Chapter

A Survivable and Reliable Network Topological Design Model

Franco Robledo, Pablo Romero, Pablo Sartor, Luis Stabile and Omar Viera

Abstract

This work is focused on the resolution of a mixed model for the design of large-sized networks. An algorithm is introduced, whose initial outcomes are promising in terms of topological robustness regarding connectivity and reliability. The algorithm combines the network survivability and the network reliability approaches. The problem of the topological design has been modeled based on the generalized Steiner problem with node-connectivity constraints (GSPNC), which is NP-hard. The aim of this study is to heuristically solve the GSP-NC model by designing low-cost highly connected topologies and to measure the reliability of such solutions with respect to a certain prefixed lower threshold. This research introduces a greedy randomized algorithm for the construction of feasible solutions for the GSP-NC and a local search algorithm based on the variable neighborhood search (VNS) method, customized for the GSP-NC. In order to compute the built network reliabilities, this work adapts the recursive variance reduction (RVR) technique, as a simulation method since the exact evaluation of this measurement is also NP-hard. The experimental tests were performed over a wide set of testing cases, which contained heterogeneous topologies, including instances of more than 200 nodes. The computational results showed highly competitive execution times, achieving minimal local optimal solutions of good quality fulfilling the imposed survivability and reliability conditions.

Keywords: survivability, meta-heuristic, VNS, VND, reliability, simulation, RVR

1. Introduction

The arrival of the optical fiber allowed for an enormous increase on communication line bandwidth. This naturally led to deploying networks with dispersed topologies, as scarce (even unique) paths linking the diverse sites are enough to fulfill all the requirements regarding data exchanges. Yet dispersed topologies have a problem: the ability of the network to keep all sites connected is affected by the failure of few (even one single) communication links or switch sites. Before the introduction of the optical fiber, topologies were denser; upon failure of few links, there was a probable reduction of throughput, yet keeping all sites connected. With disperse fiber-based designs, the network behaves much more as an "all-or-nothing" service; either, it fulfills all requirements of connectivity and bandwidth, or it fails to connect some sites. Therefore, the problem of designing networks with minimal costs and reliability thresholds has since gained relevance.

In view of the above, the networks must continue to be operative even when a component (link or central office) fails. In this context, survivability means that a certain number of pre-established disjoint paths among any pair of central offices must exist. In this case, node-disjoint paths will be required, which show a stronger constraint than the edge-disjunction ones. Assuming that both the links and the nodes have associated certain operation probabilities (elementary reliability), the main objective is to build a minimum-cost sub-network that satisfies the node-connectivity requirements. Moreover, its reliability, i.e., the probability that all sites are able to exchange data at a given point in time, ought to surpass a certain lower bound pre-defined by the network engineer. In this way the model takes into account the robustness of the topology to be designed by acknowledging its structure even in probabilistic terms.

1.1 Aim and objectives

The aim of this chapter is to introduce an algorithm that combines the two approaches so as to achieve the design of robust networks and test it on a number of instances that are representative of communication network problems. On the one hand, the network must be highly reliable from a probabilistic point of view (its reliability) assuming that the probabilities of failure of all links and sites are known. On the other hand, the network structure must be topologically robust; for this, node or edge connectivity levels between pairs of distinguished nodes are required. This means that between all pairs of distinguished nodes there exists a given number of edge paths or disjoint nodes. Then, once a minimum threshold for reliability (e.g., 0.98) is set, the algorithm here introduced:

i. Constructs feasible low-cost solutions that satisfy the connectivity levels (disjoint paths) between pairs of distinguished nodes of the network (terminal nodes).

ii. The reliability of the network built meets the pre-established threshold, thus achieving fault-resistant networks according to both approaches.

Performed research indicates that literature pertaining to algorithms on design topologies which consider both approaches (survivable networks and network reliability) is scarce. The works on the design of robust networks in general fix a level of node/edge global connectivity of the network and try to design a network at the lowest possible cost that satisfies that level (e.g., 2-node-connectivity) [1]. Nevertheless, there are contexts where this combination of approaches is imperative and demanded. For example, in the context of military telecommunication networks, it is required that the networks are topologically very robust (e.g., 3-node-connectivity) and at the same time that they are extremely reliable from the network reliability approach's point of view, surpassing very high reliability levels. Another example of the application of the combined model is the logical distribution of highly dangerous merchandise on a country's roads. In such a context, two things are desirable: high reliability in the connection of points of distribution (i.e., "reliable" roads) and high levels of connectivity between the points that must exchange cargo (availability of alternative roads to possible road cuts, traffic saturation, etc.).

1.2 Problem definition

Formally, the proposed model that combines the network survivability and network reliability approaches is the following:

Consider:

- $G = (V,E)$ as a nondirected simple graph.
- $T \subseteq V$ as a subset of distinguished nodes (denominated terminals).
- $C = \{c_{ij}\}_{(i,j)\in E}$ as a nonnegative cost matrix associated with the edges of G.
- $R = \{r_{ij}\}_{i,j\in T}$ as a node-connectivity requirements matrix among pairs of terminal nodes in which at least r_{ij} node-disjoint paths are required to be communicating bearing i and j in the solution.

In addition to the above, suppose that the edges of E and the nodes of $V\backslash T$ (usually called Steiner nodes) have associated operation probabilities given by the vectors: $P_E = \{p_e\}_{e\in E}$ and $PV\backslash T = \{p_v\}_{v\in V\backslash T}$, respectively, where the failures are assumed to be statistically independent. Given a certain probability p_{min} set as reliability lower threshold, the objective is to find a subgraph $G_S \subseteq G$ of minimum cost that satisfies the node-connectivity requirement matrix R and furthermore its T-terminal reliability. The latter is defined as the probability that the partial graph $G_R \subseteq G_S$, obtained by randomly dropping edges and nodes from G_S with probabilities given by $1-p_e$ and $1-p_v$, respectively, connects all nodes in T [2]. The T-terminal reliability $R_T(G_S)$ has to satisfy $R_T(G_S) \geq p_{min}$ (i.e., the probability that all nodes in T are connected by working edges exceed p_{min}). This model will be referred to as "generalized Steiner problem with survivable and reliable constraints," GSP-SRC.

1.3 Chapter organization

The remainder of this chapter is structured as follows:

- Sections 2 and 3 present a background of the meta-heuristic algorithm applied and the means to compute the network reliability, respectively.
- Section 4 introduces the algorithm required to solve the GSP-SRC.
- Section 5 deals with the experimental results obtained over a set of heterogeneous test instances as well as the most important contributions and conclusions of this work.

2. Greedy randomized adaptive search procedure (GRASP)

Greedy randomized adaptive search procedure (GRASP) is a well-known meta-heuristic, i.e., a particular method to find sufficiently good solutions to optimization problems, that has been successfully used to solve many difficult combinatorial optimization problems. It is an iterative multi-start process which operates in two phases, namely, the construction and the local search phases.

In the construction phase, a feasible solution is built whose neighborhood is then explored in the local search phase [3]. A neighborhood of a certain solution S is a set of solutions that differ from S in well-defined forms (e.g., replacing any link by a different one, replacing "stars" by "triangles," and so on). Regarding optimization, different neighborhoods of S will not, in general, share the same local minimum. Thus, local optima trap problems may be overcome by deterministically changing the neighborhoods [4, 5].

Figure 1 illustrates a generic GRASP implementation pseudo-code. The GRASP takes as input parameters the following:

- The candidate list size.
- The maximum number of GRASP iterations.
- The seed for the random number generator. After reading the instance data (line 1), the GRASP iterations are carried out in lines 2–6. Each GRASP iteration consists of the construction phase (line 3), the local search phase (line 4), and, if necessary, the incumbent solution update (lines 5 and 6).

In the construction phase, a feasible solution is built, with one element at a time. At each step of the construction phase, a candidate list is determined by ordering all non-already selected elements with respect to a greedy function that measures the benefit of including them in the solution. The heuristic is adaptive because the benefits associated with every element are updated at each step to reflect the changes brought on by the selection of the previous elements. Then, one element is randomly chosen from the best candidate list and added into the solution. This is the probabilistic component of GRASP, which allows for different solutions to be obtained at each GRASP iteration but does not necessarily jeopardize the power of the adaptive greedy component.

The solutions generated by the construction phase are not guaranteed to be locally optimal with respect to simple neighborhood definitions. Hence, it is beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution from its neighborhood. It terminates when there is no better solution found in the neighborhood. The local search algorithm depends on the suitable choice of a neighborhood structure, efficient neighborhood search techniques, and the starting solution.

The construction phase plays an important role with respect to this last point, since it produces good starting solutions for local search. Normally, a local optimization procedure, such as a two-exchange, is employed. While such procedures can require exponential time from an arbitrary starting point, experience has shown that their efficiency significantly improves as the initial solutions improve. Through the use of customized data structures and careful implementation, an efficient construction phase that produces good initial solutions for efficient local search can be created. The result is that often many GRASP solutions are generated in the same amount of time required for the local optimization procedure to converge from a single random start. Furthermore, the best of these GRASP solutions is generally significantly better than the solution obtained from a random starting point.

```
Procedure GRASP(ListSize,MaxIter,RandomSeed) ;
1:  InputInstance();
2:  for k = 1 to MaxIter do
3:  InitialSolution = ConstructGreedyRandomizedSolution(ListSize,RandomSize);
4:  LocalSearchSolution = LocalSearch(InitialSolution);
5:  if cost(LocalSearchSolution) < cost(BestSolutionFound) then
6:  UpdateSolution(BestSolutionFound,LocalSearchSolution);
7:  end for;
8:  return BestSolutionFound;
```

Figure 1.
GRASP pseudo-code.

3. Recursive variance reduction technique

Recursive variance reduction (RVR) is a Monte Carlo simulation method for network reliability estimation [6]. It has shown excellent performance relative to other estimation methods, particularly when component failures are rare events.

It is a recursive method that works with probability measures conditioned to the operation or failure of specific cut-sets. A cut-set is a set of links (or nodes) such that the failure of any of its members results in a failure state for the overall network. RVR computes the unreliability (i.e., 1—reliability) of a network by finding a cut-set and recursively invoking itself several times, based on exhaustive and mutually exclusive combinations of up-and-down states for the members of the cut-set that cover the "cut-set fail" state space (i.e., a partition of the latter). While finding the cut-set and linking the recursion results introduce some overhead compared to other methods (e.g., crude Monte Carlo), RVR achieves significant reductions of the unreliability estimator variance, particularly in the (realistic) setting where failures are rare events. This allows for the use of smaller sample sizes, eventually beating the alternative methods in the trade-off between processing time and precision.

4. The algorithmic solution for the GSP-SRC

4.1 Network design algorithm

NetworkDesign is the main algorithm which iteratively executes the different phases that solve the GSP-SRC. The algorithm (shown in **Figure 2**) receives as entry *G* the original graph, *MaxIter* the number of iterations that is going to be executed, *k* an integer (parameter of the construction phase), the *threshold* of *T*-terminal reliability required, and the number of replications used in reliability phase.

```
Procedure NetworkDesign;
Input: G, C, k, MaxIter, threshold, simiter;
1:   i ← 0; P ←∅; L_Sol ←∅;
2:   while (i < MaxIter) do
3:   [G*sol,P] ← ConstructionPhase(G,C,k);
4:   Gsol ← SurvivabilityOptimizerPhase(G*sol,P);
5:   R*T(Gsol) ← RealiabilityPhase(Gsol,simiter);
6:   if (R*T(Gsol) ≥ threshold) then L_Sol ← Add(Gsol) ;
7:   else Discard(Gsol);
8:   i ← i + 1;
9:   end while;
10:  return L_Sol;
end NetworkDesign;
```

Figure 2.
Global algorithm.

Each iteration computes:

i. Construction phase

ii. Survivability optimizer phase

iii. Reliability phase

The construction phase takes G as the input and returns a topology satisfying the node-connectivities given by R. Since the solution built by the construction phase is not even a local optimum, in order to improve this solution, survivability optimizer phase searches for a local optimum solution by means of a variable neighborhood search (VNS) [7, 8] algorithm designed specifically for the GSP-NC. Finally, the reliability phase is computed evaluating the T-terminal reliability of the solution achieved in (*ii*). If it surpasses the prefixed threshold, then the local optimal solution is added into the collection *L_Sol*; otherwise, it is discarded. The algorithm returns a list *L_Sol* of feasible solutions that satisfy the pre-established survivability and reliability requirements.

4.2 Construction phase algorithm

The algorithm (shown in **Figure 3**) takes as input the graph G of feasible connections, the matrix of connection costs C, the matrix R of connection requirements between terminal nodes, and a parameter k. The current solution G_{sol} is initialized with the terminal nodes without any connection among them. An auxiliary matrix M is initialized with the values of R. This is used with the purpose of maintaining on each step the connection requirements not yet satisfied between nodes of T. The paths found on each iteration are stored in a data structure P. Iteratively, the construction phase searches for node-disjoint paths between terminal nodes of T that have not yet satisfied their connection requirements. The algorithm chooses on each iteration a pair of such terminal nodes $i,j \in T$. The current solution is updated by adding a new low-cost node-disjoint path between the chosen nodes. For this, an

```
Procedure ConstructionPhase;
Input: G, C, R, k;
1:  Gsol ← (T,∅); M ← R; Pij ←∅∀i,j ∈ T;
2:  while ∃mij > 0 do
3:  Let i,j ∈ T be two nodes that satisfy mij > 0;
4:  G* ← (G \Pij);
5:  C* ← C |G*\Gsol; /* C*uv = 0 if (u,v) ∈ Gsol */
6:  Lp  ← the k shortest paths from i to j on G*, considering C*;
7:  p ← SelectRandom(Lp);
8:  Gsol ←Gsol∪ {p}; Pij ←Pij∪ {p};
9:  M ← UpdateMatrix(Gsol,M,R,p);
10: end while;
11: return Gsol, P;
end ConstructionPhase
```

Figure 3.
Construction phase.

extension of the Takahashi-Matsuyama algorithm is employed in order to efficiently compute the k shortest node-disjoint paths from i to j (lines 3–9). These paths are stored in a restricted candidate list L_p. A path is randomly selected from L_p and incorporated into G_{sol}. This process is repeated until all the connection requirements have been satisfied; then, the feasible solution G_{sol} and the set of node-disjoint paths $P = \{P_{ij}\}_{i,j \in T}$ are returned.

4.3 Survivability optimizer phase: VNS algorithm for the GSP-SRC

The variable neighborhood search algorithm is sustained on the idea of systematically changing the neighborhood at the moment of performing the local search and requires therefore a finite set of different pre-defined neighborhoods.

VNS is based on three simple facts:

- A local minimum with respect to a neighborhood structure, which is not necessary to be a local minimum with respect to another one.
- A global minimum which is a local minimum with respect to all the possible neighborhood structures.
- In many problems the local minimum with respect to one or several neighborhood structures are relatively close.

In this work, the deterministic variant called VNS descent was used (variable neighborhood descent, VND). It consists of iteratively replacing the current solution with the local search result as long as improvements are verified. If a

```
Procedure VND_SurvivabiliyOptimizerPhase;
Input: Gsol, P, cls;
1:    cost ← cost(Gsol); kmax ← 2;
2:    Gsol ← SwapKeyPathLocalSearch(Gsol,P);
3:    cost ← cost(Gsol); notimprove ← 0;
4:    while (notimprove < kmax) do
5:    G*sol ← cls[kmax] → LocalSearch(Gsol,k);
6:    cost ← cost(Gsol); newcost ← cost(G*sol);
7:    if (newcost < cost) then
8:                cost ← newcost; notimprove ← 0;
9:                Gsol ← G*sol;
10:   else notimprove ← notimprove + 1;
11:   k ← (k + 1) mod kmax;
12:  end while;
13:  return Gsol;
end VND_SurvivabiliyOptimizerPhase;
```

Figure 4.
Survivability phase.

neighborhood structure change is performed in a deterministic way every time a local minimum is reached, the descent variable neighborhood search is obtained. The final solution given by the VND is a local minimum with respect to all the considered neighborhoods. Next, the VND customization for the GSPNC will be explained. Here, the VND uses three local searches: SwapKeyPathLocalSearch, KeyPathLocalSearch, and KeyTreeLocalSearch. Details on these local searches, their respective neighborhood structures, and the extension of the Takahashi-Matsuyama algorithm mentioned above can be found in [9].

The algorithm (shown in **Figure 4**) receives as input G_{sol} the initial solution graph, P the matrix of paths (both outputs of construction phase), and *cls* the set of local searches. Initially, the cost of G_{sol} is computed, and the local search SwapKeyPathLocalSearch is applied to it. SwapKeyPathLocalSearch uses P the path's matrix as input (lines 1–2). Since only this local search uses the P information, it is executed only at the beginning of the algorithm in a single time. However, its incorporation is fundamental for the purpose of achieving important improvements in the initial solutions generated by the algorithm of construction. Line 3 computes the new cost of G_{sol} and *notimprove* is initialized to zero. In cycles 4–12, the *k*th local search is performed to find a better solution (line 5) until no more improvements can be found by exploring the neighborhood set. If an improvement is achieved, the current cost is updated, *notimprove* is reset to zero, and the new solution G_{sol} is actualized (lines 8–9). In the case of not having improvements, *notimprove* is increased by one (line 10). Finally, regardless of the fact that improvements have been made,

Procedure RVR_ReliabilityPhase;

Input: G_{sol}, N;

1: seed ← GenerateSeed();

2: s ← 0; ss ← 0;

3: for l = 1 to N

4: G'_{sol} ←G_{sol};

/* RVR computes the anti-reliability */

5: x ← 1 – RVR(G'_{sol},P_E,$P_{V \setminus T}$);

6: s ← s + x;

7: ss ← ss + (x^2);

8: end for;

/* RT(Gsol) estimation */

9: R^*_T(Gsol) ← s/N;

10: variance ← (1/(N×(N–1))) (ss–(s^2/N));

11: return R^*_T(G_{sol}), variance;

end RVR_ReliabiliyPhase;

Figure 5.
Reliability phase.

the next neighborhood is explored in a circular form (line 11). In this way, and unlike the generic VND, when finding an improvement, the algorithm continues the search for new solutions in the following neighborhood instead of returning to neighborhoods already explored. Once the loop is finalized (lines 4–12), line 13 returns the best solution found by the VND.

4.4 Reliability phase: RVR algorithm for the GSP-SRC

The reliability of G_{sol} is estimated by using the RVR method. The construction of the method is done so as to obtain an estimation of the Q_K measurement (anti-reliability for a K set of terminals). As such, the RVR method is used in order to estimate the T-terminal reliability of G_{sol} assuming that the terminal nodes T are perfect (i.e., do not fail), and the edges as well as \the Steiner nodes (nodes of $V \setminus T$) have operation probabilities $P_E = \{p_e\}e_{\in}E$ and $PV \setminus T = \{p_v\}v_{\in}V \setminus T$, respectively.

Details and properties of this adaptation can be found in [10]. **Figure 5** shows the pseudo-code of the R_T (G_{sol}) estimated computation using the RVR method.

5. Computational results and discussion

To the best of the authors' knowledge, there are no efficient optimization algorithms that design robust networks combining the requirement of network reliability and high levels of topological connectivity between pairs of distinguished nodes (topological design of survivable networks). Critical applications such as military communication networks, transport and distribution of highly risky or critical products/substances, or circuit design with high requirements of redundancy (airplanes), among others, were influential factors that facilitated the combination of efficient network design with high levels of connectivity that exceeded a pre-established threshold of reliability. Given the lack/absence of real cases in the literature and plausible instances to be used as a benchmark for the combined problem GSP-SRC, 20 instances of the traveling salesman problem (TSP) from the TSPLIB library have been selected [11], and for each of these, three GSP-SRC instances have been generated by randomly marking 20, 35, and 50% out of the nodes as terminal. The 20 TSP test instances were chosen so that their topologies are heterogeneous and their numbers of nodes (which range from 48 to 225) cover typical applications of the GSP problem in telecommunication settings. The acronyms used for the 20 instances in the TSPLIB library are att48, berlin52, brazil58, ch150, d198, eil51, gr137, gr202, kroA100, kroA150, kroB100, kroB150, kroB200, lin105, pr152, rat195, st70, tsp225, u159, and rd100. The connectivity requirements were randomly set in $r_{ij} \in \{2,3,4\}$ and $\forall i,j \in T$.

It is worth noting that in all cases the best solutions attained by the VND algorithm were topologically minimal (i.e., feasibility is lost upon removal of any edge). The improvement percentage of the VND algorithm with respect to the solution cost delivered by the construction phase ranged from 25.25 to 39.84%, depending on the topological features of the instance, showing thus the potential of proposed VND in improving the quality of the starting solution.

For the instances in which the average T-terminal reliability of the *L_Sol* solutions set returned by NetworkDesign was computed, it widely surpassed the 85% prefixed threshold. Particularly, in those instances in which the operation probabilities of nodes and edges were set at 99 and 90%, respectively, the average T-terminal reliability was bounded by 86.0 and 96.7%. On the other hand, when setting the values of the operation probabilities of nodes and edges at 99 and 95%, respectively, the average T-terminal reliability was bounded by 99.1 and 99.6%. In all these evaluated cases, the average variance was small, lower than $1.0E{-}05$.

The average times per iteration reached by the NetworkDesign approximated algorithm were below 173 seconds in all test cases.

6. Conclusions

This chapter discussed the generalized Steiner problem with survivable and reliable constraints (GSP-SRC), combining the network survivability and network reliability approaches, to design large-size reliable networks. In addition, a heuristic algorithm was introduced in order to solve the GSP-SRC that combines the GRASP and VNS meta-heuristics (for design and optimization) and the RVR simulation technique for estimating the network reliability of the solutions built.

After testing the algorithm hereby introduced on 60 instances of the GSP-SRC problem, all solutions shared the following desirable facts:

- The solution attained by the VND algorithm was topologically minimal.
- The solution was significantly improved by the VND algorithm with respect to the solution built by the construction phase (25.25–39.84%).
- The prefixed threshold of T-terminal reliability was surpassed in all applicable cases with variances lower than 1.0E − 05.

Given the fact that the GSP-SRC is a NP-hard problem, it is the authors' view that the times per iteration are highly competitive (less than 173 seconds in the worst case).

Author details

Franco Robledo[1], Pablo Romero[1], Pablo Sartor[2*], Luis Stabile[1] and Omar Viera[1]

1 Facultad de Ingeniería, Instituto de Computación, Universidad de la República, Montevideo, Uruguay

2 Departamento de Operaciones, IEEM Business School, Universidad de Montevideo, Montevideo, Uruguay

*Address all correspondence to: psartor@um.edu.uy

References

[1] Mechthild S. Design of Survivable Networks. Lecture Notes in Mathematics. Verlag Berlin, Heidelberg: Springer; 1992;**1531**:1. DOI: 10.1007/BFb0088963. ISBN: 978-3-540-56271-9, ISBN: 978-3-540-47500-2. ISSN: 0075-8434

[2] Ball MO, Colbourn CJ, Provan SJ. Network Reliability. University of Maryland, Systems Research Center; 1992. https://drum.lib.umd.edu/bitstream/handle/1903/5255/TR_92-74.pdf?sequence=1&isAllowed=y

[3] Resende MG, Ribeiro CC. Optimization by GRASP—Greedy Randomized Adaptive Search Procedures, Computational Science and Engineering. New York: Springer-Verlag; 2016

[4] Duarte A, Mladenovic N, Sanchez-Oro J, Todosijevic R. Variable Neighborhood Descent. Cham: Springer International Publishing; 2016

[5] Salhi S. Handbook of metaheuristics (2nd ed.). Journal of the Operational Research Society. 2014;**65**(2):320-320

[6] Cancela H, El-Khadiri M. The recursive variance-reduction simulation algorithm for network reliability evaluation. IEEE Transactions on Reliability. 2003;**52**(2):207-212

[7] Mladenovic N, Hansen P. Variable neighborhood search. Computers & OR. 1997;**24**(11):1097-1100. DOI: 10.1016/S0305-0548(97)00031-2

[8] Hansen P, Mladenović N. Variable neighborhood search. In: Martí R, Pardalos P, Resende M. editors. Handbook of Heuristics; Springer, Cham. 2018. pp. 759-787. DOI: 10.1007/978-3-319-07124-419. Print ISBN: 978-3-319-07123-7. Online ISBN: 978-3-319-07124-4

[9] Robledo F. GRASP heuristics for wide area network design [PhD thesis]. Rennes, France: INRIA/IRISA, Université de Rennes I; 2005

[10] Laborde SSR, Rivoir A. Diseño de Topologías de Red Confiables, tesis f657, INCO, Facultad de Ingeniería, UdelaR, 2006 (Advisors: Robledo F, Viera O)

[11] Reinelt G. n.d. Available from: https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ [Accessed: 25 February 2019]