An Evolutionary Algorithm with Solution Archive for the Generalized Minimum Spanning Tree Problem

Bin Hu and Günther R. Raidl

Vienna University of Technology
Favoritenstraße 9–11/186-1
1040 Vienna, Austria
{hu|raidl}@ads.tuwien.ac.at

1 Introduction

Attaching a solution archive to a metaheuristic for a combinatorial optimization problem in order to completely avoid evaluating duplicate solutions is a relatively novel approach [7]. When using a classical Evolutionary Algorithm (EA), for example, frequent re-evaluation of duplicate solutions cannot be avoided. This wastes valuable computation time which could have been spent in a more meaningful way otherwise. The solution archive takes advantage of this observation and stores already considered solutions in an appropriate data structure, allowing a fast detection of duplicates and efficient conversion of them into similar yet unvisited solutions.

This concept has been successfully applied on two problems where solutions are encoded as binary strings [7]. Similar methods exist where solutions are cached by hash tables [4] or stored in k-d trees [9]. However, these approaches either do not support efficient conversion of duplicates or they are applied to problems with rather simple solution representations.

In this paper we describe an archive-enhanced EA for the Generalized Minimum Spanning Tree Problem (GMSTP) which is defined as follows: Given an undirected weighted complete graph $G = \langle V, E, c\rangle$ with node set V partitioned into r pairwise disjoint clusters $V_1, V_2, \ldots, V_r$, edge set E and edge cost function $c : E \to \mathbb{R}^+$, a solution $S = \langle P, T\rangle$ is defined as $P = \{p_1, p_2, \ldots, p_r\} \subseteq V$ containing exactly one node from each cluster, i.e. $p_i \in V_i,\ i = 1, \ldots, r$, and $T \subseteq E$ being a tree spanning the nodes in P. The costs of S are the 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. The GMSTP was introduced in [5] and has been proven to be NP-hard. In recent years, many successful metaheuristic approaches [1–3] were developed for this problem.

2 Evolutionary Algorithm for the GMSTP

We use a classic steady state EA where the archive is consulted each time after a new solution is generated by crossover and mutation. In the following we briefly describe the EA components.

Solution encoding: We consider two dual encodings which can be seen as complementary. First, the *Spanned Nodes Representation* (SNR) characterizes solutions by their set of spanning nodes P. Decoding a solution means to find a classical minimum spanning tree on P. On the other hand, the *Global Structure Representation* (GSR) characterizes solutions by their so-called *global connections*, defining which clusters are adjacent in the solution. Since they always describe a tree structure on the cluster level, we store for each cluster $V_i,\ i = 2, \ldots, r$, its predecessor $pred(V_i)$ when we root the tree at V_1. For decoding, the optimal spanned node in each cluster can be obtained via dynamic programming [6].

Genetic operators: As selection we use tournament selection of size 2. Crossover and mutation operators are implemented for both representations separately. For SNR, uniform crossover and one-point-mutation are applied on P. For GSR, edge recombination for spanning trees [8] and mutation by exchanging global connections are implemented. Each time a genetic operator is carried out, we decide randomly which representation to use.

Solution Archive: The solution archive is implemented by two tries, storing solutions for each representation, respectively. Each trie is able to identify duplicate solutions in its associated solution encoding. If a duplicate is found, it is converted by the corresponding trie by applying randomized, systematic changes until it becomes a yet unvisited solution. However, since there are two tries, it is possible that the new solution created by one trie becomes a duplicate in the other trie. Hence for each conversion we alternately modify the solution and perform a re-check in the opposite trie until the derivated solution is new in both tries. Randomization is particularly important during this conversion process in order to avoid biasing (i.e., certain areas of the solution space being over-searched). The complexity of the trie-operators is relatively low. Searching and inserting new solutions can be done in $O(r)$ time. Depending on the representation, modifying duplicates requires $O(r)$ in SNR and $O(r^2)$ in GSR.

3 Preliminary Results and Conclusions

We tested our approach on TSPlib instances with up to 442 nodes partitioned into 89 clusters using geographical center clustering. For each instance we performed 30 independent runs and each run was terminated when a time limit was reached. The EA was tested in four variants: EA without archive, EA with archive based on SNR, EA with archive based on GSR, and EA with full archive using both representations. The instances (last three digits indicate the number of nodes) and their time limits are listed in Table 1. For each EA variant we show the average final solution values $\overline{C(T)}$ and the corresponding standard deviations. Best results are marked bold. We observe that the EA without archive performs worst in general. Among the two variants where the archive only uses one representation, GSR is the better choice. However, if we combine both of them, the EA performs best on all instances. The results clearly indicate that the

archive improves the search performance of the EA. Considering both solution representations is also a crucial step towards overall success.

Table 1. Results of different EA variants

		no archive		SNR archive		GSR archive		full archive	
Instance	time	$\overline{C(T)}$	std dev	$\overline{C(T)}$	std dev	$\overline{C(T)}$	std dev	$\overline{C(T)}$	std dev
gr137	150s	329.4	0.5	329.3	0.5	**329.0**	0.0	**329.0**	0.0
kroa150	150s	9830.6	31.4	9831.3	30.1	**9815.0**	0.0	**9815.0**	0.0
d198	300s	7055.1	8.7	7059.6	9.0	7044.6	2.3	**7044.0**	0.0
krob200	300s	11275.0	45.6	11248.9	7.5	**11244.0**	0.0	**11244.0**	0.0
gr202	300s	242.1	0.3	242.2	0.4	**242.0**	0.2	**242.0**	0.0
ts225	300s	62290.8	40.4	62299.1	50.9	62268.6	0.5	**62268.4**	0.5
gil262	450s	945.5	4.0	945.0	3.7	942.4	2.0	**942.0**	0.0
pr264	450s	21893.2	7.7	21898.4	20.9	**21886.0**	0.0	**21886.0**	0.0
pr299	450s	20352.1	37.4	20349.7	24.9	20318.5	11.3	**20318.1**	11.3
lin318	600s	18545.9	29.2	18547.3	25.6	18525.8	12.4	**18511.0**	10.8
rd400	600s	5953.0	15.4	5959.4	20.2	5946.4	10.8	**5940.2**	6.5
fl417	600s	**7982.0**	0.0	**7982.0**	0.0	**7982.0**	0.0	**7982.0**	0.0
gr431	600s	1034.1	1.4	1033.4	0.9	1033.3	0.7	**1033.0**	0.0
pr439	600s	51921.4	60.7	51888.5	56.3	51810.5	26.5	**51791.0**	0.0
pcb442	600s	19717.0	59.5	19708.1	70.2	19632.6	21.1	**19623.7**	15.9

References

1. B. Golden, S. Raghavan, and D. Stanojevic. Heuristic search for the generalized minimum spanning tree problem. *INFORMS Journal on Computing*, 17(3):290–304, 2005.
2. 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.
3. H. Jiang and Y. Chen. An efficient algorithm for generalized minimum spanning tree problem. In *GECCO '10: Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 217–224, New York, NY, USA, 2010. ACM.
4. J. Kratica. Improving performances of the genetic algorithm by caching. *Computers and Artificial Intelligence*, 18(3):271–283, 1999.
5. Y. S. Myung, C. H. Lee, and D. W. Tcha. On the generalized minimum spanning tree problem. *Networks*, 26:231–241, 1995.
6. P. C. Pop. *The Generalized Minimum Spanning Tree Problem*. PhD thesis, University of Twente, The Netherlands, 2002.
7. G. R. Raidl and B. Hu. Enhancing genetic algorithms by a trie-based complete solution archive. In *Evolutionary Computation in Combinatorial Optimisation – EvoCOP 2010*, volume 6022 of *LNCS*, pages 239–251. Springer, 2010.
8. G. R. Raidl and B. A. Julstrom. Edge-sets: An effective evolutionary coding of spanning trees. 7(3), 2003. IEEE Transactions on Evolutionary Computation.
9. S. Y. Yuen and C. K. Chow. A non-revisiting genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 4583–4590. IEEE Press, 2007.