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

Anna Pagacz
Vienna University of Technology
Favoritenstrasse 9–11/1861
1040 Vienna, Austria
anna.pagacz@gmail.com

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*—We consider the generalized minimum vertex-biconnected network problem (GMVBCNP). Given a graph where nodes are partitioned into clusters, the goal is to find a minimal cost subgraph containing exactly one node from each cluster and satisfying the vertex-biconnectivity constraint. The problem is NP-hard. The GMVBCNP has applications in the design of survivable backbone networks when single component outages are considered. We propose a Memetic Algorithm with two crossover operators, three mutation operators, and a local search component based on two neighborhood structures. Furthermore, two different population management approaches are investigated. Experimental results document the efficiency of the new approach. In particular, the local search component and the population management strategies have a substantial impact on solution quality as well as running time.**

***Keywords*-network design; biconnectivity; memetic algorithm; local search; population management**

I. Introduction

The *generalized minimum vertex-biconnected network problem* (GMVBCNP) is defined as follows. We consider an undirected, weighted, complete graph $G = (V, E, c)$ with node set V, edge set E, and edge cost function $c : E \to \mathbb{R}^+$. Node set V is partitioned into r pairwise disjoint clusters $V_1, V_2, \ldots, V_r$. A solution to the GMVBCNP defined on G is a subgraph $S = (P, T)$ with $P = \{p_1, \ldots, p_r\} \subseteq V$ and $T \subseteq E$ connecting exactly one node from each cluster, i.e., $p_i \in V_i,\ \forall i = 1, \ldots, r$. We denote P as the set of *spanned nodes*. Furthermore, S may not contain any *cut nodes*. A cut node is a node whose removal would disconnect the graph. An example is given in Figure 1. The cost of such a vertex-biconnected network S is 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. The GMVBCNP appears in the design of survivable backbone networks that should be fault tolerant to a single component outage.

For this problem we suggest a *memetic algorithm* (MA) that uses problem specific variation operators, local improvement procedures to enhance the solution quality, and population management techniques to control the diversity and to avoid over-hasted convergence.

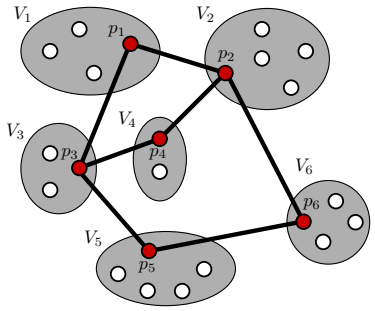


Figure 1. Example for a solution to the GMVBCNP.

II. Previous Work

There are many works in the literature on to classical vertex-biconnectivity augmentation problem (VBCAP). However the generalized version is rather new. VBCAP was first investigated by Eswaran and Tarjan [1]. They showed that it is NP-hard and introduced the so-called "block-cut graph" that allows efficient detection of cut vertices. This basic principle is used here as well. From the complexity of VBCAP it directly follows that also GMVBCNP is strongly NP-hard. A similar problem is the *generalized minimum edge-biconnected network problem* (GMEBCNP) where only edge-biconnectivity is required. It was introduced by Huygens [6] who proposed integer programming formulations, but no practical results were published. Hu et al. [3] presented variable neighborhood search (VNS) approaches for the GMEBCNP based on several types of neighborhood structures. They are adapted for the GMVBCNP in this work. Preliminary results of the MA were published in [5], and in this article the approach is extended particularly by a new population management technique. Further details can also be found in the first author's master thesis [8].

III. A Memetic Algorithm for the GMVBCNP

We use a standard memetic algorithm (MA) framework [7] for approaching the GMVBCNP as illustrated in Al-

Algorithm 2: Memetic Algorithm for GMVBCNP

randomly create initial population Π
repeat
  select two parental solutions $S_1,\ S_2 \in \Pi$
  create a new solution S_{new} by crossover on $S_1,\ S_2$
  mutate S_{new} with probability p_{mut}
  locally improve S_{new} with probability p_{ls}
  apply population management
until no new better solution found in last l iterations

gorithm 2. Tournament selection with a size of two and a worst solution replacement strategy are used. We apply two population management techniques which can be used either separately or in combination.

For the solution representation we first introduce the so-called *global graph*. Given a clustered graph $G = (V, E, c)$ the global graph denoted by $G^{\text{g}} = (V^{\text{g}}, E^{\text{g}})$ consists of nodes corresponding to clusters in G, i.e., $V^{\text{g}} = \{V_1, V_2, \ldots, V_r\}$, and the complete edge set $E^{\text{g}} = \{(V_i, V_j) \mid V_i, V_j \in V^{\text{g}},\ V_i \neq V_j\}$. 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 $S^{\text{g}} = \langle V^{\text{g}}, T^{\text{g}}\rangle$ with the global connections $T^{\text{g}} = \{(V_i, V_j) \in E^{\text{g}} \mid \exists (u, v) \in T \wedge u \in V_i \wedge v \in V_j\}$. Figure 2 illustrates a global structure of the solution in Figure 1.

A feasible candidate solution $S = (P, T)$ is characterized by P, the set of spanned nodes that are connected from each cluster, and the corresponding global structure T^{g} containing the global connections between clusters, i.e., $T^{\text{g}} = \{(V_i, V_j) \in E^{\text{g}} \mid \exists (u, v) \in T \wedge u \in V_i \wedge v \in V_j\}$. Using either P or T^{g} alone is insufficient, since decoding the solution would become NP-hard subproblems.

The initialization procedure for creating starting solutions is inspired by the fact that Hamiltonian cycles represent feasible solutions to the GMVBCNP: We first fix the set of spanned nodes P by randomly selecting $p_i \in V_i,\ \forall i = 1, \ldots, r$, and then connect them in random order to a cycle.

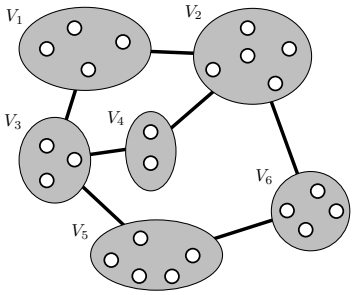
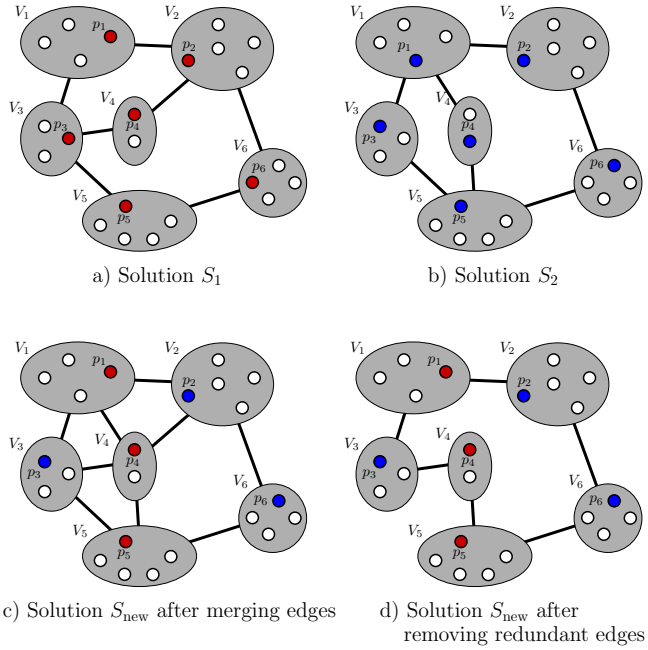


Figure 2. Global structure corresponding to solution in Figure 1.



a) Solution S_1 b) Solution S_2

c) Solution S_{new} after merging edges d) Solution S_{new} after removing redundant edges

Figure 3. Example for greedy crossover operator.

A. Recombination

For generating a new solution $S_{\text{new}} = (P_{\text{new}}, T_{\text{new}})$ from two parental solutions $S_1 = (P_1, T_1)$ and $S_2 = (P_2, T_2)$ we use two operators, the *node-oriented crossover* and the *greedy crossover*. A common step for both is the inheritance of spanned nodes. To determine P_{new} we apply classical uniform crossover, i.e., we randomly choose the spanned node of each cluster either from P_1 or P_2. Next we build the global structure $T^{\text{g}}_{\text{new}}$. Since the spanned nodes are already fixed, each global connection $(V_i, V_j) \in T^{\text{g}}_{\text{new}}$ now corresponds to a specific edge $(p_i, p_j) \in E$, $p_i \in V_i \wedge p_j \in V_j \wedge p_i, p_j \in P_{\text{new}}$.

For node-oriented crossover, $T^{\text{g}}_{\text{new}}$ is derived by visiting all clusters sequentially and for each cluster $V_i \in P^{\text{g}}_{\text{new}},\ \forall i = 1, \ldots, r$, we randomly decide from which parental edge set T^{g}_1 or T^{g}_2 the incident edges will be added. Hereby we favor the edges from the parent with lower connection cost by choosing it with probability 0.7. The resulting solution is checked by a depth-first-search algorithm and additional edges are introduced to eliminate eventually discovered cut nodes. On the other hand, greedy crossover generates $T^{\text{g}}_{\text{new}}$ by simply merging all global connections of both parental solutions, i.e., $T^{\text{g}}_{\text{new}} = T^{\text{g}}_1 \cup T^{\text{g}}_2$.

So far, both crossover operators create solutions that in general contain more edges than needed – especially greedy crossover. Therefore, in the next step we remove edges as long as the biconnectivity constraint is not violated. During this process we consider only edges $(p_i, p_j) \in T_{\text{new}}$ with $\deg(p_i) > 2 \wedge \deg(p_j) > 2$, where $\deg(p_i)$ denotes the degree of node p_i. These edges are considered in a particular

order for removal:

- α: decreasing costs
- β: decreasing perturbed costs $c'(p_i, p_j) \cdot \rho$, where ρ is a uniformly distributed random value $\in [0.5, \ldots, 1.0]$
- γ: random order

Sequence α obviously emphasizes intensification whereas γ yields a higher diversification. Each time one of the crossover operators is applied, we select the sorting criterion for edge removal randomly with probabilities $(p_\alpha, p_\beta, p_\gamma)$. These values are initially set to (0.5, 0.3, 0.2) and dynamically adapted during the search process by the population management procedure, which observes the diversity in the current population, see Section III-D. In particular, if the diversity factor *div* will be too small or too high, then p_γ is increased and p_α decreased, respectively.

Figure 3 presents an example for the greedy crossover operator. Preliminary experiments suggested to use *greedy crossover* with probability 0.95 and *node-oriented crossover* significantly less frequently with probability 0.05 since it has a much stronger heuristic bias.

B. Mutation

Each time a new solution S_{new} is generated, it is mutated with probability p_{mut}=0.6. We have three mutation operators which introduce a small amount of randomness and ensure that all solutions in the search space have the possibility of being examined:

- Change the spanned node of a random cluster.
- Add a global connection between two random clusters.
- Exchange the set of incident edges of two random clusters.

These operators are, again according to preliminary results, applied with probabilities 0.5, 0.4, 0.1, respectively.

C. Local improvement

After mutation each solution is further optimized via local improvement with probability p_{ls}. In [3], a graph reduction technique has been introduced and successfully applied to the GMEBCNP, which we also use here. The motivation is to reduce the search space for some neighborhood structures which the local improvement procedures are based on. Considering the global structure, we distinguish between *branching clusters* having a degree greater than two and *path clusters* having a degree of two. Note that there are no clusters with degree one, since this would violate the biconnectivity constraint.

Formally, for any global structure $S^g = \langle V^g, T^g \rangle$, we define a *reduced global structure* $S^g_{\text{red}} = \langle V^g_{\text{red}}, T^g_{\text{red}} \rangle$. V_{red} denotes the set of branching clusters, i.e. $V^g_{\text{red}} = \{V_i \in V^g \mid \deg(V_i) \geq 3\}$. T^g_{red} consists of edges which represent sequences of path clusters connecting these branching clusters, i.e. $T^g_{\text{red}} = \{(V_a, V_b) \mid$

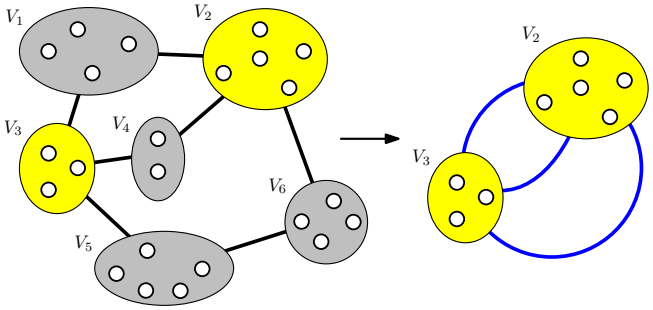


Figure 4. Example for applying graph reduction on the global structure in Figure 2.

$(V_a, V_{k_1}), (V_{k_1}, V_{k_2}), \ldots, (V_{k_{l-1}}, V_{k_l}), (V_{k_l}, V_b) \in T^g \wedge V_a, V_b \in V^g_{\text{red}} \wedge V_{k_i} \notin V^g_{\text{red}}, \forall i = 1, \ldots, l\}$. Corresponding to the reduced global structure $S^g_{\text{red}} = \langle V^g_{\text{red}}, T^g_{\text{red}} \rangle$ we define a *reduced graph* $G_{\text{red}} = \langle V_{\text{red}}, E_{\text{red}} \rangle$ with nodes representing all branching clusters $V_{\text{red}} = \{v \in V_i \mid V_i \in V^g_{\text{red}}\}$ and edges between any pair of nodes whose clusters are adjacent in the reduced global structure, i.e. $(i, j) \in E_{\text{red}} \Leftrightarrow (V_i, V_j) \in T^g_{\text{red}}, \forall i \in V_i, j \in V_j$. 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. Figure 4 shows an example for applying graph reduction on the global structure in Figure 2. V_2 and V_3 are branching clusters while all others are path clusters.

Based on this graph reduction technique, we make use of two neighborhood structures: The *Node Optimization Neighborhood* (NON) emphasizes the selection of the spanned nodes in the branching clusters while not modifying the global structure. First graph reduction is carried out on the current solution S. When V_{red} is the set of branching clusters, 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 efficient way. The pseudocode is given in

Algorithm 3: 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
```

Algorithm 4: 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 are in 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
```

Algorithm 3.

With the *Cluster Re-Arrangement Neighborhood* (CRAN) we try to improve a solution with respect to the arrangement of the clusters. Given a solution S with its global structure $S^g = (V^g, T^g)$, let $adj(V_a)$ and $adj(V_b)$ 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 CRAN means to swap these sets of adjacent clusters, resulting in $adj(V'_a) = adj(V_b)$ and $adj(V'_b) = adj(V_a)$ with V'_a and V'_b being the clusters in S' corresponding to V_a and V_b in S, respectively. S' is further improved by performing shortest path calculations to re-choose the spanned nodes in the path clusters. Whenever two path clusters are swapped only incremental updates on the paths that contain them is necessary. In case at least one of these clusters is a branching cluster, the structure of the whole solution graph may change, and consequently the graph reduction procedure is completely re-applied. The pseudocode is given in Algorithm 4.

D. Population management

Local improvement can lead to premature convergence. For this reason we apply population management in order to maintain a certain degree of diversity when new solutions are generated and added to the population. We propose two techniques. For *delta management* we measure the Hamming distance between the new solution S_{new} and each of k randomly chosen solutions in the current population Π. If the minimal distance is lower than a certain limit, S_{new} is mutated until the distance to the solutions in the population is sufficiently high. To save computation time the new solution is compared only with 10% of the population, i.e. $k = \lfloor 0.1 \cdot |P| \rfloor$.

The second technique is called *edge management*. It examines the percentage of the global connections covered by the solutions in the population in relation to all possible global connections. This ratio is calculated every t=50 iterations, and when it drops under a given threshold $p_{\text{edge}} < 70\%$ then the crossover parameters $(p_\alpha, p_\beta, p_\gamma)$ are set to (0.2,0.2, 0.6) to increase population diversity. On the other hand, when the ratio becomes higher than p_{edge} again, these parameters are reset to (0.5, 0.3, 0.2).

To conclude, if the number of covered global connections is too low, diversity is increased in the population. If it is too high, we put more emphasis on intensification. Parameters k and t control the granularity and time effort for the population management.

IV. Computational results

We tested our MA on Euclidean TSPlib[1] instances with geographical center clustering [2]. They contain up to 442 nodes partitioned into 89 clusters. The number of nodes per cluster varies. All experiments have been performed on an AMD Opteron Processor 2214 with 8GB RAM. The program was written in C++ and gcc version 4.1.3 was used. The stopping criterion for the MA was 200 generations without improvement of the so far best solution. In order to compute average solution values and corresponding standard deviations, 30 independent runs have been performed for each instance. In the default setting for the MA we use population size $|P| = 100$, local improvement probability $p_{\text{ls}} = 0.2$ and apply delta population management. These settings are justified by the following tests where best results are marked bold.

First we vary the population size between 50, 100, 150 and 200 individuals and keep the other parameters unchanged. Results are given in Table I. Listed are average objective values of finally best solutions $\overline{C(T)}$, corresponding standard deviations σ, and CPU-times $\overline{time}$ in seconds. We see that best results are obtained with $|P| = 100$ and $|P| = 150$. Due to population management, using a larger population size causes more computation time. This is the reason why $|P| = 100$ is preferred as the default value for $|P|$.

Next we show results for different probability values of applying local improvement values in Table II. We observe that local improvement has a high impact on the solution quality as well as runtime. By comparing in particular the settings of $p_{\text{ls}} = 0.2$ and $p_{\text{ls}} = 1.0$, we see that final results were better by 1 to 5% in the latter case. However, the search process took 30% to 125% more time. Therefore the following tests have been performed with $p_{\text{ls}} = 0.2$ for good balance.

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

Table I
RESULTS OF THE MA FOR DIFFERENT POPULATION SIZES, WITH PROBABILITY $p_{\text{ls}} = 0.2$ OF APPLYING LOCAL IMPROVEMENT.

Instance	$popsize = 50$			$popsize = 100$			$popsize = 150$			$popsize = 200$		
Name	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$
gr137	448.5	6.0	1.1	444.4	4.7	2.0	**443.1**	4.9	2.7	445.6	5.3	3.0
kroa150	11765.8	227.9	1.4	11809.5	289.0	2.3	**11744.1**	275.8	3.3	11752.6	289.7	7.4
d198	10869.1	152.4	2.6	**10840.6**	115.4	5.2	10844.4	128.0	6.9	10889.8	156.1	7.7
krob200	**13821.6**	316.8	2.7	13896.5	325.1	4.5	13831.3	296.0	6.3	13873.3	297.2	6.9
gr202	326.5	4.5	2.6	**323.1**	4.9	4.7	324.1	5.2	6.3	323.5	4.7	7.2
ts225	71394.7	1720.5	3.2	70296.9	754.5	5.4	**70285.3**	538.0	7.7	70305.6	444.2	8.7
pr226	67992.0	1609.3	2.8	66939.4	1186.9	5.4	**66905.7**	950.2	7.2	67088.4	702.2	8.1
gil262	1147.0	41.3	5.5	**1132.3**	27.2	10.4	1142.9	35.1	13.7	1154.6	33.0	15.2
pr264	32196.6	837.6	6.3	32031.5	511.1	10.5	**31945.2**	763.2	14.8	32151.3	729.4	16.8
pr299	24409.5	628.2	7.3	23801.5	682.3	14.1	**23726.0**	640.1	18.7	24063.4	823.5	21.0
lin318	22184.1	393.6	8.5	21811.0	396.2	15.1	**21785.0**	477.9	22.3	21961.4	419.9	24.0
rd400	7203.2	153.4	16.6	**7142.8**	137.5	30.5	7225.6	176.1	39.9	7234.6	182.8	47.1
fl417	10366.4	246.2	14.7	**10277.4**	201.4	27.7	10292.3	174.1	38.2	10304.6	224.1	44.2
gr431	1316.7	17.7	19.8	**1306.1**	13.2	36.6	1306.4	12.4	52.7	1308.7	11.4	58.6
pr439	63950.2	1718.7	20.1	62462.2	1448.1	36.4	**62259.6**	1132.1	51.0	62412.4	1008.9	57.2
pcb442	23986	594.0	22.3	**23350.9**	524.8	43.4	23745.3	604.9	57.4	24084.7	578.6	58.2

Table II
RESULTS OF THE MA FOR DIFFERENT PROBABILITIES p_{ls} OF APPLYING LOCAL IMPROVEMENT, POPULATION SIZE 100.

Instance	p_{ls}=0			p_{ls}=0.2			p_{ls}=0.6			p_{ls}=1.0		
Name	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$
gr137	476.5	17.8	1.5	444.4	4.7	2.0	441.3	2.5	2.9	**440.8**	1.7	3.9
kroa150	12994.5	691.1	2.1	11809.5	289.0	2.3	11619.8	150.3	3.6	**11601.8**	90.7	4.7
d198	11708.1	407.8	3.8	10840.6	115.4	5.2	10753.6	102.9	7.6	**10730.6**	97.9	9.3
krob200	15153.4	655.9	3.9	13896.5	325.1	4.5	13573.2	192.4	7.2	**13531.6**	167.5	9.2
gr202	355.7	11.0	3.4	323.1	4.9	4.7	320.0	2.3	7.3	**319.7**	2.5	9.1
ts225	78642.1	3100.0	4.7	70296.9	754.5	5.4	69910.2	307.4	8.9	**69768.5**	332.9	13.0
pr226	75712.8	3632.2	4.3	66939.4	1186.9	5.4	66687.1	966.5	7.4	**66126.3**	980.8	9.9
gil262	1 351.7	102.3	6.7	1132.3	27.2	10.4	1108.3	23.2	15.3	**1099.1**	18.9	20.7
pr264	36185.4	1799.4	7.0	32031.5	511.1	10.5	31409.1	847.7	17.4	**30860.8**	816.1	23.2
pr299	29313.8	2361.7	9.3	23801.5	682.3	14.1	23313.6	442.2	22.2	**23016.5**	325.0	28.1
lin318	28387	1927.6	9.7	21811.0	396.2	15.1	21562.3	373.3	24.8	**21368.9**	189.5	31.9
rd400	9678.5	689.5	15.8	7142.8	137.5	30.5	6958.8	122.5	51.2	**6887.7**	114.8	70.4
fl417	13307	844.6	18.5	10277.4	201.4	27.7	10156.3	198.4	41.5	**10129.1**	177.7	49.0
gr431	1716.6	120.4	18.7	1306.1	13.2	36.6	1296.0	10.4	60.5	**1290.4**	8.7	81.9
pr439	88956.3	6915.2	22.1	62462.2	1448.1	36.4	61284.5	670.3	59.6	**61132.1**	552.0	78.8
pcb442	32272.0	2432.3	23.1	23350.9	524.8	43.4	22877.7	295.2	69.6	**22481.1**	277.9	94.9

Table III
RESULTS OF MA WITH DIFFERENT POPULATION MANAGEMENT STRATEGIES, p_{ls}=0.2.

Instance	no management			delta management			edge management			delta & edge management		
Name	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$
gr137	445.8	6.3	1.8	**444.4**	4.7	2.0	446.6	6.2	2.5	**444.4**	5.2	3.9
kroa150	11898.2	418.2	2.3	11809.5	289.0	2.3	11858.9	328.1	3.1	**11618.3**	110.2	5.3
d198	10836.2	125.1	4.7	10835.2	115.4	5.2	**10796.3**	136.8	7.6	10798.4	124.9	12.7
krob200	13811.9	362.1	4.8	13896.5	325.1	4.5	**13856.7**	359.5	6.6	13658.4	197.7	10.8
gr202	326.5	6.1	4.4	323.1	4.9	4.7	325.7	5.8	6.5	**321.9**	4.1	10.3
ts225	70520.5	7 65.2	5.5	70296.9	754.5	5.4	70127.6	430.4	8.2	**70092.1**	517.6	14.4
pr226	66215.1	1247.7	4.8	66939.4	1186.9	5.4	**66039.0**	877.5	8.0	66852.2	816.9	16.1
gil262	1145.2	34.3	9.2	**1132.3**	27.2	10.4	1155.3	38.9	14.0	1140.7	31.3	26.8
pr264	31765.4	940.9	9.9	32031.5	511.1	10.5	31733.9	962.0	15.9	**31661.1**	760.6	29.6
pr299	24258.9	964.7	12.6	23801.5	682.3	14.1	24284.7	883.0	19.2	**23727.3**	532.6	37.7
lin318	21917.6	514.2	14.3	21811.0	396.2	15.1	21860.7	415.4	22.6	**21733.0**	358.1	42.2
rd400	7273.1	190.2	28.9	7142.8	137.5	30.5	7213.8	164.2	48.0	**7081.8**	149.1	82.3
fl417	10193.5	291.9	26.7	10277.4	201.4	27.7	**10105.7**	182.6	48.8	10212.1	181.2	104.8
gr431	1309.2	17.5	34.6	1306.1	13.2	36.6	**1305.7**	13.4	57.9	1308.2	12.5	111.9
pr439	62793.0	1587.9	36.7	62462.2	1448.1	36.4	62713.4	1882.5	55.3	62686.3	1716.5	56.4
pcb442	23473.8	633.5	42.0	**23350.9**	524.8	43.4	23782.3	653.4	63.6	24022.6	634.7	62.9

Table IV
RESULTS OF MA WITH DEFAULT CONFIGURATION AND EXTENSIVE CONFIGURATION.

Instance	default configuration			extensive configuration		
Name	$\overline{C(T)}$	σ	$\overline{time}$	$\overline{C(T)}$	σ	$\overline{time}$
gr137	444.4	4.7	2.0	441.6	2.7	8.3
kroa150	11809.5	289.0	2.3	11631.4	155.4	10.3
d198	10835.2	115.4	5.2	10716.0	100.9	24.9
krob200	13896.5	325.1	4.5	13646.2	245.8	22.2
gr202	323.1	4.9	4.7	321.8	4.5	21.1
ts225	70296.9	754.5	5.4	69706.9	418.8	28.9
pr226	66939.4	1186.9	5.4	65468.5	966.2	27.8
gil262	1132.3	27.2	10.4	1127.7	22.3	49.3
pr264	32031.5	511.1	10.5	30662.5	638.2	58.3
pr299	23801.5	682.3	14.1	23424.8	865.0	77.6
lin318	21811.0	396.2	15.1	21517.0	318.2	88.8
rd400	7142.8	137.5	30.5	7163.6	173.9	176.1
fl417	10277.4	201.4	27.7	10091.8	160.6	171.1
gr431	1306.1	13.2	36.6	1300.5	13.6	239.1
pr439	62462.2	1448.1	36.4	61654.0	817.3	233.7
pcb442	23350.9	524.8	43.4	23011.6	430.2	155.1

Table III lists results for different settings of the population management. It documents that population management is highly effective w.r.t. solution quality. Obtained results are in most cases significantly better when at least one of the strategies is turned on and usually best when both are applied. On the other hand, when both strategies were active, running times also increased considerably.

Finally, in Table IV we compare results between the default setting and an "extensive" configuration where all parameters are set towards best results, i.e., $|P| = 200$, $p_{\text{ls}} = 1.0$, and both population management strategies activated. We see that although the extensive configuration yields in general better results, the computation times increase enormously. Surprisingly, these results are not significantly superior to those when parameters are tuned for best results one at a time, compare to the last columns of Table II and III.

In order to compare the MA with an exact approach based on integer programming, we derived a multi-commodity flow *Mixed Integer Programming* (MIP) model from the GMEBCNP [3] by further adding constraints to guarantee vertex-biconnectivity. Due to space reason the full MIP formulation is not given here. Based on this model, by using the general purpose MIP solver CPLEX in version 11.2 we were able to obtain optimal solutions within reasonable time for small instances with up to around 80 nodes. However, the practical limit is quickly reached when the problem size increases. Since benchmark instances used in this article have at least 137 nodes, it was not possible to obtain optimal solutions anymore.

V. CONCLUSIONS AND FUTURE WORK

In this paper we considered the minimum vertex-biconnected network problem (GMVBCNP). We developed a memetic algorithm (MA) with advanced crossover operators, local improvement methods, and two population management techniques. The parameters for crossover are adapted dynamically during the search process by population management. Best results were obtained when local improvement is always applied, but the computation time then also grows. Two different population management strategies, which can be either used together or separately, have been investigated as well. Performed experiments clearly indicate that better results are obtained when both are used, but this configuration again requires more computation time.

Future work should include more comprehensive tests also on other types of instances, e.g., Euclidean instances with random clustering or non-Euclidean instances. Other types of genetic operators might also be promising, e.g., an edge-oriented crossover that keeps the edges but re-selects nodes. On the other hand, the mixed integer program formulation can be further developed and a hybridization with MA could also be done like in [4].

REFERENCES

[1] K. P. Eswaran and R. E. Tarjan. Augmentation problems. *SIAM Journal on Computing*, 5(4):653–665, 1976.

[2] C. Feremans. *Generalized Spanning Trees and Extensions*. PhD thesis, Universite Libre de Bruxelles, Brussels, Belgium, 2001.

[3] 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.

[4] B. Hu, M. Leitner, and G. R. Raidl. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14(5):473–499, 2008.

[5] B. Hu and G. R. Raidl. A memetic algorithm for the generalized minimum vertex-biconnected network problem. *9th International Conference on Hybrid Intelligent Systems - HIS 2009*, pages 63–68, 2009.

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

[7] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.

[8] A. Pagacz. Heuristic methods for solving two generalized network problems. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms, February 2010. supervised by G. Raidl and B. Hu.