A hybridised variable neighbourhood tabu search heuristic to increase security in a utility network

Reference:

Janssens Jochen, Talarico Luca, Sörensen Kenneth.- A hybridised variable neighbourhood tabu search heuristic to increase security in a utility network
Reliability engineering and system safety - ISSN 0951-8320 - 145(2016), p. 221-230
Full text (Publisher's DOI): http://dx.doi.org/doi:10.1016/J.RESS.2015.08.008
To cite this reference: http://hdl.handle.net/10067/1285940151162165141

A hybridised variable neighborhood tabu search heuristic to increase security in a utility network

Jochen Janssens[a,*], Luca Talarico[a], Kenneth Sörensen[a]

[a] *Universiteit Antwerpen, Faculteit Toegepaste Economische Wetenschappen, Prinsstraat 13, 2000 Antwerp, Belgium*

Abstract

We propose a decision model aimed at increasing security in a utility network (e.g., electricity, gas, water or communication network). The network is modeled as a graph, the edges of which are unreliable. We assume that all edges (e.g., pipes, cables) have a certain, not necessarily equal, probability of failure, which can be reduced by selecting edge-specific security strategies. We develop a mathematical programming model and a metaheuristic approach, that uses a greedy random adaptive search procedure to find an initial solution and uses tabu search hybridised with iterated local search and a variable neighbourhood descend heuristic to improve this solution. The main goal is to reduce the risk of service failure between an origin and a destination node by selecting the right combination of security measures for each network edge given a limited security budget.

Keywords: network security, metaheuristics, GRASP, ILS, tabu search, VND.

1. Introduction

In modern day society, utility networks such as electricity, water, gas, and communication networks are taken for granted. People expect that they function at all times, and are capable of handling all demand placed on them. However, there is a real risk of failure in all types of networks. Those failures might interrupt the service/connection between origin nodes (i.e., the points from which the service or the product is sent to the customer through the network) and destination nodes (i.e., the customer or points to which the product or service is delivered through the network).

Utility networks are critical infrastructures that deserve increased attention to limit the consequences of failures since they may result into serious breakdowns and cascade into dependent systems (Kröger, 2008). For example, the blackout in India in 2012 left over 600 million people without access to energy. This emphasizes the need for adequate protective measures against network breakdowns (Barker et al., 2013; Romero, 2012).

Network breakdowns can have *safety*-related (i.e., unintentional) causes such as natural phenomena (e.g., earthquakes, storms), human errors, or mechanical defects such as in pumps and valves, caused by the regular wear and tear. In addition, network breakdowns can be due to *security*-related causes such as intentional terrorist attacks and/or malicious sabotage. We refer the reader to Reniers et al. (2008) for a further clarification of terms *safety* and *security*. For the sake

[*] Corresponding author
Email address: `jochen.janssens@uantwerpen.be, tel: +32 (0)32654017` (Jochen Janssens)

of simplicity, in the remainder of this work we will only talk about network security, however, the solution method developed in this paper is also applicable to safety related causes.

After 9/11, the protection of utility networks against intentional attacks has received great attention among network providers. Terrorist attacks on utility network are not rare and can cause huge losses in a nation's economy (START, 2014).

Network providers and managers can reduce the risk of a network breakdown, due to failures in one or several network edges, by applying preventive measures to reduce network vulnerabilities. These preventive measures can be, amongst others, fences, camera's or security patrols to increase network security and redundancy or stronger materials for an increase in network safety.

The security budget that can be spent on these security measures, however, is generally limited. The current economic situation has increased the pressure on limiting many budgets and investments in security even further. In this paper, a combination of security measures applied to an edge is called a *security strategy* for that edge.

The problem defined in this paper is how to allocate a budget among edge-specific security strategies in order to minimise the risk of a disconnection between a given origin node and a given destination node. In the remainder, we will say that the utility network is down whenever the origin and the destination node are disconnected from each other.

Since the budget is limited and the security strategies can only be applied locally, i.e., on specific links in the network, the security strategies should be chosen in such a way that the reduction of the risk of the network service being down is as large as possible while keeping the total cost of the security strategies within the budget.

Once we consider realistic cases with a large number of edges and different security strategies (in this paper from 5 up to 20), the problem becomes computationally infeasible to solve in a reasonable amount of time with exact algorithms. Therefore we will explore the use of metaheuristics to support this decision problem.

The paper is organized as follows. In Section 2, we give a brief overview of the state of the art. Section 3 clarifies the problem of selecting the best strategies to increase the security of the whole network. It is described and modelled as an optimisation problem. An illustrative example is also provided. In Section 4, we present a metaheuristic to solve the network security problem. Section 5 presents the instance generator, reports the parameter tuning phase and the results of the computational experiments. Section 6 concludes the paper and presents some ideas for further developments.

2. Literature review

To guarantee the robustness of networks, a lot of research has been conducted in the field of survivable networks (see Kerivin & Mahjoub, 2005; Steiglitz et al., 1969). Most work in this field attempts to improve network robustness by means of redundant paths between nodes. Although significant research has been done to improve best practices in the field of security, few papers have addressed the relationship between risk-related variables and cost-effective network security decisions that impact the objective. Nevertheless, security measure selection problems have received some attention in more recent literature.

The problem of selecting the right security measures given a limited budget is clearly not an easy task. Most security planning models in the literature are qualitative and only few of them rely on quantitative approaches. In case of a pipeline network, the security risk assessment procedure elaborated by Reniers & Dullaert (2012) may be used. After a careful pipeline security risk assessment, the user is in possession of pipeline segment risk data as well as pipeline route risk data. Assuming that the security risk analyst determines a set of available security measures and defence strategies for application to the different pipeline segments and/or for the pipeline routes, a selection of the most effective security measures with respect to the available budget (either for a single pipeline segment or for a pipeline route) can be calculated. If the cost of the security measures is known in advance a mathematical approach can be used to solve the problem of optimal allocation of security resources by solving a knapsack problem with additional constraints. Reniers et al. (2012) explain how this well-known problem in the field of Operations Research can be used in security optimisation problems. A practical application to secure an illustrative pipeline infrastructure used to transport oil is described in Talarico et al. (2015).

In Bistarelli et al. (2007) a method for the identification of the assets, the threats and the vulnerabilities of ICT systems is introduced. Furthermore, a qualitative approach for the selection of security measures to protect an IT infrastructure from external attacks is discussed. In particular, two security models based on defence trees (an extension of attack trees) and preferences over security measures are proposed.

In Viduto et al. (2012) the security of a telecommunication network is analysed from a quantitative point of view. Knowledge of potential risks enables organisations to take decisions on which security measures should be implemented before any potential threat can successfully exploit system vulnerabilities. A security measure selection problem is presented in which both cost and effectiveness of an implemented set of security measures are addressed. A Multi-Objective Tabu Search (MOTS) algorithm is developed to construct a set of non-dominated solutions which can satisfy organisational security needs in a cost-effective manner.

In Sawik (2013) a similar security measure selection problem for an IT infrastructure is formulated as a single- or bi-objective mixed integer programming problem. Given a set of potential threats and a set of available security measures, the decision maker needs to determine which security measure to implement, under a limited budget, to minimize potential losses from successful cyber-attacks and mitigate the impact of disruptions caused by IT security incidents.

The prevention of heavy losses due to cyber-attacks and other information system failures in an IT network is usually associated with continuous investment in different security measures. In Bojanc & Jerman-Blažič (2008) several approaches enabling the assessment of the necessary investments in security technology are addressed from an economical point of view. The paper introduces methods for the identification of risks in ICT systems and proposes a procedure that enables the selection of the optimal level of investments in security measures.

Once security risks have been identified, the potential loss associated with their occurrence, as well as their probability of occurrence must be determined. Probability theory is used extensively in reliability theory and in reliability studies of systems. For an overview, we refer to Bazovsky (2004); Ministry of Defence (UK) (2011); Romeu (2004). Determining both probability of occurrence and potential impact of each risk is done in a process called *risk assessment*. Performing a risk assessment phase allows to take decisions regarding the necessary investment in security controls and systems. In this paper, a preliminary risk assessment phase is assumed to have been previously conducted by experts, and that they defined the probability for an edge to fail together with the costs and benefits of each available security measure. A methodology to conduct risk assessment for safety-related accidents has been proposed in Antonioni et al. (2009). In Reniers et al. (2014) and Reniers et al. (2013), security-related risk and threat assessment within chemical plants is studied.

In Agarwal et al. (2013), the focus is on probabilistic attacks and on multiple simultaneous attacks on telecommunications networks. The probabilities associated to these events are dependent on geographical locations. Vulnerable points within the network are identified by applying geometric tools. Using this approach, it is possible to identify locations which require additional protection efforts.

The work of Reniers & Dullaert (2012) gives a nice overview of risk assessment for security-related attacks on pipeline networks. Our approach represents a next step in the process. It defines a single-objective problem and proposes a quantitative method to select appropriate security measures.

In Reniers et al. (2012), a mathematically sound methodology is developed to map two types of systemic risks in chemical industrial areas. By using their model, conclusions can be drawn about possible prevention measures to lower the systemic risks.

The approach in this paper uses a different objective function which relies on the minimization of the risk of the network to be not accessible between a couple of network nodes instead of the maximization of the effectiveness of the security measures used. Moreover, in our work, since a list of security measures is defined for each edge of the network, the model incorporates not only decisions taken at the level of the network, as done in Reniers & Dullaert (2012), Reniers et al. (2012) and Sawik (2013), but it depends on the choices made at the level of single network edges.

3. Problem description

A utility network can be represented by a graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N}$ represents a set of nodes and $\mathcal{E}$ a set of edges, connecting the nodes. All edges $i \in \mathcal{E}$ have a probability of failing, denoted as p_i. A set of security strategies $\mathcal{S}_i$, is defined for each edge $i \in \mathcal{E}$.

For each security strategy $j \in \mathcal{S}_i$ of edge i, a cost c_{ij} is given, as well as the probability of failure of edge i given that security strategy j has been chosen, p_{ij}. Exactly one security strategy per edge can be selected. Each edge also has a default "do-nothing" strategy $j = 0$ with cost $c_{i0} = 0$.

The model proposed in this paper makes the assumption that only edges are vulnerable to failure and that nodes are not.

In the problem defined in this paper, an origin node o and a destination node d in the network, are gvien. The quality of a solution (e.g., a selection of a security strategy for each edge) is defined as the probability that no path exists between node o and node d. This would make it impossible for a service or good from node o to reach node d (e.g., it would be impossible to make a phone call from node o to node d). The goal is to minimise this probability.

In this paper, since the decision problem is introduced for the first time, the problem is simplified by making the assumption that only one supplier and one customer exist in the network. In a realistic network, the network operator is not only dealing with the question if there is a connection to a customer, but also with the issue of having enough capacity to deliver his product or service in a reliable way. In this paper, since we are dealing with only one customer, an abstraction is made of this last problem. The assumption is made that the load placed on the network by the customer is considerably lower than the load the network can handle. This reduces the problem to a much simpler "path or no path" problem. In future research, this can be extended to a model where multiple source and destination nodes are considered. In addition supplier capacity, customer demand and importance of either of them might be considered.

To calculate the probability that no path exists between a given origin and a given destination node, probability theory is used.

Table 1: Set of security strategies $\mathcal{S}_i$ for edge i

Strategy	Security measures	Cost	Probability
0	-	0	0.6
1	Fences	100	0.5
2	Camera type A	150	0.45
3	Camera type B	200	0.4
4	Fences&Camera type A	250	0.32
5	Fences&Camera type B	300	0.25

3.1. Illustrative Example

A security strategy can be a combination of individual security measures (see e.g., Table 1). A combination of security measures can have a different effectiveness than the sum of the impact of the individual security measures due to interaction effects. In some cases, combinations of single security measures might not be available due to their incompatibility.

To clarify the computation used to determine the total risk that no path exists between nodes o and d, an example in Figure 1 is shown. Edge i in Table 2 corresponds to the edge with probability p_i in this figure.

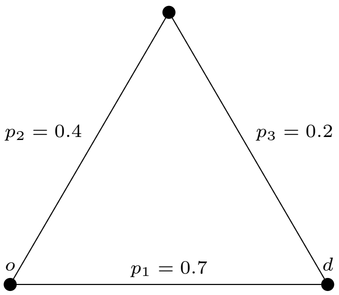


Figure 1: Utility network $\mathcal{G}$ with a source o and a destination d. On each edge the associated probability of an event happening is reported

Given the topology of the network, there are eight possible scenarios, of which three are critical, that remove every path between nodes o and d. To find these critical scenarios, in the approach discussed in this paper, all possible situations are considered and tested. Table 2 contains all scenarios l. The cases in which no path exists in the network between nodes o and d, are the scenarios 4, 5 and 7. All scenarios l are defined as a combination of edges that fail, contained in set $\mathcal{E}_l^F$, and edges that do not fail, contained in set $\mathcal{E}_l^N$. A critical scenario is a scenario where the edges that do fail, if they fail all at once, will disable every path between node o and node d. The set of critical scenarios is defined as $\mathcal{C}$. For all scenarios $l \in \mathcal{C}$ the following property holds: $\mathcal{E}_l^F \cup \mathcal{E}_l^N = \mathcal{E}$. The cardinality of set $\mathcal{C}$ depends on the topology of the network $\mathcal{G}$ and the position of nodes o and d within the network.

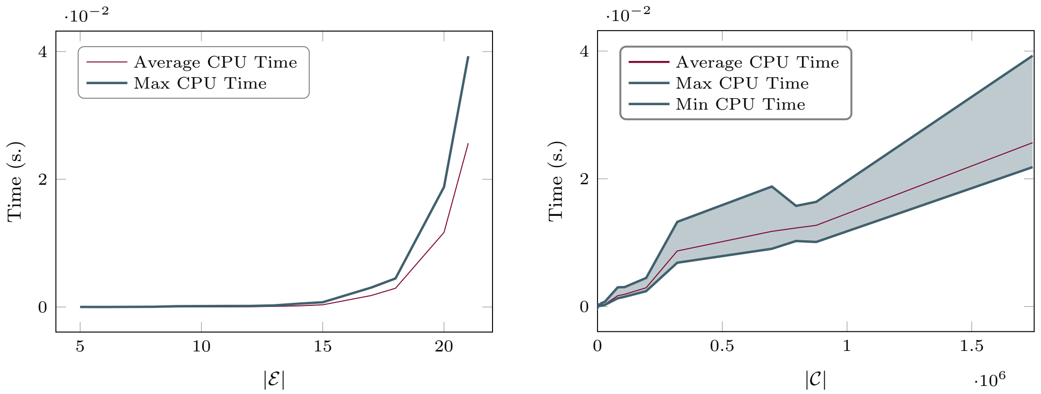
For example, the first line in Table 2 shows a scenario where no edges fail, hence all paths between node o and node d exist. The fourth line, however, shows a scenario where both edges 1 and 2 fail. Therefore no path exists from node o to node d, even if edge 3 does not fail. The

l	$\mathcal{E}_l^F$			$\mathcal{E}_l^N$			R_l
0	-	-	-	1	2	3	0.144
1	1	-	-	2	3	-	0.336
2	2	-	-	1	3	-	0.096
3	3	-	-	1	2	-	0.036
4	1	2	-	3	-	-	0.224
5	1	3	-	2	-	-	0.084
6	2	3		1	-	-	0.024
7	1	2	3	-	-	-	0.056

Table 2: List of scenarios for network $\mathcal{G}$

probability R_l associated to the occurrence of this scenario, is equal to $p_1 \cdot p_2 \cdot (1 - p_3)$ that is $0.4 \cdot 0.7 \cdot (1 - 0.2) = 0.224$. The same reasoning applies to the remaining scenarios reported in Table 2, which contains all possible scenarios for this graph. As we are dealing with every possible scenario, the scenarios are independent events. Therefore, it is possible to compute the total probability of a disconnection between node o and node d as the sum of the occurrence probabilities of all critical scenarios, defined as $\sum_{l \in \mathcal{C}} R_l$. In this example, scenarios 4, 5 and 7, for which the sum is equal to 0.364. Each time a security strategy is changed or updated, the probability for each scenario should be recalculated. And the objective function, total probability for a disconnection between node o and node d, also changes.

As expected, the computation time needed to update the objective function every time a move is applied, grows exponentially if the number of edges increases (see Figure 2(a)). This is due to the number of critical scenarios, which grows significantly depending on the number of edges and on the topology of the network. As shown in Figure 2(b), the relationship between the number of critical scenarios in $\mathcal{C}$ and the computational time is approximately linear.



(a) Number of edges versus CPU time

(b) Critical scenarios versus CPU time

Figure 2: Relationship between the number of edges (a) and critical scenarios (b) and CPU time needed to update the objective function after a move is executed.

Despite the fact that the update of the objective function is expensive in terms of computational time, an estimation of the objective value proves to be difficult. A naive approach to estimate the objective value could be the restriction of the critical scenarios to scenarios with a limited number of edges that fail, as scenarios with a higher number of failing edges have a low probability to occur. This approach, however, can result in unexpected behaviour. Depending on the topology of the network and the values assigned to edges and security strategies, it is possible that the objective value for the network, where all security strategies are the "do-nothing" strategy, is lower than the objective value of the network where one security strategy is selected. This is counterintuitive as the selection of a security strategy should always lower the objective value in comparison to the objective value where all security strategies are the "do-nothing" strategy. A small example is given in Figure 3. All edges have a probability of failure equal to 1%. Edge E is allocated a security strategy, which reduces its probability of failure to 0.7%. Table 3 shows all the critical combinations where at most 3 edges fail at the same time. The first column indicates which edges are failing. The second column contains the risk for that critical combination when the security strategy is not applied. The third column contains the risk when the security strategy is applied on edge E. The bottom line contains the objective value. As the problem is a minimisation problem, this table shows that when the security strategy is applied to edge E, the objective value worsens.

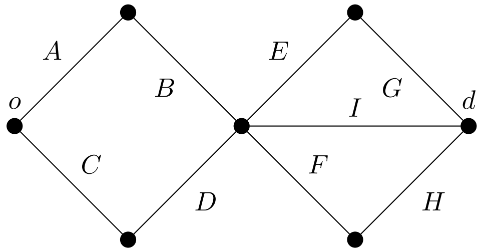


Figure 3: Relationship between the number of edges (a) and critical scenarios (b) and CPU time needed to update the objective function after a move is executed.

3.2. Mathematical model

In order to mathematically state the decision problem associated to the selection of the best set of security strategies to increase the overall network security, the risk of no path being available between node o and destination d has to be defined. For this reason, a set $\mathcal{C}$ is defined.

Let B represents the available security budget and x_{ij} a binary variable, that takes values 1 when the security strategy j on edge i is applied, and 0 otherwise.

Failing edges	no security measure	security measure
A C	0.0000932	0.0000935
A D	0.0000932	0.0000935
B C	0.0000932	0.0000935
B D	0.0000932	0.0000935
A B C	0.0000009	0.0000009
A B D	0.0000009	0.0000009
A C D	0.0000009	0.0000009
A C E	0.0000009	0.0000007
A C F	0.0000009	0.0000009
A C G	0.0000009	0.0000009
A C H	0.0000009	0.0000009
A C I	0.0000009	0.0000009
A D E	0.0000009	0.0000007
A D F	0.0000009	0.0000009
A D G	0.0000009	0.0000009
A D H	0.0000009	0.0000009
A D I	0.0000009	0.0000009
B C D	0.0000009	0.0000009
B C E	0.0000009	0.0000007
B C F	0.0000009	0.0000009
B C G	0.0000009	0.0000009
B C H	0.0000009	0.0000009
B C I	0.0000009	0.0000009
B D E	0.0000009	0.0000007
B D F	0.0000009	0.0000009
B D G	0.0000009	0.0000009
B D H	0.0000009	0.0000009
B D I	0.0000009	0.0000009
E F I	0.0000009	0.0000009
E H I	0.0000009	0.0000009
F G I	0.0000009	0.0000009
G H I	0.0000009	0.0000009
Objective value	0.0003992	0.0003993

Table 3: Result for an instance when estimating the objective value by restricting the critical combinations to those with a maximum of 3 failing edges.

$$\min \quad \sum_{l \in \mathcal{C}} R_l \tag{1}$$

s.t.

$$\sum_{i \in \mathcal{E}} \sum_{j \in \mathcal{S}_i} c_{ij} \cdot x_{ij} \leq B \tag{2}$$

$$p_i = \sum_{j \in \mathcal{S}_i} p_{ij} \cdot x_{ij} \qquad \forall i \in \mathcal{E} \tag{3}$$

$$R_l = \prod_{i \in \mathcal{E}_l^F} p_i \cdot \prod_{k \in \mathcal{E}_l^N} (1 - p_k) \qquad \forall l \in \mathcal{C} \tag{4}$$

$$\sum_{j \in \mathcal{S}_i} x_{ij} = 1 \qquad \forall i \in \mathcal{E} \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in \mathcal{E}, \forall j \in \mathcal{S}_i \tag{6}$$

The objective function (1) minimizes the total risk that no path exists between nodes o and d. The total network risk is given by the sum of risks associated to single scenarios happening, as these scenarios are statistically independent. Constraint (2) ensures that the total cost associated to the selected security strategies does not exceed the predefined security budget B. Equation (3) is used to define the probability p_i associated to a failure of edge i, given a selected security strategy for that edge. Equation (4) defines the risk of a scenario happening which disconnects all paths in the network between nodes o and d. Equation (5) forces the decision process to select at maximum one security strategy to protect each edge i. When $x_{i0} = 1$ no security strategy is applied for edge i. Finally, constraint (6) represents the domain of the decision variable, which ensures that no partial security strategies are allowed.

4. Solution approach

The decision problem of selecting appropriate security strategies given a budget constraint, in order to reduce the risk of disconnection between node o and node d, belongs to the more general category of knapsack problems.

More specifically, since the objective function is not linear, it belongs to the class of non-linear knapsack problems, also known as the non-linear resource allocation problem, which belongs to the category of combinatorial optimization problems (Bretthauer & Shetty, 2002)

The metaheuristic that has been developed in this paper is shown in Algorithm 1. Our variable neighborhood tabu search metaheuristic is composed of three consecutive steps: (1) a greedy random adaptive search procedure (GRASP) is used during the constructive phase (see Section 4.1) to generate an initial solution; (2) a variable neighbourhood descent (VND) is used during the improvement phase (see Section 4.2) to improve the solution generated by the GRASP and; (3) two perturbation heuristics are used during the diversification stage (see Section 4.3) to escape from local optima. In addition a tabu list is used during the whole execution of the heuristic to avoid an exploration of solutions that have been analysed in previous iterations.

The first step of this iterative solution approach consists of running a GRASP constructive heuristic.

Algorithm 1 Metaheuristic structure

```
Initialize both Problem and Heuristic parameters
$x \leftarrow$ GRASP()
$x^* \leftarrow \emptyset$, $f(x^*) \leftarrow \infty$
*max-iter-no-improvement* $\leftarrow 0$
*iterations* $\leftarrow 0$
while (*iterations* $<$ *MaxIterations*) do
   $x \leftarrow$ VND($x$)
   if ($f(x) < f(x^*)$) then
      $x^* \leftarrow x$,$f(x^*) \leftarrow f(x)$
      *max-iter-no-improvement* $\leftarrow 0$
   else
      *max-iter-no-improvement* $\leftarrow$ *max-iter-no-improvement* $+1$
   end if
   if (*max-iter-no-improvement*$<$*MaxPerturbuations*) then
      $x \leftarrow$ Perturbation($x$)
   else
      $x \leftarrow$ GRASP heuristic()
   end if
   *iterations* $\leftarrow$ *iterations*$+1$
end while
return $x^*$
```

After the GRASP procedure is finished, a local search is used to improve the current solution by using a VND heuristic. This local search is executed until the algorithm finds no more improvement. Once this is the case, a perturbation is applied to escape the local optimum, and the algorithm continues with a local search from this perturbed solution. If after a predefined number of perturbations no better solution can be found, the algorithm is restarted from a new solution generated by the GRASP heuristic.

4.1. Constructive phase

Our solution approach uses a greedy randomized adaptive search procedure (GRASP) to generate an initial solution x.

GRASP is a well-known constructive heuristic (see Feo et al. (1991) for more details), that, starting from an empty solution, generates a complete solution by upgrading one strategy at a time until either the security budget has been depleted or no security strategy upgrades are available.

The criticality of an edge is a value measuring the importance of that edge for the security of the network. The higher the criticality, the greater the impact of that edge on the risk reduction and thus on the improvement of the objective function. In our algorithm, the number of times that that edge occurs in a critical scenario multiplied by the remaining probability of that edge, after the application of a security strategy, is used as an estimator of criticality.

At each iteration of the GRASP heuristic the strategy to be upgraded in the current solution is randomly selected from a restricted candidate list (*RCL*). The *RCL* contains the first α *critical edges* ordered by decreasing value of criticality for which: (a) no security strategy has previously been included in the current solution and (b) at least one security strategy, not contained in the tabu list and whose cost is lower than the remaining budget, is available.

The size of the *RCL*, α, is a parameter that controls the balance between greediness and randomness. If α is large, the selection is relatively random, while if $\alpha = 1$ the greedy construction is completely deterministic.

In this way, at each iteration of the GRASP heuristic the list *RCL* contain the first α edges that can likely have a higher impact on the total risk considering the overall network. At each iteration of the GRASP heuristic an edge i is randomly chosen from the *RCL* and an available strategy j whose cost is lower than the remaining budget is selected for that edge. Two possible selection mechanisms can be used so select a strategy j for edge i at each iteration of the heuristic:

- *Best improvement* provides at each iteration the combination (i, j) which yields the largest reduction in the objective function given the α considered edges and their available security strategies; This is more demanding in terms of computation time.

- *First improvement* which guarantees a fast selection of a feasible combination (i, j). From the α edges one edge is chosen randomly, and for this edge a security strategy is selected in a random fashion. This selected combination (i, j) is than checked on feasibility, which means it is not used in the current solution, not contained in the tabu list and is less costly than the remaining budget. If the selected combination is not feasible, a new combination is selected in a random fashion, until a feasible combination is found. This combination provides a reduction of the risk for the overall network, but is not necessarily the best possible reduction, as the combination is selected randomly.

During the constructive method a tabu list is used in order to exclude the selection of strategies that have already been explored in previous iterations of the metaheuristic. After the selection process the current solution, the remaining budget and the *RCL* are all updated. The selected edge i is removed from the *RCL* as well as all the edges for which no strategy is available any longer due to a lower remaining budget. Then the *RCL* is reordered based on the new vulnerability values of the first α critical edges. The GRASP heuristic's selection mechanism is repeated until either the remaining budget is depleted or the list of available options is empty.

4.2. Improvement phase

The improvement of the solution, produced by the GRASP heuristic, is performed by a variable neighbourhood descent (VND) heuristic. VND is a deterministic variant of the well-known variable neighbourhood search (VNS) metaheuristic (Hansen & Mladenović, 2001). In general, VNS algorithms use a sequence of nested neighbourhood, $\mathcal{N}_1, \ldots, \mathcal{N}_{k_{max}}$ with an increasing size, i.e., $|\mathcal{N}_k| < |\mathcal{N}_{k+1}|$ and a perturbation move is used for diversification purposes.

The algorithm uses three neighbourhood structures. The first one, called *Internal Swap*, attempts to replace a strategy j for a given edge i with another strategy j', that is not contained in the tabu list and for which the remaining budget is larger than the difference in cost between j and j'.

The second neighbourhood structure, called *External Swap*, attempts to replace one strategy j for edge i, which will be assigned strategy $j = 0$, with another strategy j', which is not in the tabu list, associated to a different edge i'. In practice part of the security budget is transferred from edge i to edge i' with the cost of strategy j' lower than the remaining budget plus the cost of strategy j'.

The third neighbourhood structure, called *Double Swap*, is a variant of the first move where two *Internal Swap* moves are executed at the same time. In practice strategies j and j' associated to edges i and i' respectively are simultaneously swapped with other two strategies $\bar{j}$ and $\bar{j}'$ available for edges i and i'. The new strategies $\bar{j}$ and $\bar{j}'$ must not be contained in the tabu list and the cost of the strategies $\bar{j}$ and $\bar{j}'$ must not be greater than the remaining budget plus the cost of the removed strategies j and j'.

To speed up the improvement stage only a restricted number of edges, α, are considered and only moves which have a positive contribution to the quality of the current solution are executed. In particular, the *External Swap* move attempts to replace a strategy j associated to any edge i already in the solution with one of the available strategies associated to the first α edges that do not have a security strategy in the solution, which are ordered by decreasing value of criticality, as described before.

The size of the neighbourhood that *Double Swap* has to explore is equal to $\bar{\mathcal{E}} \times (\bar{\mathcal{E}} - 1)$ where $|\bar{\mathcal{E}}|$ is the number of edges that have a security strategy in the current solution. For this reason the number of evaluations that are performed by the *Double Swap* is restricted to a maximum value based on a percentage of set $\bar{\mathcal{E}}$. The candidates for the double swap are selected randomly from all edges that have a security strategy that is different from strategy 0.

Two different selection mechanisms can be used by the VNS heuristic to improve the current solution. The first one is based on a *first improvement* mechanism (see Section 4.1) while the second one relies on a *best improvement* mechanism, which are explained in section 4.1. The VND heuristic ends when no improvement of the current solution can be found.

4.3. Diversification phase

The diversification phase attempts to escape from the local optimum, reached in the improvement phase, by exploring different areas of the search space hopefully not yet explored. Two different diversification mechanism are used in our variable neighborhood tabu search metaheuristic: (1) If a maximum number of iterations without improvement of the current solution has not been reached, a perturbation heuristic is used; (2) if the the number of iterations without improvement becomes larger than a predefined maximum, a new initial solution is generated using the GRASP constructive heuristic.

The perturbation heuristic partially destroys the current solution by setting the strategies that have been used for a number of edges to zero. These strategies are inserted in a tabu list in order to avoid their reuse in other solutions for a given number of iterations. The perturbation heuristic allows to free some budget resources, making room for new, unused strategies that are not in the tabu list. These new strategies are selected using either a best improvement or a first improvement selection mechanisms (see Section 4.1) and added to the current solution as long as budget is available and at least one strategy, with a cost less than the remaining budget and not contained in the tabu list, is available. The newly generated solution becomes the input of the VND heuristic for further improvement.

When the the number of iterations without improvement becomes larger than a predefined maximum, a new solution is generated using the GRASP constructive heuristic.

5. Computational experiments

In this section, the stages used to execute our computational experiments are described. First, a set of instances is generated and used later on for tuning and testing the metaheuristic described before (Section 5.1).

In the second stage, the parameters of the metaheuristic are tuned in a statistical experiment in order to achieve the best results (Section 5.2).

Finally, the metaheuristic with its best parameter setting is used to solve the test instances and computational results are shown in Section 5.3.

5.1. Instance generation

To test the algorithm developed in this paper a library of test instances has been created. All instances can be found on `http://antor.uantwerpen.be/downloads/NS`. The software that was developed to generate the instances, takes several parameters as input, necessary to specify the properties of the instances. These parameters are: (1) the maximum budget that can be spent on strategies; (2) the number of nodes in the network; (3) the maximum number of strategies for each edge that connects two nodes; (4) the percentage of edges of the Delaunay triangulation (Delaunay, 1934) that should be added to the minimum spanning tree, to add some redundancy as explained below; (5) the number of instances to be generated for each set of parameters.

The number of scenarios contained in $\mathcal{C}$, the set of all critical scenarios, is dependent on the number of edges in and the topology of the network. In the worst case, the cardinality of this set is equal to $2^{|\mathcal{E}|}$. This implies, e.g., that adding an edge to a network containing 20 edges, requires 1048576 ($2^{21} - 2^{20}$) more scenarios to be analysed and stored in memory. Due to the limitations in memory of the computer on which the algorithm is executed, instances with 20 edges are considered to be large.

A first step in the instance generation process is the generation of nodes. Each node's coordinates are randomly selected from a uniform distribution between 0 and 100. This range can be adjusted by the decision maker. When the nodes have been generated, a Delaunay triangulation of this set is created. The Delaunay triangulation is used to identify the "neighbours" of each node, which will be used to create the final network. When the neighbours of all nodes are identified, Kruskal's algorithm (Kruskal, 1956) is used to generate a minimum spanning tree. The motivation for starting from a minimum spanning tree is that this is the most cost-effective way to connect all nodes in the network.

In real life cases, service providers often add redundancy to increase operational security. If one edge gets disabled, customers are serviced through the redundant edge. For this reason, the minimum spanning tree is extended with edges to mimic the redundancy that is present in real life networks. This is done by selecting a number of edges from the Delaunay triangulation which are not used in the minimal spanning tree. The edges selection is based on their length, shortest first, and on a parameter that is defined by the decision maker.

When the edges of the network have been generated, each edge is assigned a random probability of attack. The probability of failure for the edges are different for each edge. This is done to make the model more general. From a security related perspective, this can be justified by the difference in risks based on geographical location, length and criticality of that edge in the network. From the perspective of safety, it could be that certain sections in the network are added on a later point in time, which would yield a lower risk for failure due to the newer pieces that were used, and the lower amount of normal wear and tear.

When each edge has been assigned a probability, the algorithm generates a random number of strategies with a minimum of one, and a maximum number as defined in the parameters. Each strategy is assigned a random reduction in probability of failure between 1 and 20 percent, and a random cost based on this percentage reduction. To obtain this cost, this reduction is multiplied with a random number between 0.5 and 1.5 and the maximum budget, to get realistic values. The reduction is selected on practical grounds. An improvement larger than 20 percent will in practice not be possible given the budget constraint.

The generated test instances' names are encoded as follows:

`NS-n8-c5-C3-a30-x0`, where NS is referring to the network security problem, n represents the number of nodes in the instance, c gives the maximum number of counter measures generated for each edge, C is the maximum number of connections from and to a node (This is currently not used in the instance generator, but the option is foreseen in the encoding), a represents the percentage of extra arcs from the Delaunay triangulation added to the minimal spanning tree, and x represents the number of the instance when multiple instances are generated at once, with the same settings.

5.2. Parameter tuning

The metaheusritic described before uses some internal parameters that need to be tuned in order to have a good compromise between speed of the heuristic and quality of the solutions. The internal parameters have been described before in Section 4 and are summarized in Table 4 together with the tested values.

Table 4: Heuristic parameters

Parameter	**Description**	**Values**	**#**
`max-iter`	Number of restarts	50	1
`perturb-percent`	Percent of edges removed during the perturbation phase	10%, 30%, 50%, 70%	4
`alpha`	Size of the restricted candidate list	1, 2, 3, 4	4
`tabu-tenure`	Number of iterations that a strategy is kept in the tabu list	10, 30, 50	3
`max-iter-no-improvement`	Number of iterations without improvements	5, 10, 20	3
`double-swap-percentage`	Percentage value used to find the amount of evaluations to be performed by the Double swap move	20%, 50%, 80%	3
`selection mechanism`	Strategy to select edges	first, best	2

The parameter tuning experiment is conducted on a set of instances consisting of smaller networks of 10 nodes (10 - 12 edges) and larger networks of 18 nodes and 20 edges.

A full factorial statistical experiment is conducted using the values reported in Table 4. The objective values and times are averaged, and shown in Figure 4. The parameters with an asterisk had a significant effect on both the objective value and time, while parameter `tabu-tenure` was significant only in regard to the time. Parameter `max-iter-no-improvement` did not have a significant effect. A parameter setting equal to 10 was selected.

Only taking into account at the objective function, the chosen settings for the metaheuristic parameters are shown in Table 5.

5.3. Computational results

After having selected the best parameters settings for the solution approach, the influence of each metaheuristic component on both the quality of the solutions and the running time is analysed. A set of instances ranging from 9 up to 20 edges and 5 up to 20 security strategies per edge is used for this purpose.

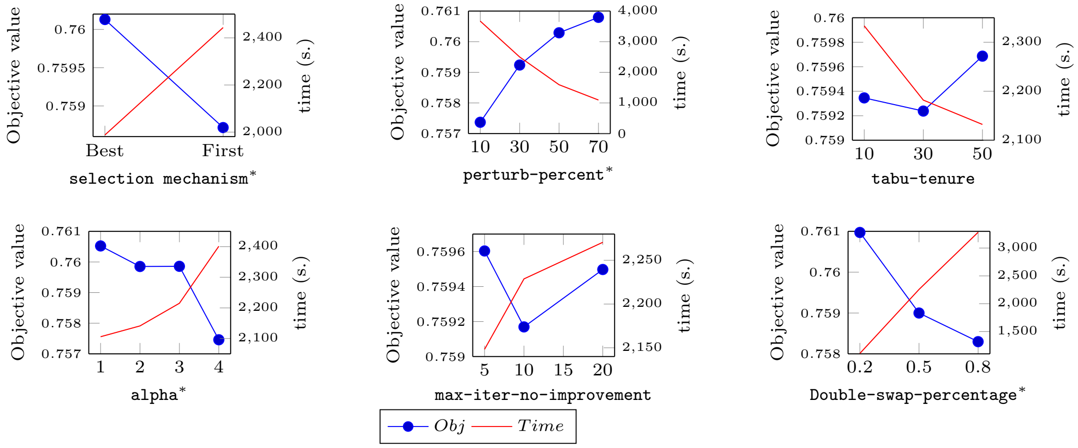


Figure 4: Plot of average objective values and times for given parameter setting

Table 5: Selected heuristic parameters

Parameter	**Setting**
`max-iter`	50
`perturb-percent`	10%
`alpha`	4
`tabu-tenure`	30
`max-iter-no-improvement`	10
`Double-swap-percentage`	80%
`selection mechanism`	first

In order to analyse the quality of the solutions, the results found by the proposed metaheuristic are compared with the optimal solutions obtained by a naive exact approach. The exact approach is based on an exhaustive exploration of the search space where the budget constraint is used as a cutting plane to speed up the running time. In Table 6 the comparison between the metaheuristic and the exact approach is reported. The metaheuristic was run 25 times on each instance to compute the average running time and objective value that are reported in Table 6 together with the best solution found over these runs. The metaheuristic obtained 9 optimal solutions (highlighted in bold in column Best Gap in Table 6) out of 16 test instances. The percentage optimality gap, used as an indicator of the quality of the obtained solutions, is around 0.51%. Considering all the solutions averaged over 25 runs the gaps from the optimal solutions is only 1.62%.

The run times needed by the exact approach, increase exponentially with the number of edges in the network, while the run time required by the metaheuristic remains stable. Analysing both the exact approach and the metaheuristic, it becomes clear that the proposed metaheuristic achieves good results in a short amount of time.

	Exact Approach		**Metaheuristic**				
Instance	Optimal Solution	Time(s.)	Best Solution	Average Solution	Average Time (s.)	Best Gap	Average Gap
NS-n8-c5-C3-a30-x0	0.64813	636.910	0.65185	0.65468	0.147	0.57%	1.01%
NS-n8-c5-C3-a30-x1	0.81428	37.275	0.81428	0.84409	0.614	**0.00%**	3.66%
NS-n8-c5-C3-a30-x2	0.18171	104.381	0.18483	0.18549	0.642	1.71%	2.08%
NS-n8-c5-C3-a30-x3	0.13410	57.631	0.13711	0.14310	0.573	2.24%	6.71%
NS-n8-c5-C3-a30-x4	0.64803	54.456	0.65369	0.66393	0.439	0.87%	2.45%
NS-n8-c5-C3-a30-x5	0.34060	13.512	0.34060	0.34333	0.504	**0.00%**	0.80%
NS-n8-c5-C3-a30-x6	0.25193	3.372	0.25833	0.26179	0.697	2.54%	3.91%
NS-n8-c5-C3-a30-x7	0.07905	65.642	0.07905	0.07947	0.549	**0.00%**	0.54%
NS-n9-c5-C3-a30-x0	0.89334	1356.729	0.89442	0.89842	0.527	0.57%	0.12%
NS-n9-c5-C3-a30-x1	0.82052	5744.238	0.82052	0.82149	0.702	**0.00%**	0.12%
NS-n9-c5-C3-a30-x10	0.41173	222.962	0.41173	0.42319	0.218	**0.00%**	2.78%
NS-n9-c5-C3-a30-x11	0.07290	588.678	0.07290	0.07290	0.254	**0.00%**	0.00%
NS-n9-c5-C3-a30-x12	0.79698	1160.176	0.79700	0.79754	0.414	**0.00%**	0.07%
NS-n9-c5-C3-a30-x14	0.11105	7825.799	0.11105	0.11123	0.503	**0.00%**	0.16%
NS-n9-c5-C3-a30-x4	0.40821	3178.317	0.40821	0.40962	0.304	**0.00%**	0.34%
NS-n9-c5-C3-a30-x8	0.43050	8751.806	0.43067	0.43340	0.594	0.04%	0.67%

Table 6: Exact approach in comparison with the metaheuristic for instances with 8 nodes and 9 edges (instances name with the code n8) and with 9 nodes and 11 edges (instances name with the code n9) and 5 security strategies per edge

For larger instances, computing times of the exact approach quickly become intractable. For an instance with 13 edges, the exact approach took around 3 days to find the optimal solution. The heuristic solution was only 1.77% from this optimal value, and was found in 0.475 seconds. A time limit of 5 hours was imposed on the exact solution method to find a good upper-bound. This value is used in Table 7 to evaluate the quality of the results provided by the metaheuristic.

It is worth noticing that the solution obtained by using a time limit for the instance NS-n10-c5-C3-a30-x2 (shown in Table 7) resulted in a good upper bound, presenting only a difference of 0.11% from the optimal known solution. This means that the selected time limit is adequate, for that instance, to achieve a good upper-bound to be used to analyse the quality of the solutions obtained by the metaheuristic.

Some results are shown in Table 7, where on average the metaheuristic improved the level of security of the initial network by 32%. In eight cases, the metaheuristic outperformed the exact approach with the time limit, finding better results in a short computation time.

		Exact Approach		**Metaheuristic**				
Instance	Original risk	Upper-bound	Time(s.)	Best Solution	Average Solution	Time(s.)	Best Gap	Average Gap
NS-n10-c5-C3-a30-x0	0.99998	0.91833	18000	0.91276	0.91320	0.614	-0.61%	-0.56%
NS-n10-c5-C3-a30-x1	0.99996	0.20114	18000	0.16652	0.16743	0.374	-17.21%	-16.76%
NS-n10-c5-C3-a30-x2*	≈1	0.86129	18000	0.87553	0.88744	0.475	1.65%	3.04%
NS-n10-c5-C3-a30-x3	≈1	0.58132	18000	0.55206	0.55669	0.453	-5.03%	-4.24%
NS-n10-c5-C3-a30-x4	≈1	0.64521	18000	0.64708	0.67119	0.353	0.29%	4.03%
NS-n18-c10-C3-a30-x0	≈1	0.99675	18000	0.98455	0.98461	438.199	-1.22%	-1.22%
NS-n18-c10-C3-a30-x1	≈1	0.61578	18000	0.55764	0.55771	207.571	-9.44%	-9.43%
NS-n18-c10-C3-a30-x2	≈1	0.94620	18000	0.90084	0.90093	363.550	-4.79%	-4.78%
NS-n18-c10-C3-a30-x3	≈1	0.40736	18000	0.31192	0.31198	343.673	-23.43%	-23.41%
NS-n18-c10-C3-a30-x4	≈1	0.93503	18000	0.84476	0.84497	255.071	-9.65%	-9.63%

Table 7: Results for instances in the range 10-20 edges and 5-20 security strategies per edge

Figure 5 reports the evolution of the objective function associated with both the best and the current solutions over time. It can be noticed that our solution approach is able to converge towards good results in a short CPU time also in case of large instances. The marginal improvement of the best solution found so far significantly decreases when the running time is increased.

Analysing the plot associated with the current solution in Figure 5 clarifies the behaviour of the solution approach. After the perturbation, that destroys the quality of the solution, small improvements obtained during the VND heuristic can lead to better solutions. One can clearly distinguish the perturbation strategy that allows the algorithm to efficiently escape from local optima. Starting from the perturbed solution, denoted with a peak in the graph in Figure 5, the VND heuristic guides the current solution through small improvements towards a new local optimum and hopefully a new and better solution. The fact that the VND heuristic is able to decrease the value of the perturbed solution and detect new local optima proves the efficiency of the VND.

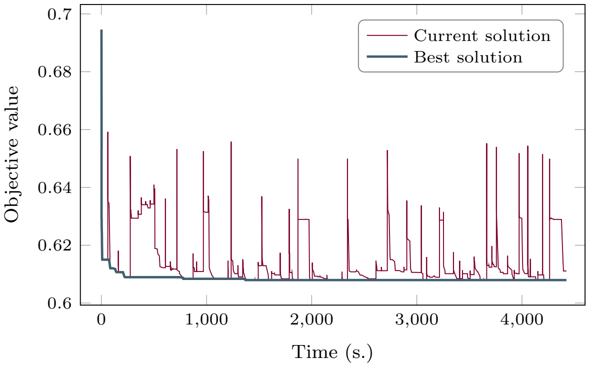


Figure 5: Plot of the objective value over time.

The effect of each metaheuristic component on the objective value is also analysed. The VND heuristic on average can improve the initial solutions generated by the GRASP constructive heuristic by 1%.

6. Conclusion

In this paper, a model for the selection of the appropriate security strategies, given a limited budget, is proposed to increase the security of a network infrastructure such as pipelines transportation systems, telecommunication networks and smart grids.

By selecting one origin node and one destination node, it is possible to define the risk of an interruption of service (or material flow) due to external malicious attacks.

An exact evaluation of the risk of the whole network being unavailable might be a difficult task, especially when several loops are present inside the network. In order to reduce the complexity of computations, a heuristic approach to have an accurate estimate of the risk of the network being down is defined.

The variable neighborhood tabu search heuristic has been tuned in order to find its best configuration when minimising the objective value. In a second stage of the computational experiments, the tuned metaheuristic has been employed to solve a set of test instances that mimic possible realistic scenarios and the performance of the algorithm is evaluated.

The variable neighborhood tabu search heuristic can find very good results in a really short amount of time. It is performing extremely well in comparison to the naive exact approach. However, due to the fact that we need to generate and evaluate all scenarios to find the critical scenarios and the sheer number of critical scenarios, to tackle larger instances, a different approach could be investigated.

Future research could be aimed at extending the model to support more realistic scenarios where e.g., the nodes are vulnerable and the connection of all nodes in the network should be ensured. In addition supplier capacity, customer demand and importance of either of them might be considered.

Acknowledgements

This research was partially supported by the Interuniversity Attraction Poles (IAP) Programme initiated by the Belgian Science Policy Office (COMEX project).

References

Agarwal, P. K., Efrat, A., Ganjugunte, S. K., Hay, D., Sankararaman, S., & Zussman, G. (2013). The resilience of WDM networks to probabilistic geographical failures. *IEEE/ACM Transactions on Networking (TON)*, *21*, 1525 – 1538.

Antonioni, G., Bonvicini, S., Spadoni, G., & Cozzani, V. (2009). Development of a framework for the risk assessment of Na-Tech accidental events. *Reliability Engineering & System Safety*, *94*, 1442 – 1450.

Barker, K., Ramirez-Marquez, J. E., & Rocco, C. M. (2013). Resilience-based network component importance measures. *Reliability Engineering & System Safety*, *117*, 89 – 97.

Bazovsky, I. (2004). *Reliability Theory and Practice*. Dover Civil and Mechanical Engineering Series. Dover Publications.

Bistarelli, S., Fioravanti, F., & Peretti, P. (2007). Using CP-nets as a guide for countermeasure selection. In *Proceedings of the 2007 ACM symposium on Applied computing* (pp. 300 – 304). March 11 - 15, Seoul, Republic of Korea: ACM.

Bojanc, R., & Jerman-Blažič, B. (2008). An economic modelling approach to information security risk management. *International Journal of Information Management*, *28*, 413 – 422.

Bretthauer, K. M., & Shetty, B. (2002). The nonlinear knapsack problem – algorithms and applications. *European Journal of Operational Research*, *138*, 459 – 472.

Delaunay, B. N. (1934). Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, *6*, 793 – 800.

Feo, T., Bard, J., & Venkatraman, K. (1991). A *GRASP* for a difficult single machine scheduling problem. *Computer and Operations Research*, *18*, 635 – 643.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, *130*, 449 – 467.

Kerivin, H., & Mahjoub, A. R. (2005). Design of survivable networks: A survey. *Networks*, *46*, 1–21.

Kröger, W. (2008). Critical infrastructures at risk: A need for a new conceptual approach and extended analytical tools. *Reliability Engineering & System Safety*, *93*, 1781 – 1787.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, *7*, 48 – 50.

Ministry of Defence (UK) (2011). Chapter 6: Probabilistic R&M Parameters and redundancy calculations. In *Applied R&M Manual for Defence Systems (GR-77), Part D - Supporting Theory*. Abbey Wood, Bristol: UK Ministery of Defence.

Reniers, G., & Dullaert, W. (2012). TePiTri: A screening method for assessing terrorist-related pipeline transport risks. *Security Journal*, *25*, 173 – 186.

Reniers, G., Dullaert, W., Audenaert, A., Ale, B., & Soudan, K. (2008). Managing domino effect-related security of industrial areas. *Journal of Loss Prevention in the Process Industries*, *21*, 336 – 343.

Reniers, G., Herdewel, D., & Wybo, J.-L. (2013). A threat assessment review planning (TARP) decision flowchart for complex industrial areas. *Journal of Loss Prevention in the Process Industries*, *26*, 1662 – 1669.

Reniers, G., Sörensen, K., & Dullaert, W. (2012). A multi-attribute systemic risk index for comparing and prioritizing chemical industrial areas. *Reliability Engineering & System Safety*, *98*, 35 – 42.

Reniers, G., Sörensen, K., Khan, F., & Amyotte, P. (2014). Resilience of chemical industrial areas through attenuation-based security. *Reliability Engineering & System Safety*, *131*, 94 – 101.

Romero, J. (2012). Blackouts illuminate india's power problems. *Spectrum, IEEE*, *49*, 11–12. doi:10.1109/MSPEC.2012.6309237.

Romeu, J. L. (2004). Understanding series and parallel systems reliability. *Selected Topics in Assurance Related Technologies (START)*, *1*, 1 – 8.

Sawik, T. (2013). Selection of optimal countermeasure portfolio in IT security planning. *Decision Support Systems*, *55*, 156 – 164.

START (2014). The Global Terrorism Database (GTD), . http://images.fedex.com/us/about/today/access/GreaterAccessChange_global.pdf, visited on 2014-10-29.

Steiglitz, K., Weiner, P., & Kleitman, D. (1969). The design of minimum-cost survivable networks. *Circuit Theory, IEEE Transactions on*, *16*, 455–460.

Talarico, L., Sörensen, K., Reniers, G., & Springael, J. (2015). Pipeline security. In S. Hakim, G. Albert, & Y. Shiftan (Eds.), *Securing Transportation Systems*. New York: Wiley.

Viduto, V., Maple, C., & Huang, D., W.and López-Peréz (2012). A novel risk assessment and optimisation model for a multi-objective network security countermeasure selection problem. *Decision Support Systems*, *53*, 599 – 610.