A Memetic Algorithm for the Generalized Minimum Vertex-Biconnected Network Problem

Bin Hu
Vienna University of Technology
Favoritenstraße 9–11/1861
1040 Vienna, Austria
hu@ads.tuwien.ac.at

Günther R. Raidl
Vienna University of Technology
Favoritenstraße 9–11/1861
1040 Vienna, Austria
raidl@ads.tuwien.ac.at

***Abstract*—The generalized minimum vertex-biconnected network problem plays an important role in the design of survivable backbone networks that should be fault tolerant to single component outage. When given a graph where the nodes are partitioned into clusters, the goal is to find a subgraph of minimum costs that connects exactly one node from each cluster in a vertex-biconnected way. We present a memetic algorithm that uses fast local improvement methods to produce high quality solutions and an intelligent crossover operator which controls the balance between diversity and intensity in the population. Tests on Euclidean TSPlib instances with up to 442 nodes show that this approach is highly efficient.**

***Index Terms*—Network Design; Biconnectivity; Memetic Algorithm;**

I. Introduction

The *Generalized Minimum Vertex-Biconnected Network Problem* (GMVBCNP) is defined as follows. We consider a complete, 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 GMVBCNP 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 the removal of any single node $v \in P$ along with all its incident edges would not disconnect S. An example is given in Figure 1. 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 feasible solution with minimum costs.

In practice, when designing large communication networks, devices belonging to the same local area network can be modelled as nodes within the same cluster. A backbone is required to connect one device per local network that can be regarded as its supernode. When additionally requiring fault tolerance by means of *edge-biconnectivity*, the network is able to survive single connection outages. However, if a supernode that is a so-called *cut point* fails, the network still breaks into multiple disconnected components. With the *vertex-biconnectivity* property, this issue is finally covered as well.

For approaching the GMVBCNP, we suggest a *Memetic Algorithm* (MA) [1] that uses problem specific variation operators and local improvement procedures in order to enhance the solution quality. MAs can be seen as hybrid techniques that combine the population concept of *Evolutionary Algorithms* (EA) with intensification mechanisms that exploit the specific knowledge on the actual problem. Therefore MAs show great capabilities in inheriting the benefits of EAs while compensating some of their weaknesses in order to find high quality solutions to difficult optimization problems.

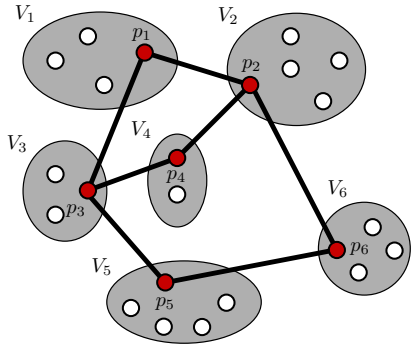

Fig. 1. Example for a solution to the GMVBCNP.

To the best of our knowledge, the GMVBCNP has not been addressed in the literature yet. However, there are works that study the strongly related *Generalized Minimum Edge Biconnected Network Problem* (GMEBCNP) which only requires edge-biconnectivity. The GMEBCNP has been introduced by Huygens [2]. He proposed integer programming formulations, but no practical results on actual instances were published. Leitner et al. [3], [4] presented Variable Neighborhood Search (VNS) approaches for the GMEBCNP based on several types of neighborhood structures which augment each other well. Some of these neighborhoods use enhanced techniques to reduce the search space. They can be adapted and applied on the current problem as well.

When neglecting the generalization, we get the classical problem of finding a minimum-cost vertex-biconnected network on a given graph which was introduced by Eswaran et al. [5]. They showed that this problem is NP-hard and introduced the so-called *block-cut graph* that allows efficient detection of cut point. This method will be used in this work as well.

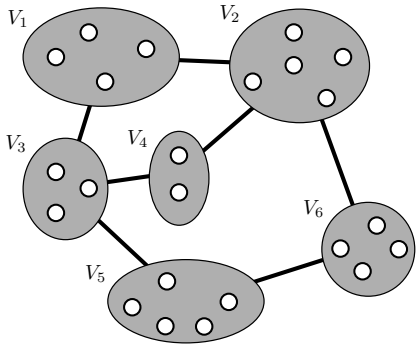

Fig. 2. Global structure corresponding to solution in Figure 1.

II. Memetic Algorithm for the GMVBCNP

In this section, we will first describe the solution representation and the initialization procedure, the mutation and crossover operators for the genetic algorithm, then discuss the local improvement methods and the techniques to optimize the search process. We use the general MA framework illustrated in Algorithm 1. Note that we apply mutation before crossover because our mutation operator potentially decreases the solution quality, but the crossover operator will fix this issue again. For selection, the standard tournament selection with tournament size of two is used.

Algorithm 1: Memetic Algorithm for GMVBCNP

```
create random initial population S of feasible solutions
repeat
    Select two parent solutions S_1, S_2 ∈ S
    With probability p_mut apply mutation on S_1, S_2
    Apply crossover on S_1, S_2 to obtain S_new
    With probability p_imp locally improve S_new
until no new best solution is found for l iterations
```

A. Terminology

For solution representation, graph reduction, and neighborhood structures, we use following terminology.

The *global graph*, denoted by $G^{\mathrm{g}} = \langle V^{\mathrm{g}}, E^{\mathrm{g}} \rangle$, consists of nodes corresponding to clusters in G, i.e., $V^{\mathrm{g}} = \{V_1, V_2, \ldots, V_r\}$, and the complete edge set $E^{\mathrm{g}} = \{(V_i, V_j) \mid V_i, V_j \in V^{\mathrm{g}},\ V_i \neq V_j\}$. Hereby, each *global connection* (V_i, V_j) represents all edges $\{(u, v) \in E \mid u \in V_i \wedge v \in V_j\}$ of graph G.

When given some feasible candidate solution $S = \langle P, T \rangle$ to the GMVBCNP, its corresponding *global structure* is defined as the induced global graph's subgraph $S^{\mathrm{g}} = \langle V^{\mathrm{g}}, T^{\mathrm{g}} \rangle$ with the global connections $T^{\mathrm{g}} = \{(V_i, V_j) \in E^{\mathrm{g}} \mid \exists (u, v) \in T \wedge u \in V_i \wedge v \in V_j\}$. Figure 2 illustrates a global structure of the solution in Figure 1.

Redundant edges of a candidate solution are edges that can be removed without violating the edge-biconnectivity property. Redundant global connections are defined analogously for global structures. Finally, an *edge-minimal solution* is a solution that contains no redundant edges.

B. Solution Representation and Initialization

Each solution is characterized by the spanned nodes $P = \{p_1, \ldots, p_r\}$, stored in an vector of length r, and the global connections T^{g}, stored as an adjacent list that represents a global structure such as the one shown in Figure 2.

The spanned nodes $p_1, \ldots, p_r$ alone are insufficient to represent a solution since finding the cheapest edges for them corresponds to the classical minimum vertex-biconnected network problem. Similarly, a representation via global connections alone is also insufficient since identifying a set of optimal nodes when restricted to a given global structure is also NP-hard. The latter is shown in [4] for the GMEBCNP. Because vertex-biconnectivity implies edge-biconnectivity, this also holds for the current problem.

The procedure of creating initial solutions is inspired by the fact that simple Hamiltonian cycles represent feasible solutions for the GMEBCNP and they can be generated quite fast. In detail we first fix the set of spanned nodes by randomly selecting $p_i \in V_i,\ \forall i = 1, \ldots, r$. Then all these nodes are connected in a random order. Obviously the initial solutions are not very good. However, in our experiments the quality of initial solutions only has a low impact on the finally best solutions found by the MA.

C. Mutation Operator

We use two simple mutation operators to recover possibly lost features in the population:

1) Change the spanned node of a random cluster to a new one.
2) Add a global connection to the current solution between two random clusters.

Note that the second procedure generates solutions that are not edge-minimal. However, the crossover operator that directly follows afterwards removes redundant edges again. So this mutation operator only creates an additional choice by means of a new connection possibility. Each operator is applied with a probability of $p_{\mathrm{mut}} = 0.05$.

D. Crossover Operator

The crossover operator generates a new solution $S_{\mathrm{new}} = \langle P_{\mathrm{new}}, T_{\mathrm{new}} \rangle$ by inheriting the properties of two parent solutions $S_1 = \langle P_1, T_1 \rangle$ and $S_2 = \langle P_2, T_2 \rangle$. For P_{new}, we apply classical uniform crossover, i.e., the spanned node of each cluster is randomly chosen from either P_1 or P_2. For generating the global connections T_{new}, we first merge all global connections of both parent solutions, i.e., $T_{\mathrm{new}} = T_1 \cup T_2$. Because the spanned nodes P_{new} are already fixed, each of these global connections $(V_i, V_j) \in T_{\mathrm{new}}$ consists of only one edge $(p_i, p_j) \in E,\ p_i \in V_i \wedge p_j \in V_j \wedge p_i, p_j \in P_{\mathrm{new}}$. Then we remove redundant edges until the new solution is edge-minimal. Edges $(p_i, p_j) \in T_{\mathrm{new}}$ where $\deg(p_i) = 2 \vee \deg(p_j) = 2$ need not to be considered during this process where $\deg(v)$ denotes the degree of node v. Since we know

the actual costs of these edges, we put them into a particular order according to

- α: decreasing costs,
- β: decreasing perturbed costs $c'(p_i, p_j) = c(p_i, p_j) \cdot \varphi$ where φ is a uniformly distributed random value in $[0.5 \ldots 1.5]$, or
- γ: random order.

Obviously, strategy α generates offsprings of superior qualities and thus emphasizes intensification whereas γ favors diversification. Each time the crossover operator is used, we select the sorting criterion for edge removal randomly with probability $(p_\alpha, p_\beta, p_\gamma)$. Initially, they are set to $(p_\alpha, p_\beta, p_\gamma) = (0.5, 0.3, 0.2)$. During the optimization process, these values are adapted according to the diversity in the population: For every 100 iterations, we count how many different spanned nodes and how many different global connections appear in all solutions in the current population. If the percentage of the lost attributes exceed a certain limit, we increase the mutation rate p_{mut} by 0.02 and change the probabilities of the crossover strategies to $(p_\alpha, p_\beta, p_\gamma) = (0.2, 0.2, 0.6)$. On the other hand, when diversity reaches a certain limit, we set these parameters back to the initial values.

E. Local Improvement

With probability of $p_{\text{imp}} = 0.2$ we apply local improvement on solutions generated by the crossover operator. The MA alternates between two neighborhood structures until no further improvements can be achieved. These neighborhood structures are adapted from the previous work on the GMEBCNP [3], [4]. For both of them, we use the so-called *graph reduction* technique to optimize the search process in a particular neighborhood.

a) Graph Reduction: The graph reduction technique has been introduced and successfully applied to the GMEBCNP in [3], [4]. The motivation is to reduce the search space for some neighborhood structures on which the local improvement procedures are based on.

As mentioned in Section II-B it is generally not possible to derive an optimal selection of spanned nodes in polynomial time when a global structure S^{g} is given. However, this task becomes feasible once the spanned nodes in a few specific clusters are fixed. Based on the global structure, we distinguish between *branching clusters* that have a degree greater than two, and *path clusters* that have a degree of two. Note that there are no clusters with degree one, since this would violate the biconnectivity constraint.

Once the spanned nodes within all branching clusters are fixed, it is possible to efficiently determine optimal selection of nodes for the path clusters: By computing the shortest path between two nodes of branching clusters which are connected by path clusters, optimal spanned nodes can be obtained.

Formally, for any global structure $S^{\text{g}} = \langle V^{\text{g}}, T^{\text{g}} \rangle$, we define a *reduced global structure* $S^{\text{g}}_{\text{red}} = \langle V^{\text{g}}_{\text{red}}, T^{\text{g}}_{\text{red}} \rangle$. V_{red} denotes the set of branching clusters, i.e., $V^{\text{g}}_{\text{red}} = \{V_i \in V^{\text{g}} \mid \deg(V_i) \geq 3\}$. $T^{\text{g}}_{\text{red}}$ consists of edges which represent sequences of path clusters connecting these branching clusters, i.e., $T^{\text{g}}_{\text{red}} = \{(V_a, V_b) \mid (V_a, V_{k_1}), (V_{k_1}, V_{k_2}), \ldots, (V_{k_{l-1}}, V_{k_l}), (V_{k_l}, V_b) \in T^{\text{g}} \wedge V_a, V_b \in V^{\text{g}}_{\text{red}} \wedge V_{k_i} \notin V^{\text{g}}_{\text{red}}, \ \forall i = 1, \ldots, l\}$. Note that $S^{\text{g}}_{\text{red}}$ is in general a multi-graph that can contain multiple edges corresponding to multiple paths in S^{g} between two nodes. Figure 3 shows an example for applying graph reduction on the global structure S^{g} of Figure 2. V_2 and V_3 are branching clusters while all others are path clusters.

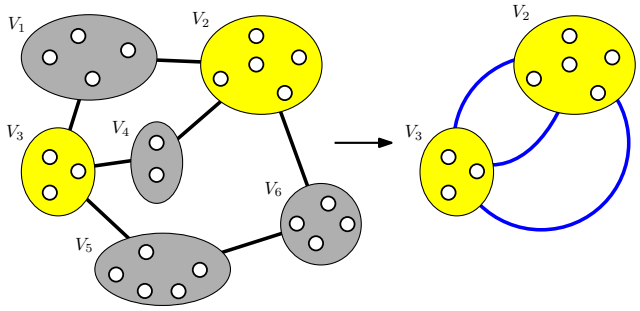

Fig. 3. Example for applying graph reduction on a global structure.

Corresponding to the reduced global structure $S^{\text{g}}_{\text{red}} = \langle V^{\text{g}}_{\text{red}}, T^{\text{g}}_{\text{red}} \rangle$ we can define a *reduced graph* $G_{\text{red}} = \langle V_{\text{red}}, E_{\text{red}} \rangle$ with the nodes representing all branching clusters $V_{\text{red}} = \{v \in V_i \mid V_i \in V^{\text{g}}_{\text{red}}\}$ and edges between any pair of nodes whose clusters are adjacent in the reduced global structure, i.e., $(i, j) \in E_{\text{red}} \Leftrightarrow (V_i, V_j) \in T^{\text{g}}_{\text{red}}, \forall i \in V_i, j \in V_j$. Each such edge (i, j) corresponds to the shortest path connecting i and j in the subgraph of G represented by the reduced structure's edge (V_i, V_j), and (i, j) therefore gets assigned this shortest path's costs.

When fixing the spanned nodes in $V^{\text{g}}_{\text{red}}$ we can determine the costs of the corresponding solution S with optimally chosen nodes in path clusters efficiently by using the precomputed shortest path costs stored with the reduced graph's edges. Decoding the corresponding solution, i.e., making the optimal spanned nodes within path clusters explicit, is done by choosing all nodes lying at the shortest paths corresponding to used edges from E_{red}.

For details on how the graph reduction can be efficiently implemented, we refer to [6]. As a matter of fact, all solutions considered for local improvement are edge-minimal since they are derived from the crossover operator. Hence they consist of $O(r)$ edges only. The overall time complexity is bounded by $O(r \cdot d^3_{\max})$, with $d_{\max}$ being the maximum number of nodes within a single cluster.

Node Optimization Neighborhood: This neighborhood structure emphasizes the selection of the spanned nodes in the branching clusters while not modifying the global structure. When V_{red} is the set of branching clusters in a current solution S, the *Node Optimization Neighborhood* (NON) consists of all solutions S' that differ from S by exactly one spanned node of a branching cluster. A move within NON is accomplished by changing $p_i \in V_i \in V_{\text{red}}$ to $p'_i \in V_i, p_i \neq p'_i, \ i \in \{1, \ldots, r\}$. By using the graph reduction technique, spanned nodes of path

clusters are computed in an optimal way. Algorithm 2 shows NON in detail.

Algorithm 2: Node Optimization (solution S)

```
compute reduced structure S^g_red = ⟨V^g_red, T^g_red⟩
forall V_i, V_j ∈ V^g_red ∧ V_i ≠ V_j do
    forall u ∈ V_i ≠ p_i do
        change used node p_i of cluster V_i to u
        forall v ∈ V_j do
            change used node p_j of cluster V_j to v
            if current solution better than best then
                save current solution as best
            restore initial solution
restore and return best solution
```

Updating the objective value for a considered neighbor can be done in $O(d_{\max})$ and $O(r)$ neighbors are to be considered. However, Applying graph reduction in advance adds $O(r \cdot d_{\max}^3)$ to the time complexity. This is also the overall time complexity of NON.

If V_{red} is empty, then the global structure is a round trip and all spanned nodes can be determined by using the shortest path calculation analogously to the generalized traveling salesman problem [7], [8] and NON is not searched at all.

Cluster Re-Arrangement Neighborhood: With this neighborhood structure we try to optimize a solution with respect to the arrangement of the clusters. Given a solution S with its global structure $S^{\text{g}} = \langle V^{\text{g}}, T^{\text{g}} \rangle$, let $\text{adj}(V_a)$ and $\text{adj}(V_b)$ be the sets of adjacent clusters of V_a and V_b in S^{g}, respectively. Moving from S to a neighbor solution S' in the *Cluster Re-Arrangement Neighborhood* (CRAN) means to swap these sets of adjacent clusters, resulting in $\text{adj}(V_a') = \text{adj}(V_b)$ and $\text{adj}(V_b') = \text{adj}(V_a)$ with V_a' and V_b' being the clusters in S' corresponding to V_a and V_b in S, respectively. S' can be further improved by using the shortest path calculations to rechoose the spanned nodes in the path clusters. Since doing this after each move is relatively time-expensive, we use graph reduction again to enhance the performance. Whenever the arrangement of two path clusters is swapped, it is possible to only apply incremental updates on the paths that contain them. However, if at least one of these cluster is a branching cluster, the graph reduction procedure must be completely re-applied as the structure of the whole solution graph may change. The pseudocode is given in Algorithm 3.

The worst case time complexity of completely examining CRAN is $O(r^3 \cdot d_{\max}^3)$ when graph reduction is applied after every move. In practice however, there are much more path clusters than branching cluster in an actual global structure, so the evaluation time is significantly lower.

III. Computational Results

We tested our algorithms on Euclidean TSPlib[1] instances with geographical center clustering [9], [10]. They contain 137 to 442 nodes partitioned into 28 to 89 clusters. The number of nodes per cluster varies.

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

Algorithm 3: Cluster Re-Arrangement (solution S)

```
compute reduced structure S^g_red = ⟨V^g_red, T^g_red⟩
for i = 1, ..., r − 1 do
    for j = i + 1, ..., r do
        swap adjacency lists of nodes p_i and p_j
        if V_i or V_j is a branching cluster then
            recompute reduced solution
            S^g_red = ⟨V^g_red, T^g_red⟩
        else
            if
            V_i and V_j belong to the same reduced path P
            then
                update P in S^g_red
            else
                update the path containing V_i in S^g_red
                update the path containing V_j in S^g_red
        if current solution better than best then
            decode and save current solution as best
        restore initial solution and S^g_red
restore and return best solution
```

All experiments have been performed on a Intel Core 2 Quad, 2.4 GHz PC with 4GB RAM. The program is written in C++ and gcc version 4.1.3 is used. In order to compute average values and standard deviations, 30 independent runs have been performed for each instance. The population of the MA consists of 100 solutions.

Because GMEBCNP and GMVBCNP are strongly related, we compare the results of our MA with the results obtained by the self-adapting VNS for the GMEBCNP [4] in Table I. For the VNS, two different stopping criteria were tested, a long run version with fixed CPU time limit and a short run version with $l = 30$ iterations without improvement. Each run of MA for the GMVBCNP was terminated after $l = 200$ iterations without improvement. For each algorithm, the table shows the objective values of the best solutions found during 30 runs $C(T_{\text{best}})$, the average values of the final solutions $\overline{C(T)}$, their standard deviations, and the absolute CPU time limits of the VNS or the average CPU times $\overline{\text{time}}$ required when the other stopping criterion was used.

In order to get a better insight on the complexity of the GMVBCNP, we also tested a *Mixed Integer Programming* (MIP) model adapted from the GMEBCNP [4] by further adding constraints to guarantee vertex-biconnectivity. Based on this model, the general purpose MIP solver CPLEX in version 11.2 was able to provide optimal solutions within reasonable time for small instances with up to 80 nodes in 16 clusters. However, the limit was quickly reached when the instances get larger. With a CPU time limit of one hour, CPLEX was only able to provide integer feasible solutions for instances gr137 and kroa150. Up to instance lin318, it could

TABLE I
RESULTS ON TSPLIB INSTANCES WITH GEOGRAPHICAL CLUSTERING.

Instance				GMEBCNP, VNS with time limit				GMEBCNP, VNS with l=30				GMVBCNP, MA with l=200			
Name	$\|V\|$	r	LP bound	$C(T_{\text{best}})$	$\overline{C(T)}$	std dev	time	$C(T_{\text{best}})$	$\overline{C(T)}$	std dev	$\overline{\text{time}}$	$C(T_{\text{best}})$	$\overline{C(T)}$	std dev	$\overline{\text{time}}$
gr137	137	28	384.5	440.0	440.0	0.0	150s	444.0	464.0	14.7	0.9s	440.0	446.0	6.9	2.2s
kroa150	150	30	9830.2	11532.0	11532.0	0.0	150s	11532.0	11873.6	304.8	0.8s	11532.0	11677.2	157.0	3.5s
d198	198	40	8478.4	10573.0	10579.0	8.0	300s	10589.0	11080.3	388.9	10.0s	10781.0	10968.0	112.8	6.3s
krob200	200	40	10666.1	13177.0	13206.4	23.8	300s	13383.0	13688.9	199.8	6.1s	13336.0	13703.9	197.2	6.1s
gr202	202	41	295.7	318.0	321.0	1.6	300s	321.0	332.3	7.8	9.5s	318.0	322.3	3.5	5.9s
ts225	225	45	50452.6	68346.0	68563.7	166.3	300s	68583.0	71062.5	2181.2	5.2s	69932.0	71027.7	1091.6	7.6s
pr226	226	46	60402.4	64023.0	64069.2	252.9	300s	64023.0	66455.9	2414.3	7.8s	67048.0	67828.7	735.3	5.8s
gil262	262	53	811.1	1059.0	1070.8	10.1	300s	1074.0	1126.5	50.8	25.6s	1083.0	1122.6	25.9	11.3s
pr264	264	54	21731.5	29810.0	29879.3	56.0	300s	30096.0	31517.4	1319.3	24.1s	31220.0	32122.1	467.7	13.2s
pr299	299	60	14553.6	22644.0	22659.1	12.7	450s	22709.0	23228.1	410.8	28.4s	22793.0	23708.7	664.7	17.4s
lin318	318	64	12302.8	20795.0	21321.0	228.7	450s	21243.0	22046.0	564.3	49.0s	21181.0	21850.7	429.5	17.8s
rd400	400	80	-	6745.0	6833.2	42.0	600s	6801.0	6992.1	117.4	60.7s	6765.0	7034.2	148.4	37.2s
fl417	417	84	-	9708.0	9881.9	160.8	600s	9984.0	10506.1	302.3	28.1s	10147.0	10592.9	207.0	29.6s
gr431	431	87	-	1284.0	1312.3	18.0	600s	1286.0	1361.9	30.3	52.0s	1291.0	1307.8	11.8	36.4s
pr439	439	88	-	60642.0	62276.9	1710.7	600s	61067.0	66805.8	3461.8	59.1s	60815.0	63738.8	2170.8	38.0s
pcb442	442	89	-	22148.0	22612.6	325.9	600s	22665.0	23957.2	1058.8	88.0s	22321.0	23177.3	566.6	47.0s

compute lower bounds by means of LP-relaxations which are also listed in Table I.

While comparing the results of MA and VNS, we have to keep in mind that solutions generated by the latter are edge-biconnected, but not necessarily vertex-biconnected. On the other hand, every solution generated by the MA for the GMVBCNP is also a valid solution for the GMEBCNP per se.

We observe that the results obtained by VNS for the GMEBCNP using long runs have lowest costs, but the run times are very high as well. When comparing VNS with MA using short runs, we see that MA not only produces better and more robust results in general, but it requires also less time, especially on larger instances. The robustness is mainly due to the fact that MA uses a population containing multiple solutions whereas VNS is dependant on a single solution and its development during the search.

In other experiments, we tested the MA without local improvement. Those results were worse by about 3% on smaller instances and more than 10% on larger instances. Further increasing the probability of applying local improvement, did not yield significantly better results, but only led to longer running times.

IV. CONCLUSIONS AND FUTURE WORK

In this article, we proposed a Memetic Algorithm (MA) for the Generalized Minimum Vertex Biconnected Network Problem (GMVBCNP). It uses two local improvement methods to increase the solution qualities in the population. The graph reduction technique is applied in order to decrease the search space and to search the neighborhoods more efficiently. The crossover operator adapts its strategy according to the actual status of the population during the search process, either favoring intensification or diversification. The MA was tested on TSPlib instances consisting of 137 to 442 nodes with geographical center clustering. Results show that this approach is highly efficient and robust for solving this problem. For obtaining high quality solutions, the algorithm required for none of the instances more than one minute CPU time on average.

In future work, we intend to test the behavior of the MA on other types of instances, such as Euclidean instances with random clustering or non-Euclidean instances. They proved to be harder to handle than TSPlib instances with geographical center clustering at least for the GMEBCNP [3], [4]. There are also many possibilities for further improving the mutation and crossover operators along with their parameter settings. We would also like to investigate other ways of measuring the diversity in the population and other mechanisms for adapting the crossover strategy during the search process. Last but not least we are working on more powerful MIP models in order to obtain better bounds and/or to solve larger instances to provable optimality.

REFERENCES

[1] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*, D. Corne *et al.*, Eds. McGraw Hill, 1999, pp. 219–234.

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

[3] M. Leitner, B. Hu, and G. R. Raidl, "Variable neighborhood search for the generalized minimum edge biconnected network problem," in *Proceedings of the International Network Optimization Conference 2007*, B. Fortz, Ed., Spa, Belgium, 2007, pp. 69/1–6.

[4] B. Hu, M. Leitner, and G. R. Raidl, "The generalized minimum edge biconnected network problem: Efficient neighborhood structures for variable neighborhood search," accepted for Networks.

[5] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, 1976.

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

[7] J. Renaud, F. F. Boctor, and G. Laporte, "A fast composite heuristic for the symmetric traveling salesman problem," *INFORMS Journal on Computing*, vol. 8, issue 2, pp. 134–143, 1996.

[8] B. Hu and G. R. Raidl, "Effective neighborhood structures for the generalized traveling salesman problem," in *Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2008*, ser. LNCS, J. van

Hemert and C. Cotta, Eds., vol. 4972. Naples, Italy: Springer, 2008, pp. 36–47.

[9] C. Feremans, "Generalized spanning trees and extensions," Ph.D. dissertation, Universite Libre de Bruxelles, Brussels, Belgium, 2001.

[10] M. Fischetti, J. J. Salazar, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, pp. 378–394, 1997.