Variable Neighborhood Search for the Generalized Minimum Edge Biconnected Network Problem

Markus Leitner, *Favoritenstraße 9-11 / 1861, 1040 Vienna, Austria*
Bin Hu, *Favoritenstraße 9-11 / 1861, 1040 Vienna, Austria*
Günther R. Raidl, *Favoritenstraße 9-11 / 1861, 1040 Vienna, Austria**

Keywords: Network Design, Variable Neighborhood Search

1. Introduction

The Generalized Minimum Edge Biconnected Network Problem (GMEBCNP) is defined as follows. We consider an undirected weighted graph $G = \langle V, E, c\rangle$ with node set V, edge set E, and edge cost function $c : E \rightarrow \mathbb{R}^+$. Node set V is partitioned into r pairwise disjoint clusters $V_1, V_2, \ldots, V_r,\ \bigcup_{i=1}^{r} V_i = V,\ V_i \cap V_j = \emptyset\ \forall i, j = 1, \ldots, r,\ i \neq j$.

A solution to the GMEBCNP defined on G is a subgraph $S = \langle P, T\rangle,\ P = \{p_1, \ldots, p_r\} \subseteq V$ connecting exactly one node from each cluster, i.e. $p_i \in V_i,\ \forall i = 1, \ldots, r$, and containing no bridges [2, 7, 8], see Figure 1. A bridge is an edge which does not lie on any cycles and thus its removal would disconnect the graph. The costs of such an edge biconnected network are its total edge costs, i.e. $c(T) = \sum_{(u,v)\in T} c(u, v)$, and the objective is to identify a solution with minimum costs. This problem obviously is NP hard since already the task of finding a minimum cost biconnected network spanning all nodes of a given graph is NP hard.

The GMEBCNP arises in the design of backbones in large communication networks. While connecting local area networks by a global network, survivability by means of a single link outage can be covered via edge redundancy [2].

Despite the importance of this problem in real world, not much research has been done specifically for it until now. Huygens [7] studied the GMEBCNP and provided integer programming formulations along with facet-defining inequalities, but no results on actual instances were published. Leitner [8] provided a Variable Neighborhood Search (VNS) approach for the GMEBCNP in his thesis which is the basis of this article.

Variable Neighborhood Search, combined with Variable Neighborhood Descent (VND) as local improvement subordinate, is a metaheuristic which exploits systematically the idea of changing between different types of neighborhoods to head for superior local optima as well as a mechanism called shaking for reaching under-explored areas. For a more detailed description on VNS, see [5, 6].

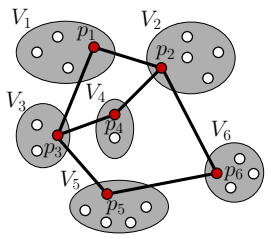

Figure 1: Example for a solution to the GMEBCNP.

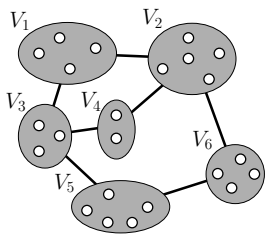

Figure 2: Example for a global solution.

Solution representation: For each solution, we store the spanned nodes $p_1, \ldots, p_r$ and the *global edges* derived from the so-called *global graph* $G^{\mathrm{g}} = \langle V^{\mathrm{g}}, E^{\mathrm{g}}\rangle$. This graph consists of nodes corresponding to

*This work is supported by the RTN ADONET under grant 504438.

clusters in G, i.e. $V^{\mathrm{g}} = \{V_1, V_2, \ldots, V_r\}$, and edge set $E^{\mathrm{g}} = \{(V_i, V_j) \mid \exists (u, v) \in E \wedge u \in V_i \wedge v \in V_j\}$. Global edges T^{g}, being a subset of E^{g}, define the connections between clusters in a *global solution* $S^{\mathrm{g}} = \langle V^{\mathrm{g}}, T^{\mathrm{g}} \rangle$, see Figure 2. Spanned nodes $p_1, \ldots, p_r$ alone are insufficient to represent a solution, as finding the cheapest edges for them is the classical minimum edge biconnected network problem, which is known to be NP hard. Similarly, a representation via global edges alone is also insufficient [8].

However, it is possible to efficiently determine the optimal selection of nodes for the majority of clusters once the spanned nodes are fixed in a few specific clusters. The underlying concept, called *graph reduction*, is based on the observation that good solutions to the GMEBCNP usually consist of only few clusters with spanned nodes of degree greater than two (*branching clusters*) and long paths of clusters with spanned nodes of degree two connecting them (*path clusters*). Once the spanned nodes within all branching clusters are fixed, it is possible to determine the optimal nodes for all remaining clusters in an efficient way by computing the shortest paths between the fixed nodes.

Formally, for any solution $S = \langle P, T \rangle$ and its corresponding global solution $S^{\mathrm{g}} = \langle V^{\mathrm{g}}, T^{\mathrm{g}} \rangle$, we can define a reduced global solution $S^{\mathrm{g}}_{\mathrm{red}} = \langle V^{\mathrm{g}}_{\mathrm{red}}, T^{\mathrm{g}}_{\mathrm{red}} \rangle$ with $V^{\mathrm{g}}_{\mathrm{red}} = \{V_i \in V^{\mathrm{g}} \mid \deg(V_i) \geq 3,\ \forall i = 1, \ldots, r\}$ and $T^{\mathrm{g}}_{\mathrm{red}} = \{(V_a, V_b) \mid \exists (V_a, V_{k_1}) \wedge (V_{k_2}, V_{k_3}) \wedge \cdots \wedge (V_{k_l}, V_b) \in T^{\mathrm{g}},\ \deg(V_a) \geq 3 \wedge \deg(V_b) \geq 3 \wedge \deg(V_{k_i}) = 2, \forall i = 1, \ldots, l\}$, see Figure 3.

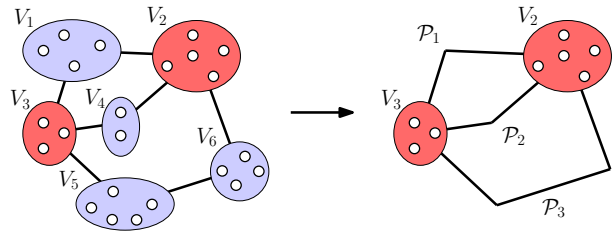

Figure 3: Example for graph reduction: Clusters V_2 and V_3 are branching clusters, while V_1, V_4, V_5, and V_6 can be reduced to paths $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$.

Initial solution: We start with a solution to the Generalized Minimum Spanning Tree Problem computed via *Improved Kruskal Heuristic* [4]. This algorithm considers edges in increasing cost-order and adds an edge to the solution iff it does not introduce a cycle and does not connect a second node of any cluster. By fixing an initial node to be in the resulting generalized spanning tree, different solutions can be obtained. Therefore, this process is carried out $|V|$ times, once for each node to be initially fixed, and the best spanning tree is adopted. Then, we determine all nodes with odd degree and match them with respect to edge costs in a greedy way. Unfortunately, this does not necessarily yield a solution satisfying the edge-biconnectivity property, so we additionally include the cheapest edges between biconnected components. At the end, we greedily remove *redundant edges* which might occur due to the previous step. Edges are redundant if they can be deleted without violating the edge biconnectivity property. This initialization algorithm, called *Adapted Christofides Heuristic* (ACH), is described in detail in [8].

2. The Neighborhood Structures

Our VND uses five different neighborhood structures, each of them focusing on different aspects of solutions to the GMEBCNP.

Node Optimization Neighborhood (NON): This neighborhood structure puts emphasis on the optimization of spanned nodes for a given set of global edges while considering the graph reduction described above. NON consists of all solutions S' that differ from S by at most two spanned nodes within branching clusters. Spanned nodes of path clusters are selected in an optimal way once the solution is decoded.

In order to efficiently evaluate neighbor solutions with exchanged spanned nodes in branching clusters, we precompute shortest paths between any pair of nodes of branching clusters that are connected via path clusters in S^{g}. Graph reduction including this additional precomputation has a worst case complexity of $O(r^2 \cdot d_{\max}^3)$, with $d_{\max}$ being the maximum number of nodes within a single cluster. As updating the objective value can be done in $O(d_{\max})$ the overall complexity of NON is $O(r^2 \cdot d_{\max}^3)$.

Node Re-Arrangement Neighborhood (NRAN): With this neighborhood structure we try to optimize a solution with respect to the arrangement of nodes. A neighbor solution in NRAN differs by exactly one swap move which exchanges for two nodes a and b their sets of incident nodes I_a and I_b. Set I_a, in respect to solution $S = \langle P, T \rangle$, is defined as $I_a = \{w \in P | (a, w) \in T\}$. After this swap move, $S' = \{P, T'\}$ consists of $T' = T \setminus I_a \setminus I_b \cup \{(a, v) | v \in I_b\} \cup \{(b, u) | u \in I_a\}$

Updating the objective value for a single move means to subtract the costs of the original edges and to add the costs of the new ones. Therefore, a complete evaluation of NRAN, which consists of all solutions S' differing from S by exactly one swap move, can be done in $O(r^2 \cdot d_{\max})$.

Cluster Re-Arrangement Neighborhood (CRAN): This neighborhood structure is an extension to NRAN which makes use of the concept of graph reduction. Moving from the current solution S to a neighbor solution S' in CRAN means swapping two nodes in an analogous way as for NRAN, then computing the reduced graph, and finally determining the best nodes in all path clusters. As applying the whole graph reduction after each move would be too expensive, an incremental update is done whenever two nodes of degree two are swapped, which will happen in the majority of cases. Whenever two nodes a, b of degree two, being part of the same path (i.e. $\exists$ path $\mathcal{P} = \{(a, v_1), (v_1, v_2), \ldots, (v_k, b)\}$ with $\deg(v_i) = 2 \; \forall i = 1, \ldots, k$) are swapped, only this path has to be updated. If a and b belong to different paths, both paths must be recomputed. However, if at least one of these nodes is of degree three or more, the graph reduction procedure needs to be re-applied as the structure of the whole solution may change.

The complexity of fully examining CRAN is $O(r^4 \cdot d_{\max}^3)$ when graph reduction is applied after every move. Since the complete evaluation might require too much time on larger instances, the exploration is aborted after a certain time limit, returning the so-far best neighbor instead of following a strict best neighbor strategy.

Edge Augmentation Neighborhood (EAN): In this neighborhood structure, modifications on the edges are primarily involved. Precisely, EAN of a solution $S = \langle P, T \rangle$ consists of all solutions S' reachable from S by adding a single edge $e \notin T$ and removing redundant edges. Edges are redundant if they can be deleted without violating the edge biconnectivity property. Obviously, removing e itself is not allowed since this would directly lead to the original solution S. We do not have to consider edges $e = (a, b)$ if $\deg(a) = \deg(b) = 2$ and a, b are part of the same path of nodes with degree two. In these cases, adding e obviously results in a graph where e would be the only redundant edge.

The time complexity for evaluating EAN is bounded by $O(r^5)$. However, since good solutions are typically rather sparse, we can omit the evaluation of many neighbor solutions due to the above observation.

Node Exchange Neighborhood (NEN): This neighborhood structure addresses both aspects, changing the spanned nodes as well as the edges connecting them. A neighbor solution in NEN may differ from the original solution by exactly one spanned node and an arbitrary number of edges. A single move within NEN is accomplished by first changing $p_i \in V_i$ to $p_i' \in V_i, p_i \neq p_i'$ and removing all edges incident to p_i. This leads to a graph consisting of at least two and at most $\deg(p_i) + 1$ components. We reconnect this graph by adding the cheapest edges between any pair of these components. Once this step is completed, edge biconnectivity is restored by adding an additional edge between any two edge biconnected components. Finally, redundant edges are removed.

The process of covering all bridges with additional edges can be expensive in practice. When disconnecting a node in a sparse graph, many bridges arise typically. Therefore, we first determine all nodes with degree one in the current solution and connect each of them with its cheapest partner. If only a single node with degree one exists, we connect it with the first reachable node of degree greater than two. This strategy helps to cover many bridges with only few edges. Remaining bridges are covered by simply adding the cheapest edges between any pair of edge biconnected components. Even with this clever bridge covering strategy, examining NEN still needs $O(|V| \cdot r^3)$. Therefore, analogous to CRAN, we stop the evaluation of NEN after a certain time limit and return the so-far best neighbor.

Arrangement of the Neighborhoods: In VND, we alternate between NON, NRAN, CRAN, EAN, and NEN in this order. This arrangement has been determined by taking the computational complexity of the neighborhoods, as well as concrete experiments into account.

Shaking: Most of our neighborhood structures focus on the optimization of the spanned nodes instead of the edges between them. In order to enhance diversity, our shaking procedure is based on EAN. It starts by augmenting a current solution by a single edge and increases the amount of augmented edges up to $\lfloor \frac{r}{4} \rfloor$. Similar to EAN, redundant edges are removed after augmenting a solution.

3. Computational Results

We tested our algorithm on a benchmark set consisting of instances introduced by Ghosh [3] for the Generalized Minimum Spanning Tree Problem, along with larger instances that have been created in an analogous manner, as well as on most of the larger TSPlib[1] based instances with geographical center clustering [1]. Since no previous computational results of other approaches for the GMEBCNP are available, we compare the objective values of the final solutions of our VNS approach with the initial values computed by ACH in Table 1 and 2. In order to compute mean values as well as standard deviations, all tests have been repeated 30 times. The used machine was a Pentium 4, 2.8 GHz PC with 2GB RAM.

Table 1: Results on TSPlib instances with geographical clustering, $\frac{|V|}{r} = 5$.

TSPlib Instances				ACH		VNS	
Name	$\|V\|$	r	time	$\overline{C(T)}$	std dev	$\overline{C(T)}$	std dev
gr137	137	28	150s	562.0	0.00	442.2	2.94
kroa150	150	30	150s	17234.0	0.00	11542.7	24.74
krob200	200	40	300s	17779.0	0.00	13300.9	102.37
ts225	225	45	300s	83729.0	0.00	69110.0	641.85
gil262	262	53	300s	1434.0	0.00	1117.2	30.70
pr264	264	54	300s	39860.0	0.00	31641.9	1027.65
pr299	299	60	450s	28684.0	0.00	23397.0	321.22
lin318	318	64	450s	28039.0	0.00	22599.6	780.27
rd400	400	80	600s	9605.0	0.00	7291.1	276.53
fl417	417	84	600s	12177.0	0.00	10875.7	196.74
gr431	431	87	600s	1681.0	0.00	1399.6	45.02
pr439	439	88	600s	86968.0	0.00	73193.7	2941.80
pcb442	442	89	600s	29573.0	0.00	25960.8	621.32

To perform a more detailed analysis on the contributions of all neighborhood structures to the overall search process, we additionally logged how often each of them could improve a solution in relation to how often

[1] http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsplib.html

Table 2: Results on instance sets from Ghosh [3], 600s CPU-time for VNS. Three different instances are considered for each set.

Instances				ACH		VNS	
Set	$\|V\|$	r	$\|V\|/r$	$\overline{C(T)}$	std dev	$\overline{C(T)}$	std dev
Grouped Eucl 125	125	25	5	227.1	0.00	159.7	0.33
	125	25	5	209.5	0.00	163.5	0.00
	125	25	5	230.9	0.00	166.1	0.00
Grouped Eucl 500	500	100	5	939.6	0.00	712.2	16.33
	500	100	5	993.6	0.00	736.7	33.86
	500	100	5	943.7	0.00	751.5	28.87
Grouped Eucl 600	600	20	30	172.6.	0.00	105.8	2.66
	600	20	30	151.0.	0.00	105.3	0.07
	600	20	30	179.0	0.00	107.5	0.00
Grouped Eucl 1280	1280	64	8	590.2	0.00	402.2	22.90
	1280	64	8	585.4	0.00	399.9	14.22
	1280	64	8	562.5	0.00	417.0	15.83
Random Eucl 250	250	50	5	4398.9	0.00	3521.8	108.83
	250	50	5	5110.0	0.00	3227.5	222.37
	250	50	5	4975.1	0.00	3015.2	139.05
Random Eucl 400	400	20	20	3237.8.	0.00	906.8	61.17
	400	20	20	2582.8	0.00	867.4	40.78
	400	20	20	2308.6	0.00	802.2	18.18
Random Eucl 600	600	20	30	2984.3	0.00	602.6	4.03
	600	20	30	2964.1	0.00	785.4	39.64
	600	20	30	2550.8	0.00	759.2	32.63
Non-Eucl 200	200	20	10	1569.7	5.73	237.5	29.33
	200	20	10	1223.9	0.00	217.0	22.00
	200	20	10	1465.6	0.00	195.6	32.20
Non-Eucl 500	500	100	5	2045.9	1.72	1049.0	114.92
	500	100	5	2073.6	146.39	998.1	94.98
	500	100	5	1565.0	0.00	1020.4	78.15
Non-Eucl 600	600	20	30	1469.6	0.00	122.7	12.43
	600	20	30	1754.6	0.00	118.0	16.04
	600	20	30	414.3	0.00	117.9	16.54

they were evaluated. Table 3 shows contribution values grouped by the different instance types. They are retrieved by first calculating the success rate of each neighborhood structure and then scaling these values s.t. they sum up to 1 for each instance type. While all neighborhood structures contribute substantially to the search process, we observe that CRAN performs best overall. Some neighborhood structures seem to work well only on particular instance types, but NON's performance is relatively steady. We conclude that the concept of graph reduction, which is used by CRAN and NON, is a very efficient technique when designing neighborhood structures for GMEBCNP.

4. Conclusions and Future Work

In this article, we introduced five neighborhood structures for the Generalized Minimum Edge Biconnected Network Problem. Each of them addresses particular properties as spanned nodes and/or the edges between them. For the more complex neighborhood structures, we apply techniques to optimize the search process, which consist of clever evaluation strategies, as well as methods to omit senseless computation.

We implemented a Variable Neighborhood Search algorithm with Variable Neighborhood Descent using all five neighborhood structures as local improvement subordinate. Tests were performed on TSPlib instances

Table 3: Relative contributions of NON, NRAN, CRAN, EAN, and NEN.

Instance Type	$\|V\|$	r	$\|V\|/r$	NON	NRAN	CRAN	EAN	NEN
TSPlib based	n.a.	n.a.	5	0.11	0.18	0.37	0.23	0.11
Grouped Euclidean	125	25	5	0.29	0.14	0.40	0.12	0.05
	500	100	5	0.15	0.18	0.28	0.22	0.16
	600	20	30	0.21	0.12	0.49	0.11	0.06
	1280	64	20	0.21	0.14	0.29	0.16	0.20
Random Euclidean	250	50	5	0.11	0.26	0.34	0.19	0.09
	400	20	20	0.20	0.21	0.44	0.10	0.04
	600	20	30	0.14	0.24	0.46	0.11	0.04
Non-Euclidean	200	20	10	0.24	0.03	0.45	0.16	0.12
	500	100	5	0.13	0.05	0.27	0.33	0.22
	600	20	30	0.26	0.02	0.55	0.09	0.09

with geographical center clustering, Euclidean instances with grid clustering, Euclidean instances with random clustering, and non-Euclidean instances. Results show that the combination of these neighborhood structures works well and each of them contributes significantly to the whole success.

In future, we want to investigate additional large neighborhood structures that are evaluated by means of Integer Linear Programming. Another idea is to extend the graph reduction concept by shrinking the graph even further s.t. fewer spanned nodes have to be fixed, and more of them can be efficiently derived in an optimal way.

References

[1] Corinne Feremans. *Generalized Spanning Trees and Extensions*. PhD thesis, Universite Libre de Bruxelles, 2001.

[2] Corinne Feremans, Martine Labbe, and Gilbert Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1–13, 2003.

[3] Diptesh Ghosh. Solving medium to large sized Euclidean generalized minimum spanning tree problems. Technical Report NEP-CMP-2003-09-28, Indian Institute of Management, Research and Publication Department, Ahmedabad, India, 2003.

[4] Bruce Golden, S. Raghavan, and Daliborka Stanojevic. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304, 2005.

[5] Pierre Hansen and Nenad Mladenovic. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-heuristics, Advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer Academic Publishers, 1999.

[6] Pierre Hansen and Nenad Mladenovic. A tutorial on variable neighborhood search. Technical Report G-2003-46, Les Cahiers du GERAD, HEC Montreal and GERAD, Canada, 2003.

[7] David Huygens. Version generalisee du probleme de conception de reseau 2-arete-connexe. Master's thesis, Universite Libre de Bruxelles, 2002.

[8] Markus Leitner. Solving two generalized network design problems with exact and heuristic methods. Master's thesis, Vienna University of Technology, 2006.