






# Detecting Weak Points in Networks Using Variable Neighborhood Search

Sergio Pérez-Peló , Jesús Sánchez-Oro <sup>(✉)</sup> , and Abraham Duarte 

Department of Computer Sciences, Universidad Rey Juan Carlos, Móstoles, Spain  
{sergio.perez.pelo,jesus.sanchezoro,abraham.duarte}@urjc.es

**Abstract.** Recent advances in networks technology require from advanced technologies for monitoring and controlling weaknesses in networks. Networks are naturally dynamic systems to which a wide variety of devices are continuously connecting and disconnecting. This dynamic nature force us to maintain a constant analysis looking for weak points that can eventually disconnect the network. The detection of weak points is devoted to find which nodes must be reinforced in order to increase the safety of the network. This work tackles the  $\alpha$  separator problem, which aims to find a minimum set of nodes that disconnect the network in subnetworks of size smaller than a given threshold. A Variable Neighborhood Search algorithm is proposed for finding the minimum  $\alpha$  separator in different network topologies, comparing the obtained results with the best algorithm found in the state of the art.

**Keywords:** Alpha-separator · Reduced VNS · Betweenness

## 1 Introduction

Nowadays cybersecurity is becoming one of the most relevant fields for any kind of users: from companies and institutions to individual users. The increase in the number of attacks to different networks in the last years, as well as the relevance of the privacy in the Internet, have created the necessity of having more secure, reliable and robust networks. A cyberattack to a company that causes loss of personal information of their clients can result in important economic and social damage [1]. Furthermore, Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are becoming more common since a successful attack can result in disabling a service of an Internet provider, for instance. Even more, if several services are dependent on the service under attack, a cascade failure can occur, affecting to a large number of clients [3].

It is important to identify which are the most relevant nodes in a network. This is a matter of interest for both actors in a cyberattack: the attacker and the defender. The former is interested in disabling these nodes in order to make

---

This work has been partially founded by Ministerio de Economía y Competitividad with grant ref. TIN2015-65460-C2-2-P.

© Springer Nature Switzerland AG 2019

A. Sifaleras et al. (Eds.): ICVNS 2018, LNCS 11328, pp. 141–151, 2019.

[https://doi.org/10.1007/978-3-030-15843-9\\_12](https://doi.org/10.1007/978-3-030-15843-9_12)

the network more vulnerable while the latter is focused on reinforcing these important nodes with more robust security measures. On the one hand, the attacker is interested in causing the maximum damage to the network while consuming the minimum amount of resources. On the other hand, the defender wants to reinforce the network minimizing the increase in the maintenance and security costs. Therefore, it is interesting for both parts to identify which are the weak points in a network.

We define a network as a graph  $G = (V, E)$ , where  $V$  is the set of vertices,  $|V| = n$ , and  $E$  is the set of edges,  $|E| = m$ . A vertex  $v \in V$  represents a node of the network while an edge  $(v, u) \in E$ , with  $v, u \in V$  indicates that there is a connection in the network between vertices  $v$  and  $u$ . Let us also define a separator of a network as a set of vertices  $S \subseteq V$  whose removal cause the partition of the network into two or more connected components. More formally,

$$V \setminus S = C_1 \cup C_2 \dots \cup C_p \\ \forall (u, v) \in E^* \exists C_i : u, v \in C_i$$

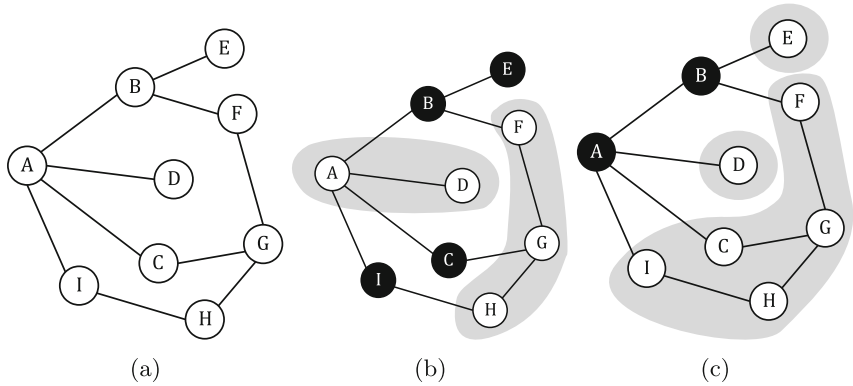
where  $E^* = \{(u, v) \in E : u, v \notin S\}$ .

This work is focused on finding a minimum  $\alpha$ -separator  $S^*$  which is able to split a network  $G$  into connected components of sizes smaller than  $\alpha \cdot n$ . In mathematical terms,

$$S^* \leftarrow \arg \min_{S \in \mathcal{S}} |S| \quad : \quad \max_{C_i \in V \setminus S} |C_i| \leq \alpha \cdot n$$

It is worth mentioning that the number of resulting connected components is not relevant in this problem neither as a constraint nor for evaluating the objective function value. The actual constraint of the  $\alpha$ -SP is that the number of vertices in each connected component must be lower or equal than  $\alpha \cdot n$ , where  $\alpha$  is an input value. This problem is  $\mathcal{NP}$ -hard for general networks topologies when considering  $\alpha \leq \frac{2}{3}$  [6]. Some polynomial-time algorithms have been proposed when the topology of the network is a tree or a cycle [15]. However, these algorithms require to have previous knowledge on the topology of the network, which is not usual in real-life problems.

Figure 1 shows an example of a network and two feasible solutions for the  $\alpha$ -SP. The network depicted in Fig. 1(a) is conformed with 9 vertices and 10 edges connecting those vertices. We consider  $\alpha = \frac{2}{3}$  for this example, so the connected components of the network must contain  $\lceil \frac{2}{3} \cdot 9 \rceil = 6$  vertices at most. Figure 1(b) shows a feasible solution  $S_1 = \{B, C, E, I\}$  which divides the network into two connected components  $C_1 = \{A, D\}$ , and  $C_2 = \{F, G, H\}$ , whose number of vertices (2 and 3, respectively), are smaller than 6. The second solution,  $S_2 = \{A, B\}$ , depicted in Fig. 1(c), divides the network in three connected components:  $C_1 = \{E\}$ ,  $C_2 = \{D\}$ , and  $C_3 = \{C, G, H, I\}$  (all of them with sizes smaller than 6). Notice that  $S_2$  is better than  $S_1$  in terms of objective function value since it is able to disconnect the network by removing just 2 vertices, while  $S_1$  requires to remove 4 vertices in order to disconnect the network. Notice that neither the size nor the number of connected components affect to the quality of the solution.



**Fig. 1.** (a) Example of a graph derived from a network, (b) a feasible solution with 4 nodes in the separator (B,C,E, and I), and (c) a better solution with 2 nodes in the separator (A and B)

This problem has been tackled for both exact and heuristic perspectives. Specifically, polynomial-time algorithms have been presented for special topologies as trees or cycles, as well as a greedy algorithm with approximation ratio of  $\alpha \cdot n + 1$  [15]. Additionally, a heuristic algorithm for studying the node separators in the Internet Autonomous Systems was proposed [17]. Depending on the  $\alpha$  value, the  $\alpha$ -separator problem can be related to different well-known problems. In particular, when  $\alpha = \frac{1}{n}$ , it is equivalent to the minimum vertex cover problem, and when  $\alpha = \frac{2}{n}$  it is analogous to the minimum dissociation set problem. Therefore, the  $\alpha$ -separator problem is a generalization of these problems, which are also  $\mathcal{NP}$ -hard [9]. As far as we know, the best previous heuristic consists of a random walk algorithm which is based on a Markov Chain Monte Carlo method [13].

## 2 Algorithmic Proposal

Variable Neighborhood Search (VNS) [11] is a metaheuristic framework based on systematic changes of neighborhoods. As a metaheuristic algorithm, it does not guarantee the optimality of the solutions obtained, but it is focused on obtaining high quality solutions in reasonable computing times. There are several variants of VNS, which are classified taking into account fundamentally the exploration of the considered neighborhoods. Typically, the neighborhood structures are analyzed using three different criterion: stochastic (Reduced VNS, RVNS), deterministic (Variable Neighborhood Descent, VND), or a combination of both deterministic and stochastic (Basic VNS, BVNS). Furthermore, several additional variants have been proposed in the last years: General VNS (GVNS), Variable Neighborhood Decomposition Search (VNDS), Skewed VNS (SVNS), Variable Formulation Search (VFS), among others.

The stochastic exploration of the search space is usually recommended for those problems where the local search method is very computationally demanding. Additionally, it is useful for problems in which the design of a local search is not clear since the definition of the neighborhoods to be explored is very complex to be considered in a fast heuristic. In the context of  $\alpha$ -SP, any neighborhood that performs small moves over a feasible solution will probably result in a non-feasible solution and, therefore, a repair method must be applied after performing each move. The repair method should consider the feasibility of the solution, which is very time consuming in the problem under consideration, increasing the time required to perform a local search.

As stated in Sect. 1, solutions for the  $\alpha$ -SP should be generated as fast as possible, since the integrity of the network highly depends on the performance of the algorithm, not only in terms of solution quality but also with respect to the computing time required to produce a solution. RVNS is usually compared with a Monte-Carlo method, but being RVNS more systematic [14]. Indeed, RVNS has been able to obtain results competitive with the Fast Interchange [18] method when applied to the  $p$ -Median problem [10]. This work presents a Reduced VNS algorithm in order to generate high quality solutions in short computing time. Algorithm 1 presents the general framework of RVNS.

---

**Algorithm 1.** *RVNS*( $S, k_{\max}$ )

---

```

1: repeat
2:    $k \leftarrow 1$ 
3:   while  $k \leq k_{\max}$  do
4:      $S' \leftarrow Shake(S, k)$ 
5:      $k \leftarrow NeighborhoodChange(S, S', k)$ 
6:   end while
7: until StoppingCriterion
8: return  $S$ 

```

---

RVNS starts from an initial solution  $S$ , which can be generated using a random procedure or any other complex heuristic or metaheuristic. Additionally, the maximum neighborhood  $k_{\max}$  to be explored must be indicated. As stated in previous works [12], the maximum neighborhood to be considered in the RVNS algorithm is usually small, to avoid exploring completely different solutions in each iteration. Finally, the third parameter of the algorithm indicates the stopping criterion. For this work, we consider a maximum number  $\lambda$  of RVNS iterations.

In each iteration, the algorithm starts from the first considered neighborhood  $k = 1$  (step 2). Then, the algorithm iterates until reaching the maximum neighborhood  $k_{\max}$  (steps 3–6). In each iteration, a random solution  $S'$  in the current neighborhood  $k$  is generated (step 4). Then, the algorithm selects the next neighborhood to be explored (step 5). In particular, if the objective function value of  $S'$  is better than the one of  $S$ , an improvement is found, updating the best solution found ( $S \leftarrow S'$ ) and restarting the search from the first neighborhood ( $k = 1$ ). Otherwise, the search continues with the next neighborhood

( $k \leftarrow k + 1$ ). The method ends when performing  $\lambda$  iterations of RVNS (steps 1–7), returning the best solution found during the search (step 8).

## 2.1 Constructive Method

The main objective of the  $\alpha$ -SP is to identify the most important nodes in a network trying to minimize the size of the separator. To achieve this goal, we can leverage several characteristics of a node of a graph, such as its degree (number of edges of a node), its position in the network, or any centrality measure, among others, as a selection criterion to find the most important vertices in a graph.

This work proposes a greedy constructive procedure for generating the initial solution. In particular, we propose a greedy function that evaluates the relevance of a vertex in a graph using a centrality metric known as betweenness centrality [2], an extended metric in the context of finding relevant users in social networks.

Betweenness centrality considers that a node is important within a network if it acts as a flow of information in the graph. In order to look for relevant nodes with respect to this metric, it is necessary to evaluate all the paths between any pair of nodes of the graph, being the relevance of a vertex directly related to the number of paths in which it appears. The rationale behind this is that if a vertex  $v$  participates in several paths of the network, then any information transmitted through it will eventually traverse  $v$ .

The betweenness centrality of a node  $v \in V$ , named as  $b(v)$ , is evaluated as the number of paths between any pair of nodes  $s$  and  $t$  in which  $v$  is included,  $\sigma(s, t|v)$ , divided by all the paths that connect  $s$  and  $t$ ,  $\sigma(s, t)$ . More formally,

$$b(v) \leftarrow \sum_{s, t \in V \setminus \{v\}} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

It is worth mentioning that the evaluation of the betweenness is a computationally demanding process. Specifically, the betweenness of a node  $v$  requires from the evaluation of all shortest paths between every pair of vertices  $s, t \in V$ .

In order to reduce the complexity of this evaluation, we consider an approximation of this metric by evaluating a number of shortest paths between every pair of nodes using a fast algorithm for finding shortest paths without loops in networks based on Yen's algorithm [19].

We have selected this criterion because it seems logical to think that, the more information circulates through a node in a graph, the more important this node will be within the network. Therefore, it will be easier to disconnect the network if priority is given to eliminate the nodes with a higher value of betweenness centrality.

A totally greedy algorithm will always produce the same initial solution, since it is focused on intensifying the search. However, several works [4, 16] have shown that introducing some randomness in the search, thus increasing the diversification, results in better initial solutions. We propose the use of Greedy Randomized Adaptive Search Procedure (GRASP) methodology in order to generate more diverse initial solutions that will eventually lead the algorithm to obtain better results.

GRASP methodology was originally proposed in 1989 [7] but it was not formally defined until 1994 [8]. Traditional GRASP algorithms consists of two well-differenced phases: construction and local search. In this work we just consider the construction stage, since a local search for the  $\alpha$ -SP is rather computationally demanding, resulting in large computing times.

The constructive procedure proposed starts by randomly selecting the first node  $v$  to be removed from the graph. Then, a candidate list is constructed with all the vertices  $u \in V \setminus \{v\}$ . In each iteration, a vertex is selected from the candidate list and removed from the graph. The selection of the next vertex is performed as follows. Firstly, a restricted candidate list is created with the most promising nodes of the candidate list. For this problem, we evaluate the betweenness of all the candidate vertices. Let us consider that  $g_{\max}$  and  $g_{\min}$  are the maximum and minimum values for this metric, respectively. Then, a threshold  $\mu$  is evaluated as:

$$\mu = g_{\max} - \beta * (g_{\max} - g_{\min})$$

The restricted candidate list contains all the candidate vertices whose betweenness value is larger or equal than the threshold  $\mu$ . Notice that  $\beta \in [0, 1]$  is a parameter of the constructive method that controls its degree of randomness. On the one hand,  $\beta = 0$  results in  $\mu = g_{\max}$ , which considers a totally greedy algorithm. On the other hand,  $\beta = 1$  results in  $\mu = g_{\min}$ , which considers a completely random procedure. Section 3 will evaluates different values for this parameter, discussing how it affects to the initial solution quality.

We have made a series of preliminary experiments in order to test what is the  $\alpha$  parameter that works better with our greedy criterion. Concretely, we have tested our algorithm using 0.25, 0.5, 0.75 and RND values. We present the results of these experiments in Sect. 3.

## 2.2 Perturbing Solutions for the $\alpha$ -SP

*Shake* method is responsible for finding new solutions in the neighborhood under exploration in the RVNS framework. First of all, it is important to define the neighborhoods considered in this work. We define the neighborhood  $N_1(S)$  of a solution  $S$  as the insertion of a new node in the solution. In mathematical terms,

$$N_1(S) = \{S' \leftarrow S \cup \{v\} : v \in V \setminus S\}$$

Analogously, we define the neighborhood  $N_k(S)$  as the insertion of  $k$  new nodes in the solution  $S$ . More formally,

$$N_k(S) = \{S' \leftarrow S \cup T : \forall v \in T, (v \in V \setminus S) \wedge (|T| = k)\}$$

Notice that any solution  $S'$  obtained in the neighborhood  $N_k$  of solution  $S$  presents more vertices included in it. If the quality of a solution is given by the number of vertices included in it and  $\alpha$ -SP is a minimization problem, then any solution  $S' \in N_k(S)$  is worse than  $S$  in terms of objective function

value. However, the main advantage of exploring this neighborhood is that every solution is always feasible, being unnecessary to check the constraint of the problem, which is one of the most time consuming parts of the algorithm.

Since the *Shake* procedure always deteriorates the quality of the solution perturbed, it is necessary to define a post-processing method that tries to improve its quality. For this purpose, we define a refining process that consists of removing all the vertices that are unnecessarily included in the perturbed solution  $S'$ . Specifically, the method randomly traverses the set of vertices that were originally in the solution (i.e.,  $S' \setminus T$ ). If the solution becomes unfeasible after removing a vertex, it is included again in  $S'$ . After trying to remove all vertices in  $S' \setminus T$ , the method repeats this instructions for all the vertices included in  $T$ . Following this procedure, if the number of removed nodes from the solution during the refining process is larger or equal than  $k$ , an improvement has been found, restarting the search from the first neighborhood. Otherwise, the search continues in the next neighborhood. The method stops when reaching the maximum predefined one,  $k$ , returning the best solution found during the search.

### 3 Computational Results

This Section is devoted to analyze the performance of the proposed algorithms and compare the obtained results with the best previous method found in the state of the art. The algorithms have been developed in Java 9 and all the experiments have been conducted on an Intel Core 2 Duo 2.66 GHz with 4 GB RAM.

The set of instances used in this experimentation has been generated using the same graph generator proposed in the best previous work [13]. Specifically, graphs are generated by using the Erdős-Rényi model [5], in which a new node is linked to the nodes already in the graph with the same probability. We have generated a set of 50 instances whose number of vertices ranges from 100 to 200 and the number of edges ranges from 200 to 2000.

We have divided the experiments into two different parts: preliminary and final experimentation. The former is designed for finding the best parameters of the GRASP constructive procedure and RVNS algorithm, while the latter is devoted to analyze the performance of the best variant when compared with the best previous method found in the state of the art.

All the experiments report the following metrics: Avg., the average objective function value; Time (s), the average computing time in seconds; Dev (%), the average deviation with respect to the best solution found in the experiment; and # Best, the number of times that an algorithm reaches the best solution of the experiment.

The preliminary experiments consider a subset of 20 representative instances out of 50 in order to avoid overfitting, while the final experimentation considers the total set of 50 instances.

The first experiment is designed for evaluating the effect of the  $\beta$  parameter in the quality of the initial solutions generated. We have considered the following

values for  $\beta = \{0.25, 0.50, 0.75, RND\}$ , where *RND* indicates that a random value is selected in each iteration of the construction phase. Table 1 shows the results obtained in this experiment. It is worth mentioning that 100 solutions have been generated for each instance, returning the best solution found and computing the accumulated time required for constructing all of them.

**Table 1.** Performance of GRASP constructive procedure with different values for the  $\beta$  parameter.

$\beta$	Avg.	Time (s)	Dev (%)	#Best
0.25	59.80	592.43	0.59	16
0.5	60.45	590.30	2.08	12
0.75	62.25	590.79	4.26	9
-1.00	59.85	612.49	0.94	13

Analyzing Table 1 we can clearly see that the best results are obtained when considering  $\beta = 0.25$ , closely followed by  $\beta = RND$ . Specifically,  $\beta = 0.25$  is able to find 16 out of 20 best solutions, and the average deviation of 0.59% indicates that in those instances in which it is not able to reach the best value, the constructive method obtains a high quality solution really close to the best one. The value of  $\beta = 0.25$  indicates that the best results are obtained when introducing a small random part in the constructive procedure, and increasing the randomness of the method results in worse solutions. The worst results are obtained with the largest  $\beta$  value, 0.75, obtaining just 9 out of 20 best solutions with an average deviation of 4.26%. This behavior also confirms that the betweenness metric is a good selection as a greedy function value for the constructive procedure.

One of the main advantages of VNS is the reduced number of parameters that must be tuned in order to obtain high quality solutions. In particular, the RVNS algorithm proposed requires from just one parameter,  $k_{\max}$ , that corresponds to the largest neighborhood to be explored. The preliminary experiment then considers the following values for  $k_{\max} = \{0.05, 0.10, 0.25, 0.50\}$ . We do not consider larger values of  $k_{\max}$  since a solution in such a large neighborhood will be completely different from the original one. It is worth mentioning that the number of vertices that will be included in the neighbor solution is evaluated as  $k \cdot |S|$ , being  $|S|$  the number of vertices of the initial solution. Table 2 shows the performance of the different values for  $k_{\max}$ .

The experiment clearly shows that the best value for  $k_{\max}$  is 0.25, finding all the best solutions, while the remaining values are able to obtain just one best solution out of 20. This results are in line with those presented by Mladenovic et al. [14], which recommends considering small values of  $k_{\max}$  in the context of RVNS. This is mainly because large values for  $k_{\max}$  explores solutions that are not close in the search space, resulting in a completely random search, which is against the VNS methodology.



**Table 2.** Performance of the RVNS when considering different values for the  $k_{\max}$  parameter

$k_{\max}$	Avg.	Time (s)	Dev (%)	#Best
0.05	49.00	303.09	15.40	1
0.10	48.40	314.81	14.16	1
0.25	43.45	310.06	0.00	20
0.50	47.65	316.44	12.76	1

Analyzing the results obtained in the preliminary experiments, the RVNS algorithm for the final experiment is configured with  $\beta = 0.25$  and  $k_{\max} = 0.25$ .

The final experiment is intended to compare the best variant of RVNS with the best previous method found in the state of the art [13]. Specifically, it consists of a random walk (RW) algorithm with Markov Chain Monte Carlo method. It is worth mentioning that we have not been able to contact to the authors of the previous work neither to obtain the set of instances nor an executable file of the algorithm. Therefore, we have reimplemented the previous algorithm following, in detail, all the steps described in the manuscript. Table 3 shows the results obtained by the proposed algorithm (RVNS) and the best previous method found (RW).

**Table 3.** Comparison of the best variant of RVNS with the best previous method found in the state of the art.

	Avg.	Time (s)	Dev (%)	#Best
RW	71.78	1070.35	25.98	5
RVNS	55.58	473.72	0.10	46

The results obtained clearly confirm the superiority of our proposal. In particular, RVNS is able to obtain 46 out of 50 best solutions, while RW only reaches the best solution in 5 out of 50 instances. Furthermore, the computing time for RVNS is half of the time required by RW. Finally, the average deviation of RVNS is close to zero, which indicates that, in those instances in which RVNS is not able to match the best solution, it stays close to it. However, the deviation of RW is higher, indicating that its results are not close to the best solution obtained by the RVNS.

We have finally conducted the nonparametric Wilcoxon Signed Test in order to confirm that there exists statistically significant differences between both algorithms. The  $p$ -value obtained is lower than 0.0001, which confirms the superiority of our proposal.

## 4 Conclusions

This work has proposed a Reduced VNS algorithm for detecting critical nodes in networks. The initial solution is generated by using a Greedy Randomized Adaptive Search Procedure whose greedy criterion is adapted from the social network field of research, which is named betweenness. The RVNS proposal is able to obtain better results than the best previous method found in the literature, which consists of a random walk algorithm in both quality and computing time. This results, supported by non-parametric statistical tests, confirms the superiority of the proposal. The adaptation of a social network metric to the problem under consideration in this work has led us to obtain high quality solutions, which reveals the relevance of the synergy among different fields of research.

## References

1. Andersson, G., et al.: Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance. *IEEE Trans. Power Syst.* **20**(4), 1922–1928 (2005)
2. Brandes, U.: A faster algorithm for betweenness centrality. *J. Math. Soc.* **25**(2), 163–177 (2001)
3. Crucitti, P., Latora, V., Marchiori, M.: Model for cascading failures in complex networks. *Phys. Rev. E* **69**, 045104 (2004)
4. Duarte, A., Sánchez-Oro, J., Resende, M.G., Glover, F., Martí, R.: Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. *Inf. Sci.* **296**, 46–60 (2015)
5. Erdős, P., Rényi, A.: On random graphs. *Publ. Math.* **6**, 290 (1959)
6. Feige, U., Mahdian, M.: Finding small balanced separators. In: Kleinberg, J.M. (ed.) *STOC*, pp. 375–384. ACM (2006)
7. Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8**(2), 67–71 (1989)
8. Feo, T.A., Resende, M.G., Smith, S.H.: Greedy randomized adaptive search procedure for maximum independent set. *Oper. Res.* **42**(5), 860–878 (1994)
9. Garey, M., Johnson, D.: *Computers and Intractability - A guide to the Theory of NP-Completeness*. Freeman, San Fransisco (1979)
10. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. *Eur. J. Oper. Res.* **130**(3), 449–467 (2001)
11. Hansen, P., Mladenović, N.: Variable neighborhood search. In: Burke, E., Kendall, G. (eds.) *Search Methodologies*, pp. 313–337. Springer, Boston (2014)
12. Hansen, P., Mladenović, N., Pérez, J.A.M.: Variable neighbourhood search: methods and applications. *Ann. Oper. Res.* **175**(1), 367–407 (2010)
13. Lee, J., Kwak, J., Lee, H.W., Shroff, N.B.: Finding minimum node separators: a Markov chain Monte Carlo method. In: *13th International Conference on Design of Reliable Communication Networks, DRCN 2017*, pp. 1–8, March 2017
14. Mladenovic, N., Petrovic, J., Kovacevic-Vujcic, V., Cangalovic, M.: Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *Eur. J. Oper. Res.* **151**(2), 389–399 (2003)
15. Mohamed-Sidi, M.: K-separator problem. (Problème de k-Séparateur). Ph.D. thesis, Telecom & Management SudParis, Évry, Essonne, France (2014)

16. Quintana, J.D., Sánchez-Oro, J., Duarte, A.: Efficient greedy randomized adaptive search procedure for the generalized regenerator location problem. *Int. J. Comput. Intell. Syst.* **9**(6), 1016–1027 (2016)
17. Wachs, M., Grothoff, C., Thurimella, R.: Partitioning the internet. In: Martinelli, F., Lanet, J.L., Fitzgerald, W.M., Foley, S.N. (eds.) *CRISIS*, pp. 1–8. IEEE Computer Society (2012)
18. Whitaker, R.: A fast algorithm for the greedy interchange for large-scale clustering and median location problems. *INFOR: Inf. Syst. Oper. Res.* **21**(2), 95–108 (1983)
19. Yen, J.Y.: Finding the k shortest loopless paths in a network. *Manag. Sci.* **17**(11), 712–716 (1971)