Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 154 (2004) 323–329
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Short Communication

Solving group Steiner problems as Steiner problems

C.W. Duin [a,*], A. Volgenant [a], S. Voß [b,c]

[a] *Operations Research Group, Faculty of Economics and Econometrics, University of Amsterdam, Roetersstraat 11, 1018 WB Amsterdam, The Netherlands*
[b] *Technische Universität Braunschweig, Abteilung ABWL, Wirtschaftsinformatik und Informations management, Germany*
[c] *Institute of Information Systems, University of Hamburg, Von-Melle-Park 5, D-20146 Hamburg, Germany*

Received 24 September 2001; accepted 19 September 2002

Abstract

The generalized spanning tree or group Steiner problem (GSP) is a generalization of the Steiner problem in graphs (SPG): one requires a tree spanning (at least) one vertex of each subset, given in a family of vertex subsets, while minimizing the sum of the corresponding edge costs.

Specialized solution procedures have been developed for this problem. In this paper we investigate the performance of a known but so far neglected transformation to the undirected Steiner problem in graphs. When combined with a recent metaheuristic for the SPG this straightforward approach compares favorably with specialized GSP heuristics. Thus we set a standard for future algorithms.

Keywords: Combinatorial optimisation; Group Steiner problem; Generalized minimal spanning tree; Metaheuristics; Pilot method

1. Introduction

Lately in literature, there has been a growing interest for a problem related to the Steiner tree problem in graphs. This more general problem, the so-called group Steiner problem (GSP), considers in a weighted graph (V, E, c) a family of vertex sets $V_k \subset V$ for indices $k \in K$. The problem is to find a minimum cost tree that spans at least one vertex of each group V_k.

Two well-known special cases of GSP arise when each group is singleton. If for each vertex $v \in V$ there is a singleton group $\{v\}$, then the problem is a standard minimum spanning tree (MST) problem. If $V \setminus (\cup_k V_k)$ is non-empty with again all groups singleton, then the problem specializes to the Steiner problem in graphs (SPG for short). Thus the NP-hardness of the GSP follows directly from the NP-hardness of the SPG.

The foregoing definition of GSP slightly differs from other problem variants given in literature; it includes these other variants as special cases. Firstly, we allow for 'free vertices' that do not lie in any of the given groups, i.e., the set $V \setminus (\cup_k V_k)$ may be non-empty; we denote this set as V_0. Secondly,

* Corresponding author.
E-mail addresses: ceesd@fee.uva.nl (C.W. Duin), stefan.voss@tu-bs.de (S. Voß).

0377-2217/$ - see front matter
doi:10.1016/S0377-2217(02)00707-5

we do not require that the groups are mutually disjoint vertex sets. In the generalized MST problem the groups V_k are disjoint and they partition the set V. For this type of the GSP Feremans et al. (2002) have analytically compared the linear relaxations of eight different formulations.

An Euclidean version of the GSP, in the two-dimensional plane, was introduced by Cockayne and Melzak (1968) and later studied by Weng (1985). Reich and Widmayer (1989) were the first to study the problem in graphs. They model an interesting application with regard to the layout of integrated circuits into the *group* (or *class*) *Steiner problem* and formulated two heuristics. Ihler et al. (1999) gave computational results and showed the problem to be MAX SNP hard, even so, if restricted to a tree graph embedded in a unit grid. Dror et al. (2000) formulated an agricultural application and tested several heuristics; among them a genetic algorithm performed best in terms of solution quality. The heuristics mentioned so far can, at least theoretically, produce solutions as much as $O(|K|)$ times worse than optimal. Helvig et al. (2001) designed an algorithm with for any given fixed $\varepsilon > 0$, a performance bound of $O(|K|^{\varepsilon})$ times the optimal cost and such that this algorithm runs in time polynomial with exponent $1/\varepsilon$.

This paper considers a straightforward, but so far neglected transformation of the GSP to the SPG. A GSP instance can be transformed in linear time as follows: For each group V_k introduce a new vertex, say v_k, into the graph and connect it with artificial edges (v_k, i) of high cost $c(v_k, i) = M$ to every vertex $i \in V_k$. Defining in this enlarged graph $\{v_k | k \in K\}$ as the set of terminal nodes, a Steiner tree on this terminal set corresponds to a minimal group Steiner tree in the original graph, by means of a removal of the artificial edges. Consequently one might solve the GSP directly by one of the various algorithms available for the SPG.

Though the transformation is known, see e.g. Voß (1990) and Hwang et al. (1992), one might have been regarded it as just a theoretic option. This paper shows however that the transformation can perform quite satisfactorily, when applied with the right SPG solution procedures. As we can solve GSP instances of respectable size up to optimality we can now compare in this paper heuristic results with optimal values.

The transformation can shed a new light on specialized heuristics for the GSP as developed in literature. For instance, as we will see, an existing SPG heuristic corresponds by means of the transformation to the GSP heuristic H1 of Dror et al. (2000), in such a way that we obtain for H1 a reduced running time. However, more modern SPG heuristics can generate solution trees of superior quality. Computational results are given in Section 3. A set of test instances with up to 500 vertices, solved heuristically in Dror et al. (2000), is now solved speedily to optimality. Experiments with other instances do indicate that the type of SPG instance as derived from GSP-instances is indeed (much) harder to solve to optimality than the 'normal' SPG type. A metaheuristic, the pilot method of Duin and Voß (1999) still performs satisfactorily. We will review the pilot method shortly in Section 2.

2. Steiner tree heuristics for the GSP

Suppose that one applies the heuristic of Takahashi and Matsuyama (1980) on the SPG instance as derived by the transformation from a GSP-instance. And suppose that that one repeats this heuristic once for every start vertex v_k $(k \in K)$, while saving the best result; i.e., the repetitive heuristic given for the SPG by Winter and MacGregor Smith (1992). It is easy to see that one then obtains for the considered GSP instance a solution equivalent to the solution obtained by the heuristic H1 of Dror et al. (2000), a specialized heuristic with time complexity $O(|K|^2|V|^2)$.

In Duin and Voß (1997) a simple procedure named CONSTRUCT constructs the heuristic tree T of Takahashi and Matsuyama (1980) from an arbitrary start vertex in $O(|K||T|)$ time, when given as input all shortest path distances d_{ik} for vertices with $i \in V$ and $k \in K$. These distances d_{ik} become available after running $|K|$ times the algorithm of Dijkstra with running time $O(|V|\log|V| + |E|)$; see, e.g., Ahuja et al. (1993). Consequently the time complexity of H1 can reduce from $O(|K|^2|V|^2)$ to $O(|K||V|\log|V| + |K||E| + |V||K|^2)$.

With the time complexity being reduced, this heuristic is pretty fast; however, as already shown in Dror et al. (2000), for small values of $|K|$ its results are sometimes quite bad. However after the transformation other SPG algorithms are likely to produce better results than H1.

A still rather speedy SPG-heuristic, which will produce on GSP-derived instances better results than H1, is the heuristic of Minoux (1991) with time complexity $O(|V|^3)$. It takes as input the all-pair shortest path distances d and (re)computes a MST on the so-called *distance graph* (K, d), the complete graph with vertex set K and weights d. The heuristic operates as follows:

Step 1 (*Initialization*)
  Compute the MST S on the distance graph (K, d), defined with vertex set K and a complete set of edges (i, j) with weight d_{ij};
Step 2 (*Computation*)
  For all vertices $v \notin S$ **do** compute S_v the MST of the distance graph $(K \cup \{v\}, d)$;
  Let the cost $d(S_v)$ of the MST extension with v be at minimum for vertex v_0;
Step 3 (*Extension/termination*)
  If $d(S_{v0}) < d(S)$ **then**
    replace S by S_{v0}, K by $K \cup \{v_0\}$
      and return to step 2
  else
    replace the 'distance edges' in S
      by path edges and **stop**.

In Duin and Voß (1999) further improved SPG heuristics are studied as *pilot methods*. Roughly speaking, the idea is to use an existing heuristic, the so-called *pilot*, to produce the solutions S on K and S_v on $K' = K \cup \{v\}$. The best performance is obtained while using as pilot heuristic SVERTEX of Duin and Voß (1997).

SVERTEX can, again, be best compared to the heuristic of Minoux, producing a solution of comparable quality, but it is faster. First it uses (several calls to) an $O(|K|)$ subroutine to (re)compute S_v, see also Spira (1975). Secondly, it picks its vertices v_0 from the top of a heap data structure. This heap contains unaccepted vertices v with as key an (optimistic) estimate of $d(S) - d(S_v)$. For the vertex on the top of the heap, the tree S_v and its key are updated. If the key is still maximal the vertex is accepted as next v_0, otherwise vertex is pushed down in the heap. If accepted the vertex is removed from the heap, and negative key values are updated heuristically in $O(1)$ per node. Thus the running time of the algorithmic phase of SVERTEX reduces to $O(\alpha\beta|V||K| + \beta|V|\log|V| + \gamma|V|)$, where α is the average number of calls to the $O(|K|)$ subroutine, β is the average number of times that a vertex v, taken from the top of the heap is pushed down, and γ is the number of vertices accepted in master solution S. For 'normal' Steiner problem instances it was observed that the coefficients α and β do hardly grow with the problem size and that coefficient γ is of $O(|K|)$.

3. Computational results

After the transformation to the SPG, one may either try to solve the resulting SPG instance exactly using an exact SPG-solver, or approximately using heuristic algorithms.

Firstly, we have solved to optimality a series of test problems from literature with an exact SPG-solver similar to that of Duin (1993). Table 1 gives the results for the test problem set used in Dror et al. (2000).

The 20 instances were generated as follows: for given values for the numbers of vertices $|V|$ and edges $|E|$, first a random tree over all vertices was constructed and then the remaining edges were randomly added to the tree. Integer edge weights were uniformly generated between 1 and 50. For a partitioning in k groups, the first k vertices were assigned to different groups and the remaining vertices were randomly assigned to a group, thus vertex groups of different sizes might arise.

Dror et al. (2000) run their genetic algorithm on a Pentium II 200 MHz processor using Microsoft Visual C^{++} (the obtained objective values [1] are in column z_{GA}) and we obtained the optimal values (column z^*) on a Celeron 500 MHz processor using

[1] For three instances, where $(|V|, |E|)$ is equal to $(100, 300)$, $(150, 500)$ and $(250, 500)$, the results were updated as listed in the table by private communication.

Table 1
Results of the transformation compared to the genetic algorithm of Dror et al. (2000)

Problem parameters			Genetic algorithm		Transformation to SPG		
$\|V\|$	$\|E\|$	k	z_{GA}	cpu seconds PII 200 MHz	z^*	cpu seconds Celeron 500 MHz	Number of subproblems
25	50	4	23	1.3	23	0.0	1
	100	8	41	4.6	41	0.0	1
	150	10	36	7.8	36	0.0	1
50	150	5	18	5.8	18	0.0	1
	300	10	27	28.5	27	0.0	1
75	200	8	60	25.4	55	0.1	2
	300	10	67	60.4	67	0.2	6
	400	15	55	132	53	0.1	3
100	300	7	37	38.7	37	0.0	3
	500	10	50	109	48	0.0	1
150	300	8	62	86.8	50	0.1	2
	500	12	68	101	68	0.2	3
200	500	10	44	107	44	0.2	2
	1000	20	58	600	50	0.2	2
250	500	10	60	201	60	0.1	3
	1000	25	148	310	117	0.4	3
300	1000	20	91	1339	88	0.7	9
	2000	30	101	1605	85	3.0	25
	3000	40	98	2102	88	6.1	38
500	5000	50	141	2283	105	84.8	292

Delphi Pascal version 3. With a quite substantial computational effort the genetic heuristic found an optimal solution for 10 of the 20 test instances; over the other instances the gap to the optimal value is 15.1% on average and 34.3% maximal. The transformation found optima in far shorter times, whatever the difference of hard- and software may be.

Secondly, we evaluated approximate SPG algorithms, testing their quality on other GSP-derived instances. Generating also instances conform the more general definition of the GSP given here, we revised the generation scheme as follows: first a random number $f \in (0, 0.5)$ determines the number of vertices in the free Steiner group V_0 as $f|V|$, truncated to the integer value and one randomly selects this set from V. Out of $V \setminus V_0$, we then first assign a single (different) vertex to each of the k groups. The set of as yet unchosen vertices forms a pool W with $|W| = |V| - |V_0| - k$ vertices. Now, for each group in turn one randomly takes $-1 + (|V| - |V_0|)/k$ additional vertices out of W, and pool W is renewed before the next group is filled up in this way. Thus some vertices of W may become part of more than one group and others may not become part of any group; the latter, if any, are added to the Steiner group V_0.

In Table 2 we compare a classical heuristic, we have chosen the algorithm of Takahashi and Matsuyama (1980), with two implementations of the pilot method. The results are given for Euclidean instances (100 per line), random instances (100 per line) and rectilinear instances (20 per line). The random and Euclidean problems do not differ in structure only in the edge weights. A connected graph is produced by a random graph generator, taking as input the desired number of vertices $|V|$ and edges $|E|$. In the random problem type the edge weights are uniform random integer weights in the interval $[1, 100]$. In the Euclidean problem type each

Table 2
Heuristic results on GSP's with *unrestricted* groups (panel A) and *disjoint* groups (panel B): fraction solved optimally (*), average (a%) and maximum (m%) gap and average cpu time in second (exclusive the time for all-pair shortest paths)

Problem	Takahashi and Matsuyama				Pilot–Rush				Pilot–Drop			
$\|V\|$	*	a%	m%	Time	*	a%	m%	Time	*	a%	m%	Time
Panel A												
Euclidean												
100	0.11	17.8	107	0.0	0.82	0.3	9.6	0.09	0.92	0.1	2.4	0.8
200	0.00	21.1	112	0.0	0.67	0.3	2.9	0.25	0.91	0.0	0.8	5.0
300	0.01	27.4	157	0.0	0.55	0.6	4.1	0.58	0.79	0.1	1.3	16.5
400	0.04	25.1	135	0.01	0.46	0.7	4.5	1.04	0.64	0.2	2.1	36.7
Random												
100	0.16	12.7	102	0.0	0.93	0.1	3.1	0.09	0.99	0	0.3	0.7
200	0.06	21.4	165	0.0	0.76	0.3	4.0	0.25	0.96	0	0.9	4.6
300	0.04	21.7	204	0.0	0.67	0.4	4.3	0.53	0.87	0.1	2.0	13.9
400	0.02	24.1	124	0.01	0.57	0.6	4.7	0.93	0.85	0.2	3.4	31.7
Rectilinear												
100	0.05	25.5	77	0.0	0.60	0.9	5.2	0.10	1.00	0	0	1.0
196	0	40.8	302	0.0	0.55	0.8	3.6	0.33	0.80	0	0.2	6.6
289	0	38.2	216	0.0	0.45	1.0	3.3	0.72	0.60	0.3	1.2	20.6
400	0	53.2	276	0.01	0.35	1.6	4.1	1.51	0.65	0.2	2.7	57.1
Panel B												
Euclidean												
100	0.09	21.1	108	0.0	0.75	0.5	5.9	0.10	0.88	0.1	2.6	0.7
200	0.01	24.7	148	0.0	0.56	0.9	5.1	0.24	0.85	0.2	3.7	4.7
300	0.01	29.3	171	0.0	0.64	0.8	6.5	0.46	0.84	0.2	3.8	7.1
400	0.00	33.0	141	0.0	0.60	1.0	7.1	0.83	0.84	0.2	2.6	14.2
Random												
100	0.11	23.7	118	0.0	0.88	0.3	7.5	0.10	0.95	0.1	2.5	0.7
200	0.02	31.4	263	0.0	0.76	0.5	13.5	0.24	0.93	0.0	1.2	4.2
300	0.04	33.3	124	0.0	0.77	0.9	10.6	0.43	0.96	0.1	2.6	6.5
400	0.03	36.6	179	0.01	0.68	0.9	7.5	0.78	0.89	0.2	5.3	13.0
Rectilinear												
100	0.0	49.9	283	0.0	0.65	0.5	2.8	0.10	0.85	0.1	0.9	0.8
196	0.0	48.1	235	0.0	0.55	1.3	5.6	0.28	0.65	0.2	2.0	5.3
289	0.0	43.0	100	0.0	0.40	1.4	6.2	0.49	0.93	0.1	1.2	7.4
400	0.0	52.6	129	0.02	0.47	1.3	7.5	0.89	0.93	0.1	1.5	16.4

node is first associated with a point in the plane, drawing as coordinates uniform random reals in the interval $[1, 100]$; then each edge weight is given as the Euclidean distance between the corresponding end-points (rounded up to integer value).

Concerning the number of groups k and the number of edges $|E|$, we considered 20 different types of instances: four group densities, taking k equal to $[\log_2|V|]$, $[\sqrt{|V|}]$, $2\sqrt{[|V|]}$ and $|V|/4$ ($[\cdot]$ denoting integer truncation), were combined with five densities of edges, taking $|E|$ equal to $1.5|V|$, $2|V|$, $[|V|\ln|V|]$, $2[|V|\ln|V|]$ and $[|V|(|V|-1)/4]$. Generating five instances per type, we thus considered 100 instances per size $|V|$. The same group densities were tested in the rectilinear case, but as the number of edges is now fixed by the imposed grid structure, we have considered here 20 instances per size $|V|$. For the rectilinear type, we generated 'square' grids with the number of vertical lines equal to the number of horizontal lines; the distance between adjacent lines is a uniform random integer in the interval $[1, 100]$.

Table 2 (panel A) gives the results for the case where overlapping is allowed (unrestricted groups) and panel B strictly considers disjoint groups. Mind that the latter table does not consider 100 instances for $|V| = 289$, 300, 400 but 75 instances (and 15 for the rectilinear type); as one misses here the results for the type $k = [|V|(|V| - 1)/4]$. Here we could not solve up to optimality all the Euclidean and rectilinear instances, and we left them out altogether. To improve the readability the number of instances solved to optimality is given as a fraction of the instances tested.

Clearly the results of the classical Takahashi and Matsuyama (1980) heuristic are very poor, with average gaps from about 10% to over 50% for the more difficult rectilinear test instances. The maximal gaps are of course larger, even up to 300% and the fraction of optimally solved instances is low. We note that with other classical heuristics of similar speed, e.g., the heuristic of Kou et al. (1981) in the implementation of Mehlhorn (1988), one would not attain a better solution quality.

For a higher computational price the Pilot–Drop method gives the best results, with average gaps always below 0.3%. The Pilot–Rush method gives, at a modest time effort, good results with average gaps nearly always below 1% (but for the largest rectilinear instances). Where the times for the determination of all-pair shortest paths have to be added to the given times, the relative differences in the computation times are smaller. For larger $|V|$ the average times listed in Table 2 (panel A) are higher than those in Table 2 (panel B); however, this is due to the fact that the instances with higher k, which are missing in Table 2 (panel B), are the more time consuming. The fraction of optimally solved instances is substantial for both Pilot heuristics; the quality of the Pilot–Drop is also illustrated by the largest fractions. For all three heuristics the fraction of optimally solved instances is decreasing for growing sizes of the test instances, as expected.

4. Conclusion

We applied a linear transformation from the GSP to the SPG. It enables SPG algorithms to solve the GSP problem. Computational evidence was given showing that this transformation of the GSP to the SPG is not just a theoretic option. For test instances with up to 500 vertices with various numbers of groups and edge-densities, the approach of transformation has been computationally compared to a specialized genetic algorithm. In all cases we attained optimal values in computation times shorter than as for this heuristic.

Also our heuristic tests illustrate the usefulness of the transformation. Though classical heuristics like that of Takahashi and Matsuyama (1980) fail with respect to the quality of the produced solution, a metaheuristic like the pilot method obtains consistently solutions of near optimal cost. Faster variants like Pilot–Rush and Pilot–Drop retain more or less the quality of the produced solution, while their computation time is modest, especially that of Pilot–Rush. We inserted Pilot–Drop to show that the relation between quality and time is as expected. For the considered test problems the heuristics give average gaps within 1% for the Euclidean and the random instances; the rectilinear instances are clearly more difficult, resulting in average gaps of up to 1.6%.

Salazar (2000) stated that the transformation from the GSP to the SPG requires working on an enlarged graph and managing large costs and that both features are not suitable from a practical point of view. We can, however, conclude that the transformation is suited to solve heuristically the GSP, and also optimally if the graph's size is moderate.

Acknowledgement

We are grateful to Prof. Haouari for providing us with the data of his test instances (Table 1).

References

Ahuja, R.K., Magnanti, T., Orlin, J.B., 1993. Network Flows. Prentice Hall, New Jersey.

Cockayne, E.J., Melzak, Z.A., 1968. Steiner's problem for set terminals. Quarterly Applied Mathematics 26, 213–218.

Dror, M., Haouari, M., Chaouachi, J., 2000. Generalized spanning trees. European Journal of Operational Research 120, 583–592.

Duin, C.W., 1993. Steiner problem in graphs: Approximation, reduction, variation, Ph.D. thesis, University of Amsterdam.

Duin, C.W., Voß, S., 1997. Efficient path and vertex exchange in Steiner tree algorithms. Networks 29, 89–105.

Duin, C.W., Voß, S., 1999. The Pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs. Networks 34, 181–191.

Feremans, C., Labbé, M., Laporte, G., 2002. A comparative analysis of several formulations for the generalized minimum spanning tree problem. Networks 39, 29–34.

Helvig, C.S., Robins, G., Zelikovsky, A., 2001. An improved approximation scheme for the group Steiner problem. Networks 37, 8–20.

Ihler, E., Reich, G., Widmayer, P., 1999. Class Steiner trees and VLSI-design. Discrete Applied Mathematics 90, 173–194.

Kou, L., Markowsky, G., Berman, L., 1981. A fast algorithm for Steiner trees. Acta Informatica 15, 141–145.

Mehlhorn, K., 1988. A faster approximation algorithm for the Steiner problem in graphs. Information processing Letters 27, 125–128.

Minoux, M., 1991. Efficient greedy heuristics for Steiner tree problems using reoptimization and supermodularity. INFORMS 28, 221–233.

Reich, G., Widmayer, P., 1989. Beyond Steiner's problem: A VLSI oriented generalization. In: Nagl, M. (Ed.), Graph-Theoretic Concepts in Computer Science. Lecture Notes in Computer Science, vol. 411. Springer-Verlag, Berlin.

Salazar, J.J., 2000. A note on the generalized Steiner tree polytope. Discrete Applied Mathematics 100, 137–144.

Spira, P.M., 1975. On finding and updating spanning trees and shortest paths. SIAM Journal on Computing 4, 375–380.

Takahashi, H., Matsuyama, A., 1980. An approximate solution for the Steiner problem in graphs. Mathematica Japonica 24, 573–577.

Voß, S., 1990. A survey on some generalizations of Steiner's problem. In: Papathanassiu, B., Giatas, K. (Eds.), 1st Balkan Conference on Operational Research Proceedings 1988. Hellenic Productivity Center, Thessaloniki, pp. 41–51.

Weng, J.F., 1985. Generalized Steiner problem and hexagonal coordinate system. Acta Mathematicae Applicatae Sinica 8, 383–397.

Winter, P., MacGregor Smith, J., 1992. Path-distance heuristics for the Steiner tree problem in undirected networks. Algorithmica 7, 309–327.

Hwang, F.K., Richards, D.S., Winter, P., 1992. The Steiner tree proble. Annals of Discrete Mathematics, 53.