\}, \tag{28}$$

where

$$\widehat{S}^t := \bigcup_{k=1:\ t-t_w-k \geq 1}^{t_s} S^{t-t_w-k}$$

is the set of all nodes that may be sick at time period t. Similar to our discussion at the end of Section 4, by definition, M can now be safely set to $\max\{\widehat{z}_{it}^{max} : i \in \mathcal{N},\ t \in \mathcal{T}\}$. Also, in Constraints (18) and (19), M can be simply replaced by $\widehat{z}_{it}^{max}$.

6.3. Symmetry breaking

An integer linear program is *symmetric* if its variables can be permuted without changing the structure of the problem [22,23]. Our proposed integer programming formulation is symmetric because of the following reason.

Suppose that in a given feasible solution, node $i \in \mathcal{N}$ is selected to be isolated at time period $t \in \mathcal{T} \setminus \{1, \ldots, t_w + 2\}$. It is evident that if the time period t is not the first day of the illness for node i, and $\sum_{i \in \mathcal{N}} x_{i,t-1} < \alpha \sum_{i \in \mathcal{N}} s_{i,t-1}$, then we can isolate node i at time period $t - 1$ (without making the objective value worse) since $d_1 > d_2 > d_3$. So, to avoid this, for each $i \in \mathcal{N}$ and $t \in \mathcal{T} \setminus \{1, \ldots, t_w + 2\}$, we can add the following inequalities to our model if $|\widehat{S}^{t-1}| > 0$:

$$x_{it} - y^{(2)}_{i,t-t_w-1} - y^{(3)}_{i,t-t_w-1} - (1 - \frac{\alpha \sum_{i\in\mathcal{N}} s_{i,t-1} - \sum_{i\in\mathcal{N}} x_{i,t-1}}{|\widehat{S}^{t-1}|}) \leq x_{i,t-1}. \tag{29}$$

In Inequality (29), if $x_{it} - y^{(2)}_{i,t-t_w-1} - y^{(3)}_{i,t-t_w-1} > 0$ then we know that the time period t is not the first day of illness for node i. Also, if $\frac{\alpha \sum_{i\in\mathcal{N}} s_{i,t-1} - \sum_{i\in\mathcal{N}} x_{i,t-1}}{|\widehat{S}^{t-1}|} > 0$ (and $|\widehat{S}^{t}| > 0$), then we know that $\sum_{i\in\mathcal{N}} x_{i,t-1} < \alpha \sum_{i\in\mathcal{N}} s_{i,t-1}$. Also, note that by definition, $\widehat{S}^t$ is the set of all nodes that may be sick at time period $t \in \mathcal{T}$ in the worst case scenario. So, if $|\widehat{S}^t| > 0$ then

$$\frac{\alpha \sum_{i\in\mathcal{N}} s_{i,t-1} - \sum_{i\in\mathcal{N}} x_{i,t-1}}{|\widehat{S}^{t-1}|} \in [0, 1].$$

Consequently, the inequality (29) implies that if $x_{it} - y^{(2)}_{i,t-t_w-1} - y^{3}_{i,t-t_w-1} > 0$ and $\sum_{i\in\mathcal{N}} x_{i,t-1} < \alpha \sum_{i\in\mathcal{N}} s_{i,t-1}$, then $x_{i,t-1} > 0$. Obviously, this cannot be true for integer feasible solutions based on Constraint (2).

7. Computational results

To evaluate the performance of the proposed formulation and the potential enhancements, a computational study is conducted. We use Julia (JuMP) to implement the formulation and enhancements and use GUROBI 7.0.1 as the integer linear programming solver. All computational experiments are carried out on a Dell PowerEdge R630 with two Intel Xeon E5-2650 2.2 GHz 12-Core Processors (30 MB), 128 GB RAM, and the RedHat Enterprise Linux 6.8 operating system, and using eight threads. We impose a run time limit of 1800 s for each experiment in this computational study. It is worth mentioning that, unfortunately, in the literature, there is no standard dataset (to the best of our knowledge). In addition to this fact, we are the first studying minimizing the spread of influenza virus by using node removal strategy and incorporating many practical aspects. This makes finding standard instances even harder and so, in this computational study, random instances are generated.

We now explain how the values of β_2 and β_3 are determined in this computational study. Note that, by our assumptions, $\beta_1 = 0$. However, unfortunately, computing the values of β_2 and β_3 are not obvious. We first explain how the value of β_3 is determined. It is clear that if we set the value of β_3 too high, the rate of mortality is probably always zero even if we do not isolate anyone. On the other hand, if we set it too low, many nodes will die. Both of these cases are not reasonable since WHO reports that for an influenza virus the mortality rate may be up to around 4% of the total number of infected people [21]. So, in our computational study, we simply set

$$\beta_3 = \lfloor \frac{\sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} \widehat{z}^{max}_{it}}{N \times T} \rfloor,$$

Note that $\widehat{z}^{max}_{it}$ is the maximum total risk of the network for node i at time period t. So, we basically set β_3 to the average total risk of the network for each node at any time (by assuming that the sick nodes have the highest contagious degree).

Similar to β_3, if we set the value of β_2 too high, not many nodes can become infected. On the other hand, if we set it too low, a lot of nodes can become infected. Both of these cases are not reasonable since WHO reports that for an influenza virus, the infection rate may be up to around 50% − 60% of the total population size [21]. So, in our computational study, we simply set

$$\beta_2 = \lfloor \frac{\sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} \frac{\widehat{z}^{max}_{it}}{\max\{\sum_{j\in\widehat{S}^t\setminus\{i\}} I(w_{jit}),\, 1\}}}{N \times T} \rfloor,$$

where $I(w_{jit})$ is a binary function that takes the value of one if $w_{jit} > 0$, and zero otherwise. So, we basically set β_2 to the average risk for each node at any time from each other node with some interactions (by assuming that all such nodes are sick and have the highest contagious degree). In this case, we assure that a healthy and susceptible node i can become infected by a sick node j at time period t only if it has a (relatively) strong interaction with the sick node.

Next, we explain how a test instance is generated in this computational study, i.e. how the values of N, T, α, w_{jit} for each $i, j \in \mathcal{N}$ and $t \in \mathcal{T}$, and $\textsc{Data}_i$ for each $i \in \mathcal{N}$ are generated.

In our test instances, we set $T = 60$ (days) and the population size N is either 60 or 70 or 200. We also assume that α is either 0.5 or 0.75. Note that, by definition, it is expected that the spread of the infection to be controlled faster for higher values of α.

In order to generate the values of w_{jit} for each $i, j \in \mathcal{N}$ and $t \in \mathcal{T}$, we first create an undirected graph, denoted by $G(V, E)$, as follows:

- Step 0: We first generate N nodes.
- Step 1: We randomly create a Hamiltonian path, i.e. a path that visits each node exactly once (and contains all nodes). This step ensures that the graph is connected.
- Step 2: For each node, $i \in \mathcal{N}$, we randomly generate a number showing its *minimum degree*, i.e. the minimum number of edges incident to that node, from a discrete uniform distribution on the interval $[2, \lfloor 0.05 \times N \rfloor]$.
- Step 3: We now explore the nodes one by one. For node $i \in \mathcal{N}$, we add extra randomly generated edges to the partially constructed graph (if necessary) until the minimum number of edges incident to that node is at least equal to its corresponding minimum degree.

Graph $G(V, E)$ shows that which nodes have interactions with node i for each $i \in \mathcal{N}$. It is worth mentioning that, on average, the number of edges incident to each node is about $0.08N$ for the graphs constructed by this method in this paper. After creating the graph, we generate the values of w_{jit} for all $i, j \in \mathcal{N}$ and $t \in \mathcal{T}$ as follows. We first set $w_{jit} = 0$ for each $i, j \in \mathcal{N}$ and $j \in \mathcal{T}$. For each edge $(i, j) \in E$, we then randomly generate two numbers from a discrete uniform distribution on the interval $[1, 10]$ and assign them to w_{jit} and w_{ijt}, respectively. For the sake of simplicity, we assume that $w_{ji1} = w_{ji2} = \cdots = w_{jiT}$ for all $i, j \in \mathcal{N}$.

To avoid generating trivial instances, we assume that at the beginning (approximately) 4% of the nodes are infected. To accomplish that, we first set all $\textsc{Data}_i = 0$. We then randomly select $\lfloor 0.04 \times N \rfloor$ nodes and set their corresponding $\textsc{Data}_i$ to 160. Note that in our test instances $w_{jit} \leq 10$ for all $i, j \in \mathcal{N}$ and $j \in \mathcal{T}$, and also $u = 16$. Therefore, by construction of β_2 in this computational study, we must have that $\beta_2 \leq 10 \times 16 = 160$. Consequently, by setting $\textsc{Data}_i = 160$ it is guaranteed that node i becomes infected.

7.1. Analyzing the optimal solution generated by solving the model

In this section, we analyze the optimal solution generated by solving the model. The goal is to numerically show that how an optimal solution may change if we have initially created the network of people incorrectly in a sense that some arcs are missing. It is evident that if an optimal solution does not change significantly (or it is still partially correct) then that would be interesting from the practical point of view. Because that is an indication that as long as the generated network is a reasonable estimation of real network, an optimal solution of the estimated network would be almost optimal for the real network.

So, in this section, we first create a single network with $N = 60$ and $\alpha = 0.75$, and based on that we generate 25 instances by just randomly selecting the nodes that are supposed to initially become infected, i.e. we set $\textsc{Data}_i = 160$ for those nodes. Note that based on our assumptions, only about 4% of the nodes are supposed to initially become infected in this computational study. This implies that only $\lfloor 60 \times 0.04 \rfloor = 2$ nodes are randomly selected to initially become infected for each instance.

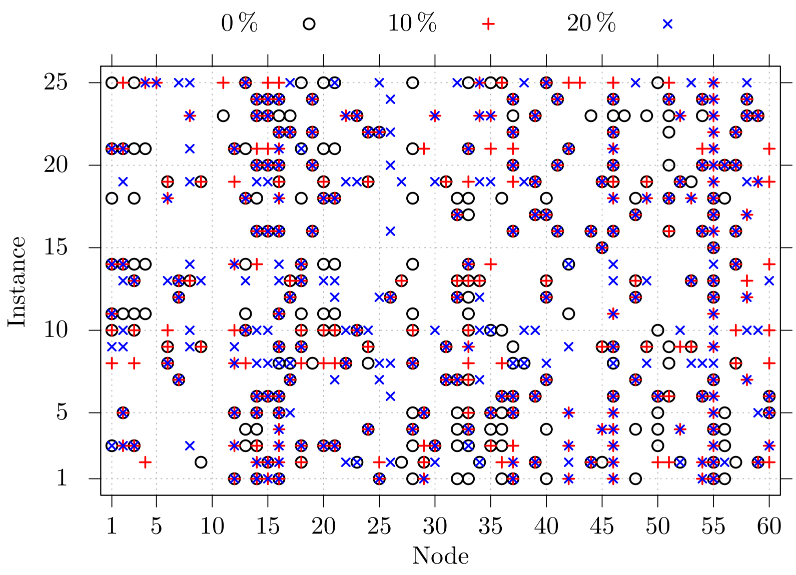


Fig. 5. Infected nodes in the optimal solution obtained by solving the proposed model for each instance when adding 0%, 10%, and 20% extra arcs.

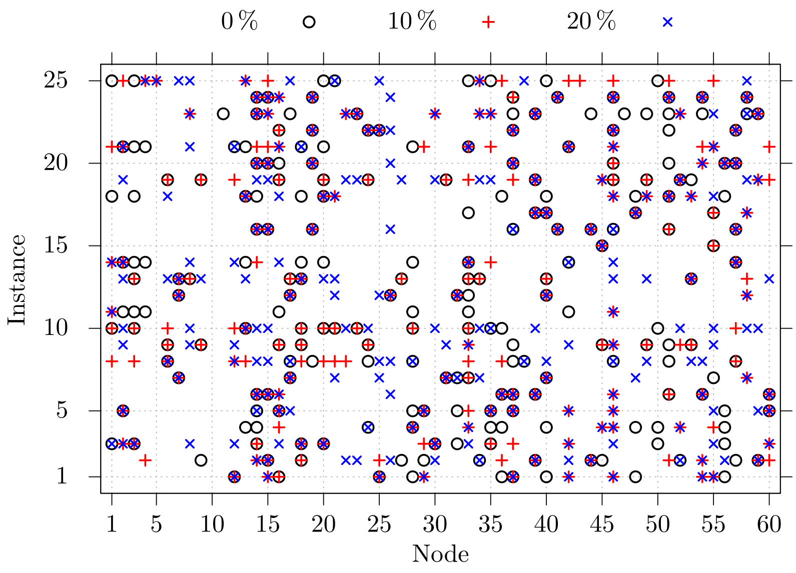


Fig. 6. Isolated nodes in the optimal solution obtained by solving the proposed model for each instance when adding 0%, 10%, and 20% extra arcs.

We solve these 25 instances under three different settings that are constructed by adding 0% or 10% or 20% extra arcs with positive weights to the original network. Note that the network with 0% extra arcs is precisely the original network, and it is a subset of the network with 10% extra arcs. Similarly, the network with 10% extra arcs is a subset of the network with 20% extra arcs.

Fig. 5 illustrates which nodes are infected under each setting for each instance in the optimal solution produced by solving the model. Fig. 6 illustrates which nodes are isolated under each setting for each instance in the optimal solution produced by solving the model. The summary of statistics of these two figures are given in Table 1 where the column 'Ins' shows the instance index/number, columns with label '#Inf' shows

Table 1
Statistics of the optimal solution obtained by solving the proposed model for each instance when adding 0%, 10%, and 20% extra arcs.

Ins	0%		10%		20%		Common infected nodes			Common isolated nodes		
	#Inf	#Iso	#Inf	#Iso	#Inf	#Iso	(0,10)	(0,20)	(10,20)	(0,10)	(0,20)	(10,20)
1	13	9	11	10	11	9	0.46	0.46	1.00	0.44	0.33	0.90
2	14	12	18	12	18	14	0.50	0.57	0.56	0.42	0.50	0.50
3	15	10	16	11	15	14	0.47	0.47	0.75	0.60	0.50	0.64
4	13	10	8	7	8	6	0.31	0.31	1.00	0.10	0.20	0.71
5	14	10	12	9	13	12	0.64	0.57	0.92	0.60	0.70	0.89
6	11	8	12	10	12	10	1.00	0.91	0.92	1.00	0.88	0.90
7	8	7	9	6	11	10	1.00	0.88	0.89	0.71	0.71	0.83
8	12	10	14	12	18	14	0.33	0.67	0.29	0.20	0.50	0.17
9	12	10	13	11	11	8	0.83	0.33	0.46	0.80	0.00	0.18
10	12	12	13	12	17	15	0.75	0.25	0.15	0.75	0.17	0.08
11	12	7	4	2	4	2	0.17	0.17	1.00	0.00	0.00	1.00
12	8	6	8	6	11	8	0.88	0.88	1.00	0.83	0.83	1.00
13	13	10	14	11	15	12	1.00	0.38	0.36	1.00	0.30	0.27
14	12	10	9	6	11	9	0.42	0.50	0.67	0.30	0.40	0.67
15	2	2	2	2	2	1	1.00	1.00	1.00	1.00	0.50	0.50
16	11	9	12	7	12	9	1.00	0.91	0.92	0.78	0.89	0.86
17	6	5	6	5	6	4	0.83	0.83	1.00	0.80	0.60	0.80
18	16	11	10	7	10	10	0.31	0.31	1.00	0.27	0.27	1.00
19	12	11	17	16	17	14	0.92	0.25	0.29	0.91	0.18	0.25
20	11	9	11	8	12	8	0.91	0.91	1.00	0.78	0.67	0.88
21	12	10	15	13	10	9	0.42	0.50	0.53	0.30	0.50	0.46
22	9	8	9	7	10	7	0.89	0.89	1.00	0.88	0.75	0.86
23	16	13	13	11	13	13	0.38	0.38	1.00	0.31	0.38	1.00
24	10	9	11	10	12	10	1.00	1.00	1.00	1.00	0.89	0.90
25	12	8	15	12	16	11	0.25	0.25	0.40	0.00	0.13	0.33
Avg	**11.44**	**9.04**	**11.28**	**8.92**	**11.80**	**9.56**	**0.67**	**0.58**	**0.76**	**0.59**	**0.47**	**0.66**

the number of infected nodes in the generated optimal solution, columns with label '#Iso' shows the number of isolated nodes in the generated optimal solution, and finally columns with label '(a,b)' show the ratio of the number of common infected/isolated nodes in the optimal solutions generated for the networks with a% and b% extra arcs to the number of infected/isolated nodes in the optimal solution generated for the network with a% extra arcs.

We observe that the number of infected and/or isolated nodes does not change much as we increase the number of arcs. This can be justified since more arcs imply that more nodes may become infected at early time periods in the planing horizon. However, since the cap for the number of sick nodes that can be isolated is proportional to the total number of sick nodes, we were able to find a solution that does not change the number of infected and/or isolated nodes significantly on average. Observe too that, on average, more than 50% of isolated or infected nodes have remained the same in the optimal solution by increasing the number of arcs. Specifically, by comparing the optimal solutions produced by solving the networks with 0% and 10% extra arcs, we see that the ratio of common infected nodes is 0.67 and the ratio of common isolated nodes is 0.59 on average. These ratios for the networks with 0% and 20% extra arcs are 0.58 and 0.47, respectively. Also, for the networks with 10% and 20% extra arcs, the ratios are 0.76 and 0.66, respectively.

7.2. Analyzing the overall performance of the model

In this section, we analyze the overall performance of the formulation and enhancements techniques. So, almost all our experiments are conducted under four different settings in this section:

- **E0:** No potential enhancements are employed.
- **E1:** All variable fixing techniques are employed.
- **E2:** All variable fixing, and moderating big-M coefficients techniques are employed.
- **E3:** All variable fixing, moderating big-M coefficients, and symmetry breaking techniques are employed.

Table 2
Overall performance of the model under setting E0 for a single network with $N = 60$.

Ins	$\alpha = 0.5$						$\alpha = 0.75$					
	T (s)	%Gap	OV	#Inf	#D	#Iso	T (s)	%Gap	OV	#Inf	#D	#Iso
1	46	0	21	21	0	14	45	0	13	13	0	9
2	1800	4	28	28	0	25	85	0	14	14	0	12
3	63	0	23	23	0	15	58	0	15	15	0	10
4	122	0	22	22	0	20	68	0	13	13	0	10
5	575	0	25	25	0	19	73	0	14	14	0	10
6	98	0	22	22	0	18	43	0	11	11	0	8
7	164	0	25	25	0	22	30	0	8	8	0	7
8	89	0	21	21	0	18	44	0	12	12	0	10
9	97	0	24	24	0	19	46	0	12	12	0	10
10	1800	79	47	47	0	36	48	0	12	12	0	12
11	334	0	28	28	0	19	55	0	12	12	0	7
12	595	0	23	23	0	14	20	0	8	8	0	6
13	91	0	27	27	0	22	66	0	13	13	0	10
14	113	0	21	21	0	18	43	0	12	12	0	10
15	13	0	2	2	0	2	11	0	2	2	0	2
16	129	0	25	25	0	19	40	0	11	11	0	9
17	86	0	25	25	0	19	16	0	6	6	0	5
18	365	0	22	22	0	18	415	0	16	16	0	11
19	124	0	22	22	0	17	65	0	12	12	0	11
20	88	0	25	25	0	16	45	0	11	11	0	9
21	98	0	23	23	0	20	47	0	12	12	0	10
22	62	0	22	22	0	16	17	0	9	9	0	8
23	105	0	27	27	0	22	121	0	16	16	0	13
24	260	0	23	23	0	15	38	0	10	10	0	9
25	534	0	22	22	0	19	46	0	12	12	0	8
Avg	**314.0**	**3.3**	**23.8**	**23.8**	**0.0**	**18.5**	**63.5**	**0.0**	**11.4**	**11.4**	**0.0**	**9.0**

To compare the performance of the proposed formulation under these four different settings, we have used *performance profiling*. A performance profile is a graph with along the horizontal axis the ratio of the run time (or optimality gap) of an instance to the minimum run time (or minimum optimality gap) for that instance among all methods and along the vertical axis the fraction of instances that achieved a ratio that is less than or equal to the ratio on the horizontal axis [24]. This implies that values in the upper left-hand corner of the graph indicate the best performance. To deal with the value of zero in the run time or the optimality gap, we have added a small positive value, i.e. 0.01, to all numbers while drawing performance profiles in our computational study.

7.2.1. A single network

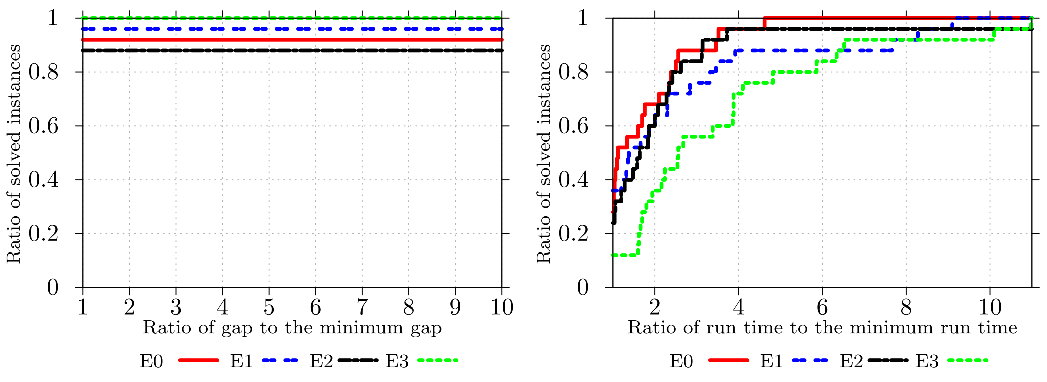
In this section, only two networks are created, one has $N = 60$ and the other has $N = 70$. In total, we generate 50 instances, 25 instances for the network with $N = 60$ and 25 instances for the network with $N = 70$, by just randomly selecting the nodes that are supposed to initially become infected, i.e. their corresponding $\textsc{Data}_i = 160$. The instances of the network with $N = 60$ are the same as the ones used in Section 7.1.

The performance of the proposed formulation with no enhancement (E0), for $N = 60$ and $N = 70$, are given in Tables 2 and 3, respectively. In these tables, 'T(sec.)' shows the run time in seconds; '%Gap' shows the optimality gap obtained; OV shows the best objective value obtained, and finally '#D' shows the number of dead nodes.

Based on the information given in Tables 2 and 3, we observe that instances with $\alpha = 0.75$ are significantly easier than instances with $\alpha = 0.5$. More precisely, when $N = 60$ and $\alpha = 0.5$, 23 out of 25 instances are solved to optimality within the time limit. However, when $N = 60$ and $\alpha = 0.75$, all instances are solved to optimality within the time limit. Similarly, when $N = 70$ and $\alpha = 0.5$, 15 out of 25 instances are solved to optimality within the time limit. However, when $N = 70$ and $\alpha = 0.75$, 20 out of 25 instances are solved to optimality within the time limit. Also, we observe that instances with $N = 70$ are significantly harder

Table 3
Overall performance of the model under setting E0 for a single network with $N = 70$.

Ins	$\alpha = 0.5$						$\alpha = 0.75$					
	T (s)	%Gap	OV	#Inf	#D	#Iso	T (s)	%Gap	OV	#Inf	#D	#Iso
1	1800	42	31	31	0	22	71	0	13	13	0	12
2	1800	74	42	42	0	39	47	0	12	12	0	10
3	1800	29	35	35	0	33	54	0	15	15	0	13
4	1318	0	30	30	0	18	257	0	12	12	0	10
5	805	0	25	25	0	22	51	0	11	11	0	9
6	1800	45	22	22	0	16	20	0	7	7	0	4
7	618	0	23	23	0	19	1800	45	20	20	0	17
8	1800	41	32	32	0	24	93	0	14	14	0	10
9	463	0	26	26	0	21	78	0	17	17	0	13
10	1800	6	32	32	0	26	19	0	7	7	0	4
11	1800	55	44	44	0	38	73	0	15	15	0	12
12	1328	0	26	26	0	14	1800	8	13	13	0	9
13	640	0	22	22	0	15	81	0	16	16	0	16
14	576	0	31	31	0	24	60	0	12	12	0	11
15	555	0	32	32	0	28	169	0	15	15	0	13
16	1394	0	27	27	0	24	80	0	16	16	0	13
17	248	0	29	29	0	24	1800	25	20	20	0	16
18	1800	61	46	46	0	35	57	0	14	14	0	12
19	1800	38	39	39	0	32	59	0	11	11	0	11
20	229	0	26	26	0	24	77	0	12	12	0	11
21	535	0	29	29	0	29	544	0	15	15	0	14
22	1800	3	32	32	0	28	1800	50	16	16	0	11
23	1075	0	26	26	0	22	1800	36	14	14	0	9
24	1381	0	26	26	0	21	79	0	16	16	0	13
25	658	0	23	23	0	21	466	0	16	16	0	13
Avg	**1193.0**	**15.7**	**30.2**	**30.2**	**0.0**	**24.8**	**457.4**	**6.5**	**14.0**	**14.0**	**0.0**	**11.4**

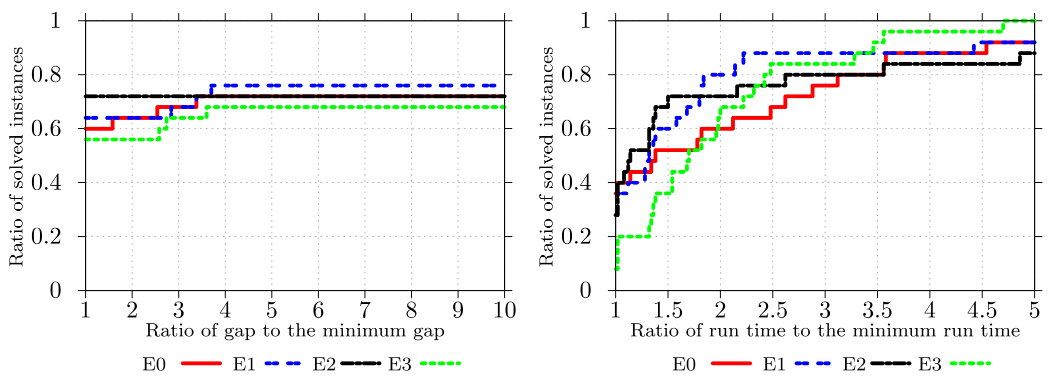


(a) Optimality gap (b) Run time

Fig. 7. Performance profile of the model under different settings for a single network with $N = 60$ and $\alpha = 0.5$.

than instances with $N = 60$. For example, when $\alpha = 0.5$, on average, the optimality gap of instances with $N = 70$ is more than the optimality gap of instances with $N = 60$ by a factor of around 5.

The performance profile of E0, E1, E2, and E3 when $\alpha = 0.5$ for $N = 60$ and $N = 70$ are given in Figs. 7 and 8, respectively. Note that in the performance profile related to the optimality gap, the run time does not matter and only the optimality gap produced within the time limit is important. On the contrary, in the performance profile related to the run time, only solution time obtained within the imposed time limit is important. Observe that E0 and E2 are competitive in terms of the run time for the network with $N = 60$ and perform better than the others. However, in terms of the optimality gap, it seems that E3 performs better than the others. For the network with $N = 70$, we see that E1 and E2 are competitive for both the run time and the optimality gap. Note that the lines in Fig. 7a are flat since we have cut off the horizontal axis from the value of 10. However, if we do not do that then all the lines will eventually increase until they



(a) Optimality gap (b) Run time

Fig. 8. Performance profile of the model under different settings for a single network with $N = 70$ and $\alpha = 0.5$.

reach to the value of 1 on the vertical axis. In that case, the maximum value on the horizontal axis can be 10,001 since, in the worst case scenario, one method may have reached to the optimality gap of 0% and one other method may have the optimality gap of 100%. Hence, since for generating the graph, we revise the optimality gaps by adding 0.01 to them, in the worst case scenario, we have that $\frac{100.01}{0.01} = 10{,}001$.

The performance profile of E0, E1, E2, and E3 when $\alpha = 0.75$ for $N = 60$ and $N = 70$ are given in Figs. 9 and 10, respectively. Observe that for the network with $N = 60$, E2 has the best performance. For the network with $N = 70$, in terms of the optimality gap E3 is better, but in terms of the run time E1 is dominating.

So, in conclusion, E1 and E2 seem to be competitive and they are overall the best choices. We observe that E3 has some advantageous in some cases, but overall it is not as good as E1 and E2. This is probably because the symmetry breaking is not effective. To show this, we combine the instances with $N = 60$, $N = 70$, $\alpha = 0.5$ and $\alpha = 0.75$ together. Let E4 denote the performance of the model when all enhancements but symmetry breaking are disactivated. The performance profile of E0 and E4 on the combined instances are shown in Fig. 11. We see that E0 dominates E4 in terms of both the optimality gap and the run time.

7.2.2. Different networks

In this section, we randomly generate 25 networks with $N = 60$ and 25 networks with $N = 70$. Again, for each of these 50 instances, we randomly select the nodes that are supposed to initially become infected. In this section, we assume that $\alpha = 0.75$.

The performance of the proposed formulation with no enhancement (E0) on the generated instances is shown in Table 4. Not surprisingly, we observe that on average, the run time of instances with $N = 70$ is more than the run time of the instances with $N = 60$ by a factor of around 3.

The performance profile of E0, E1, E2, and E3 for $N = 60$ and $N = 70$ are given in Figs. 12 and 13, respectively. Again, we observe that E1 and E2 are competitive. Specifically, for the instances with $N = 60$, E1 is slightly better but for the instances with $N = 70$, E2 is slightly better.

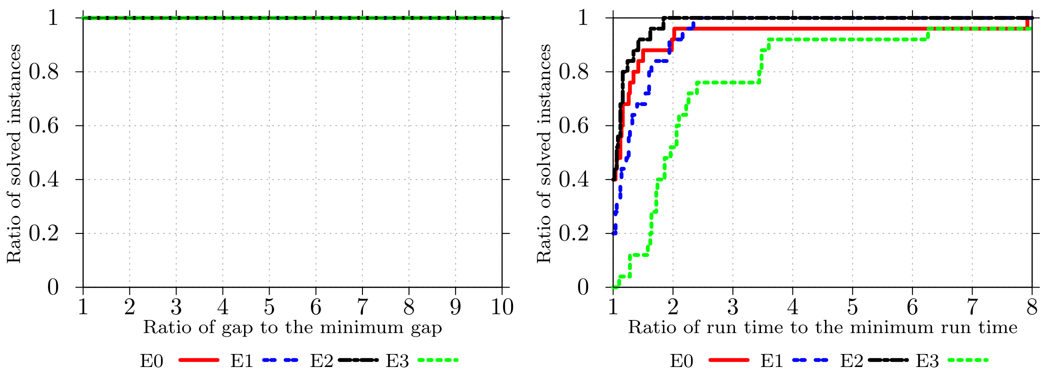
7.2.3. Large instances

In this section, 10 large different networks with $N = 200$ are generated. For each of these 10 instances, we randomly select $\lfloor 200 \times 0.04 \rfloor = 8$ nodes to become initially infected. In this section, we again assume that $\alpha = 0.75$.

The performance of the proposed formulation with no enhancement (E0) on the generated instances is shown in Table 5. Not surprisingly, we observe that the optimality gap is no less than 80% for each instance since the size of the test instances is large.

Table 4
Overall performance of the model under setting E0 with $\alpha = 0.75$.

Ins	$N = 60$						$N = 70$					
	T (s)	%Gap	OV	#Inf	#D	#Iso	T (s)	%Gap	OV	#Inf	#D	#Iso
1	37	0	11	11	0	9	135	0	13	13	0	8
2	45	0	11	11	0	10	155	0	19	19	0	16
3	35	0	10	10	0	10	68	0	16	16	0	13
4	77	0	15	15	0	15	137	0	21	21	0	19
5	74	0	16	16	0	11	263	0	13	13	0	11
6	122	0	16	16	0	13	1800	23	13	13	0	10
7	63	0	13	13	0	11	101	0	19	19	0	16
8	29	0	10	10	0	7	13	0	5	5	0	4
9	16	0	9	9	0	8	53	0	17	17	0	14
10	1800	8	12	12	0	11	105	0	18	18	0	15
11	14	0	8	8	0	4	22	0	9	9	0	7
12	168	0	14	14	0	11	1800	48	21	21	0	20
13	54	0	15	15	0	15	149	0	21	21	0	20
14	46	0	10	10	0	8	93	0	16	16	0	16
15	30	0	11	11	0	8	1800	80	44	44	0	44
16	59	0	13	13	0	12	145	0	12	12	0	12
17	36	0	11	11	0	10	18	0	4	4	0	3
18	70	0	14	14	0	13	254	0	13	13	0	12
19	41	0	12	12	0	9	77	0	16	16	0	15
20	30	0	12	12	0	10	193	0	18	18	0	18
21	274	0	19	19	0	14	92	0	17	17	0	13
22	209	0	14	14	0	9	967	0	17	17	0	14
23	62	0	15	15	0	14	212	0	12	12	0	10
24	15	0	9	9	0	8	1800	68	34	34	0	32
25	18	0	6	6	0	4	65	0	14	14	0	11
Avg	**137.0**	**0.3**	**12.2**	**12.2**	**0.0**	**10.2**	**420.7**	**8.7**	**16.9**	**16.9**	**0.0**	**14.9**



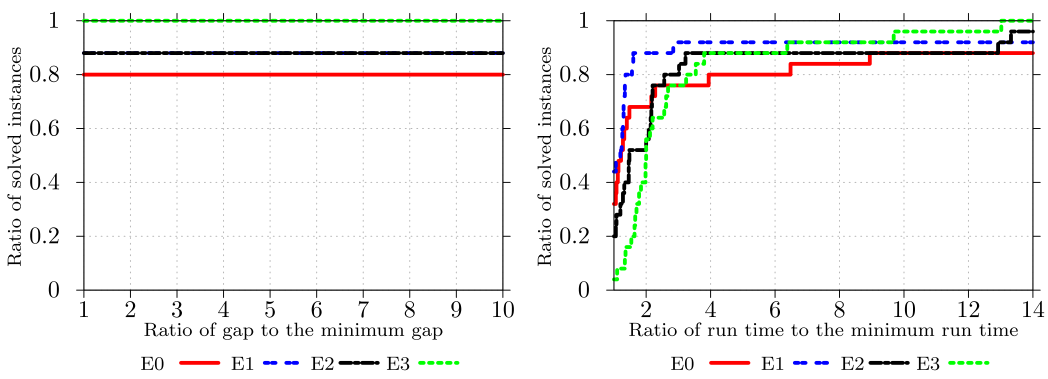
(a) Optimality gap (b) Run time

Fig. 9. Performance profile of the model under different settings for a single network with $N = 60$ and $\alpha = 0.75$.

The performance profile of E0, E1, E2, and E3 for the instances with $N = 200$ are given in Fig. 14. Note that since no instance is solved to optimality within the time limit, only the performance profile of the optimality gap is given. Observe that E1 is performing better than others, but its superiority is almost negligible.

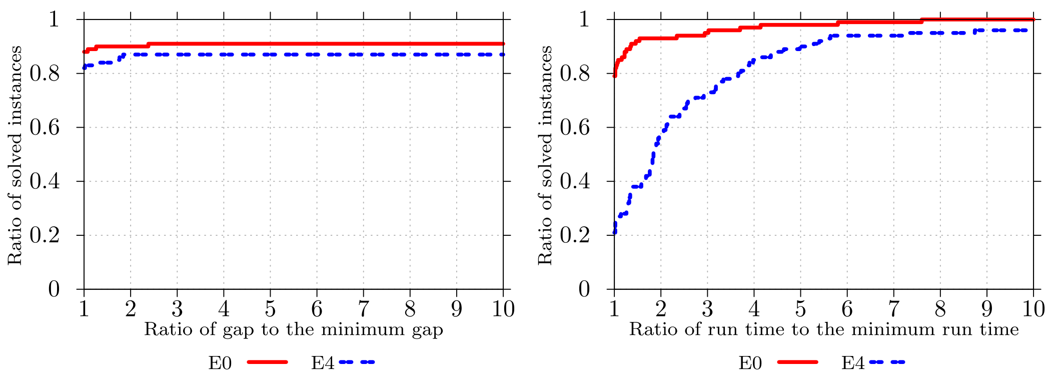
Overall, we observe that by the proposed formulation and its enhancement techniques, solving instances with 200 nodes (or larger) seems to be quite challenging (if not impossible). This is mainly because the optimality gap is around 80% for each instance on average (even under E1). So, the main question is now how the solution time can be improved even further?

Let $\mathcal{F}$ be the formulation presented in Section 4. Below, we give two alternative integer linear programming formulations and conduct an experiment where they reach better optimality gaps for the same time limit. We define $\mathcal{F}_1$ as $\mathcal{F}$ after removing Constraints (11), (12), (14), (15), (17), and (18). Also, we define $\mathcal{F}_2$ as



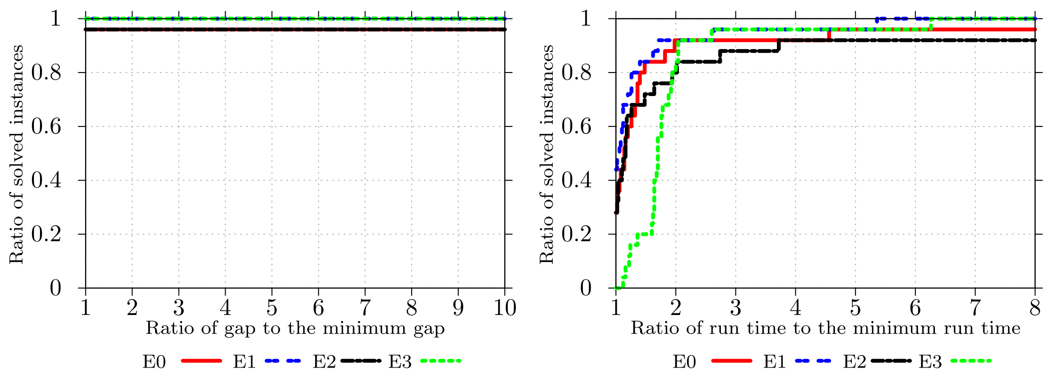
(a) Optimality gap (b) Run time

Fig. 10. Performance profile of the model under different settings for a single network with $N = 70$ and $\alpha = 0.75$.



(a) Optimality gap (b) Run time

Fig. 11. Performance profile of the model under settings E0 and E4 for a single network with $N \in \{60, 70\}$ and $\alpha \in \{0.5, 0.75\}$.

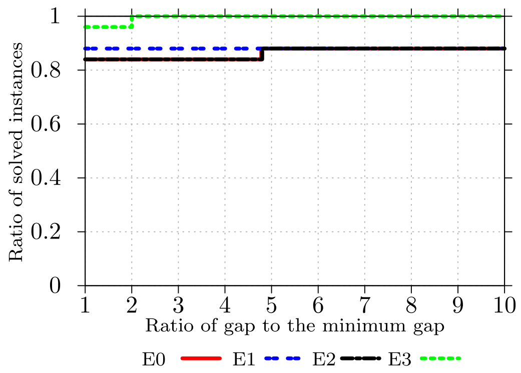


(a) Optimality gap (b) Run time

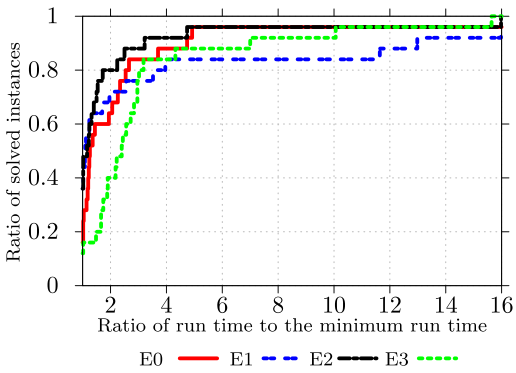
Fig. 12. Performance profile of the model under different settings for different networks with $N = 60$ and $\alpha = 0.75$.

$\mathcal{F}_1$ after replacing Constraints (13) and (16) by

$$\bar{y}_{jtk} \geq y^{(2)}_{j,t-t_w-k} + y^{(3)}_{j,t-t_w-k} - \sum_{t'=1}^{t} x_{jt'}$$



(a) Optimality gap



(b) Run time

Fig. 13. Performance profile of the model under different settings for different networks with $N = 70$ and $\alpha = 0.75$.

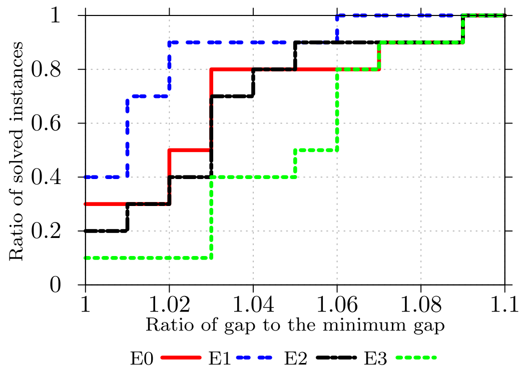


Fig. 14. Performance profile of the model under different settings for instances with $N = 200$ and $\alpha = 0.75$.

Table 5
Overall performance of the model under setting E0 with $N = 200$ and $\alpha = 0.75$.

Ins	T (s)	%Gap	OV	#Inf	#D	#Iso
1	1800	80	158	158	0	154
2	1800	81	176	176	0	176
3	1800	81	160	160	0	159
4	1800	85	200	200	0	148
5	1800	87	224	199	**1**	8
6	1800	81	166	166	0	160
7	1800	82	200	200	0	193
8	1800	82	157	157	0	125
9	1800	83	174	174	0	173
10	1800	84	173	173	0	171
Avg	**1800.0**	**82.5**	**178.8**	**176.3**	**0.1**	**146.7**

for each $j \in \mathcal{N}$, $t \in \{t_w + k + 1, \dots, T\}$ and $k \in \{1, \dots, t_s\}$ where $\bar{y}_{jtk} \in \{0, 1\}$ is a binary variable. Note that by doing so, the previous definition of c_{jt} (in Section 4) can be simplified to

$$c_{jt} = \sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k \bar{y}_{jtk}$$

for each $j \in \mathcal{N}$ and $t \in \mathcal{T}$.

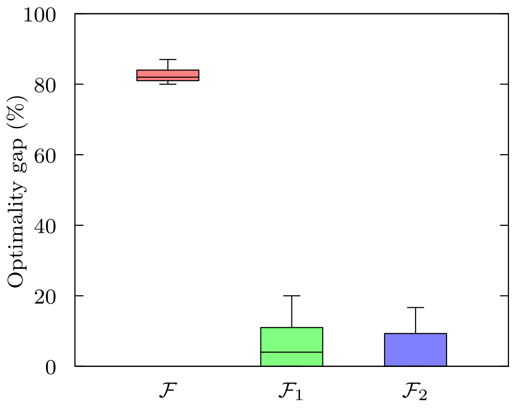


Fig. 15. Optimality gap of different models under setting E1 for instances with $N = 200$ and $\alpha = 0.75$.

Proposition 1. *$\mathcal{F}_1$ and $\mathcal{F}_2$ are correct formulations.*

Proof. We first prove the correctness of $\mathcal{F}_1$. Note that Constraints (11), (12), (14), (15), (17), and (18) are generated as a result of linearizing Constraint (8), and they all simply impose upper bounds for their corresponding variables. For each $i \in \mathcal{N}$ and $t \in \mathcal{T} \setminus \{1\}$, Constraint (8) captures the value of z_{it} which is the total risk of the network for node i at time period t. Note that the objective function of the proposed integer linear program attempts to minimize the (positive) weighted summation of the total number of infected nodes. This implies that the proposed integer linear program tends to minimizes z_{it} because this variable indirectly shows whether node i is infected or not at time period t. So, Constraints (11), (12), (14), (15), (17), and (18) are all unnecessary for an optimal solution.

We now prove the correctness of $\mathcal{F}_2$. Again note that Constraints (13) and (16) are generated as a result of linearizing Constraint (8). Specifically, they are defined to linearize c_{jt} for $j \in \mathcal{N}$ and $t \in \mathcal{T}$:

$$c_{jt} = (1 - \sum_{t'=1}^{t} x_{jt'})(\sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k y^{(2)}_{j,t-t_w-k} + \sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k y^{(3)}_{j,t-t_w-k}),$$

which is equivalent to:

$$c_{jt} = (1 - \sum_{t'=1}^{t} x_{jt'}) \sum_{k=1}^{\min\{t_s, t-t_w-1\}} d_k (y^{(2)}_{j,t-t_w-k} + y^{(3)}_{j,t-t_w-k}).$$

So, we could simply linearize $(1 - \sum_{t'=1}^{t} x_{jt'})(y^{(2)}_{j,t-t_w-k} + y^{(3)}_{j,t-t_w-k})$ rather than $(1 - \sum_{t'=1}^{t} x_{jt'}) y^{(2)}_{j,t-t_w-k}$ and $(1 - \sum_{t'=1}^{t} x_{jt'}) y^{(3)}_{j,t-t_w-k}$ individually. In other words, let $\bar{y}_{jtk} := (1 - \sum_{t'=1}^{t} x_{jt'})(y^{(2)}_{j,t-t_w-k} + y^{(3)}_{j,t-t_w-k})$. By Constraint (4), we know that $\sum_{t\in\mathcal{T}} y^{(2)}_{jt} + y^{(3)}_{jt} \leq 1$. Also, note that Constraints (11), (12), (14), (15), (17), and (18) can be removed. So, the result immediately follows. □

To show the strength of the new formulations, we compare the optimality gap obtained by $\mathcal{F}$, $\mathcal{F}_1$, and $\mathcal{F}_2$ under setting E1 for instances with $N = 200$ and $\alpha = 0.75$ within 1800 s in Fig. 15. Observe that the optimality gap is reduced from around 80% on average for $\mathcal{F}$ to around 6.3% on average for $\mathcal{F}_1$ and to around 4.7% on average for $\mathcal{F}_2$. In fact more than half of the instances are solved to optimality for $\mathcal{F}_2$ within the imposed time limit.

So, observe that even (many) large instances with 200 nodes can be solved to optimality (within half an hour) using the new proposed integer linear programs. It is worth mentioning that a recent study conducted by Nandi and Medal [3] also solves similar size instances to optimality within two hours. Our proposed formulation is significantly different from the one given by Nandi and Medal [3]. In our case, the removal is produced on nodes instead of links. We also take into account many practical aspects of the spread of influenza virus infections that none of the previous studies considered them. However, this computational study shows that even with such differences, similar size instances can still be solved.

8. Final remarks

We studied the spread of influenza virus infections on (dynamic) networks of people. We presented an integer linear programming formulation to minimize the spread of infections by removing nodes which can be interpreted as isolating infected nodes (or vaccinating the healthy and susceptible nodes). The novelty of the formulation comes from the fact that it incorporates many practical aspects of the spread of an influenza virus infection. Moreover, several potential enhancement techniques for improving the performance of the formulation were introduced and tested via a computational study. Specifically, it was shown that employing variable fixing and moderating big-M coefficients techniques can reduce the run time and/or the optimality gap. It is worth mentioning that the proposed formulation is a deterministic formulation. However, some of the parameters of the formulation are naturally uncertain, for example t_w and t_s, in practice. Hence incorporating some levels of uncertainty for such parameters require further research. In other words, applying stochastic or robust optimization techniques can be a future research direction for this study. Also, our proposed formulation can be used for either isolation or vaccination and not both. So, modifying the formulation to include both of them requires further research. Finally, developing effective custom-built heuristic/exact solvers for the problem is another future research direction.

Acknowledgments

We are grateful for the time and effort that the anonymous reviewers spent on the manuscript. Their detailed and constructive comments (especially about linearization) resulted in a stronger, more readable, and more comprehensive paper.

References

[1] L. Sun, G.W. DePuy, G.W. Evans, Multi-objective optimization models for patient allocation during a pandemic influenza outbreak, Comput. Oper. Res. 51 (2014) 350–359.
[2] CDC, 2009. Link. https://www.cdc.gov/h1n1flu/estimates_2009_h1n1.htm (Last accessed: 14-07-17).
[3] A.K. Nandi, H.R. Medal, Methods for removing links in a network to minimize the spread of infections, Comput. Oper. Res. 69 (2016) 10–24.
[4] E.A. Enns, J.J. Mounzer, M.L. Brandeau, Optimal link removal for epidemic mitigation: A two-way partitioning approach, Math. Biosci. 235 (2) (2012) 138–147.
[5] M. Kimura, K. Saito, H. Motoda, Blocking links to minimize contamination spread in a social network, ACM Trans. Knowl. Discov. Data 3 (2) (2009) 9:1–9:23.
[6] J. Marcelino, M. Kaiser, Reducing influenza spreading over the airline network, PLOS Curr. Influenza (2009).
[7] J. Marcelino, M. Kaiser, Critical paths in a metapopulation model of H1N1: Efficiently delaying influenza spreading through flight cancellation, PLOS Curr. Influenza (2012).
[8] A.K. Nandi, H.R. Medal, S. Vadlamani, Interdicting attack graphs to protect organizations from cyber attacks: A bi-level defender-attacker model, Comput. Oper. Res. 75 (2016) 118–131.
[9] B. Addis, M.D. Summa, A. Grosso, Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth, Discrete Appl. Math. 161 (16–17) (2013) 2349–2360.
[10] A. Arulselvan, C.W. Commander, L. Elefteriadou, P.M. Pardalos, Detecting critical nodes in sparse graphs, Comput. Oper. Res. 36 (7) (2009) 2193–2200.
[11] M. Di Summa, A. Grosso, M. Locatelli, Branch and cut algorithms for detecting critical nodes in undirected graphs, Comput. Optim. Appl. 53 (3) (2012) 649–680.

[12] S. Shen, J.C. Smith, R. Goli, Exact interdiction models and algorithms for disconnecting networks via node deletions, Discrete Optim. 9 (3) (2012) 172–188.
[13] M. Ventresca, D. Aleman, A derandomized approximation algorithm for the critical node detection problem, Comput. Oper. Res. 43 (2014) 261–270.
[14] A. Veremyev, V. Boginski, E.L. Pasiliao, Exact identification of critical nodes in sparse networks via new compact formulations, Optim. Lett. 8 (4) (2014) 1245–1259.
[15] A. Veremyev, O.A. Prokopyev, E.L. Pasiliao, An integer programming framework for critical elements detection in graphs, J. Comb. Optim. 28 (1) (2014) 233–273.
[16] J. He, H. Liang, H. Yuan, Controlling infection by blocking nodes and links simultaneously, in: N. Chen, E. Elkind, E. Koutsoupias (Eds.), Internet and Network Economics: 7th International Workshop, WINE 2011, Singapore, December 11–14, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 206–217.
[17] M. Hemmati, J. Cole Smith, M.T. Thai, A cutting-plane algorithm for solving a weighted influence interdiction problem, Comput. Optim. Appl. 57 (1) (2014) 71–104.
[18] N.M. Ferguson, D.A. Cummings, S. Cauchemez, C. Fraser, S. Riley, M. Aronrag, et al., Strategies for containing an emerging influenza pandemic in southeast asia, Nature 437 (7056) (2005) 209–214.
[19] B.J. Cowling, L. Jin, E.H. Lau, Q. Liao, P. Wu, H. Jiang, T.K. Tsang, J. Zheng, V.J. Fang, Z. Chang, et al., Comparative epidemiology of human infections with avian influenza A H7N9 and H5N1 viruses in china: a population-based study of laboratory-confirmed cases, Lancet 382 (9887) (2013) 129–137.
[20] N.M. Ferguson, D.A. Cummings, C. Fraser, J.C. Cajka, P.C. Cooley, D.S. Burke, Strategies for mitigating an influenza pandemic, Nature 442 (7101) (2006) 448–452.
[21] WHO, 2008. Link. http://www.who.int/diseasecontrol_emergencies/HSE_EPR_DCE_2008_3rweb.pdf (Last accessed: 14-07-17).
[22] F. Margot, Symmetry in integer linear programming, in: M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, L.A. Wolsey (Eds.), 50 Years of Integer Programming 1958-2008, Springer Berlin Heidelberg, 2010, pp. 647–686.
[23] H.D. Sherali, J.C. Smith, Improving discrete model representations via symmetry considerations, Manage. Sci. 47 (10) (2001) 1396–1407.
[24] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213.