Seismic Vessel Problem

Gregory Gutin[*], Helmut Jakubowicz[†], Shuki Ronen[‡] and Alexei Zverovitch[§]

November 14, 2003

Abstract

We introduce and study a new combinatorial optimization problem, the Seismic Vessel Problem (SVP), that arose in an industrial application. SVP generalizes the Stacker Crane Problem. We suggest a transformation from SVP to the Symmetric Traveling Salesman Problem. We report our computational experience with solving SVP instances drawn from industrial practice (geophysical seismic acquisitions).

Keywords: Travelling Salesman, Stacker Crane Problem, Seismic Acquisition.

1 Introduction

The Seismic Vessel Problem (SVP) is defined by a set of line segments (survey lines), all of which need to be traversed (shot) exactly once; see Fig. 1. Some lines can be shot in either direction, other have directional constraints imposed on them. The objective is to minimize the travel time between lines by choosing an optimal ordering of lines (and specifying in which direction each line has to be shot). The function that defines the travel time between lines can be of arbitrary complexity and in general is defined as a matrix of "line change" costs for all combinations of pairs of lines and shooting directions.

[*]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, gutin@cs.rhul.ac.uk. Corresponding author.

[†]Veritas DGC Ltd, Manor Royal Estate, Crawley, West Sussex, RG10 9QN, UK, Helmut_Jakubowicz@veritasdgc.com

[‡]Veritas DGC Ltd, 10300 Town Park Drive, Houston, TX 77072, USA, Shuki_Ronen@veritasdgc.com

[§]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, zvero@cs.rhul.ac.uk

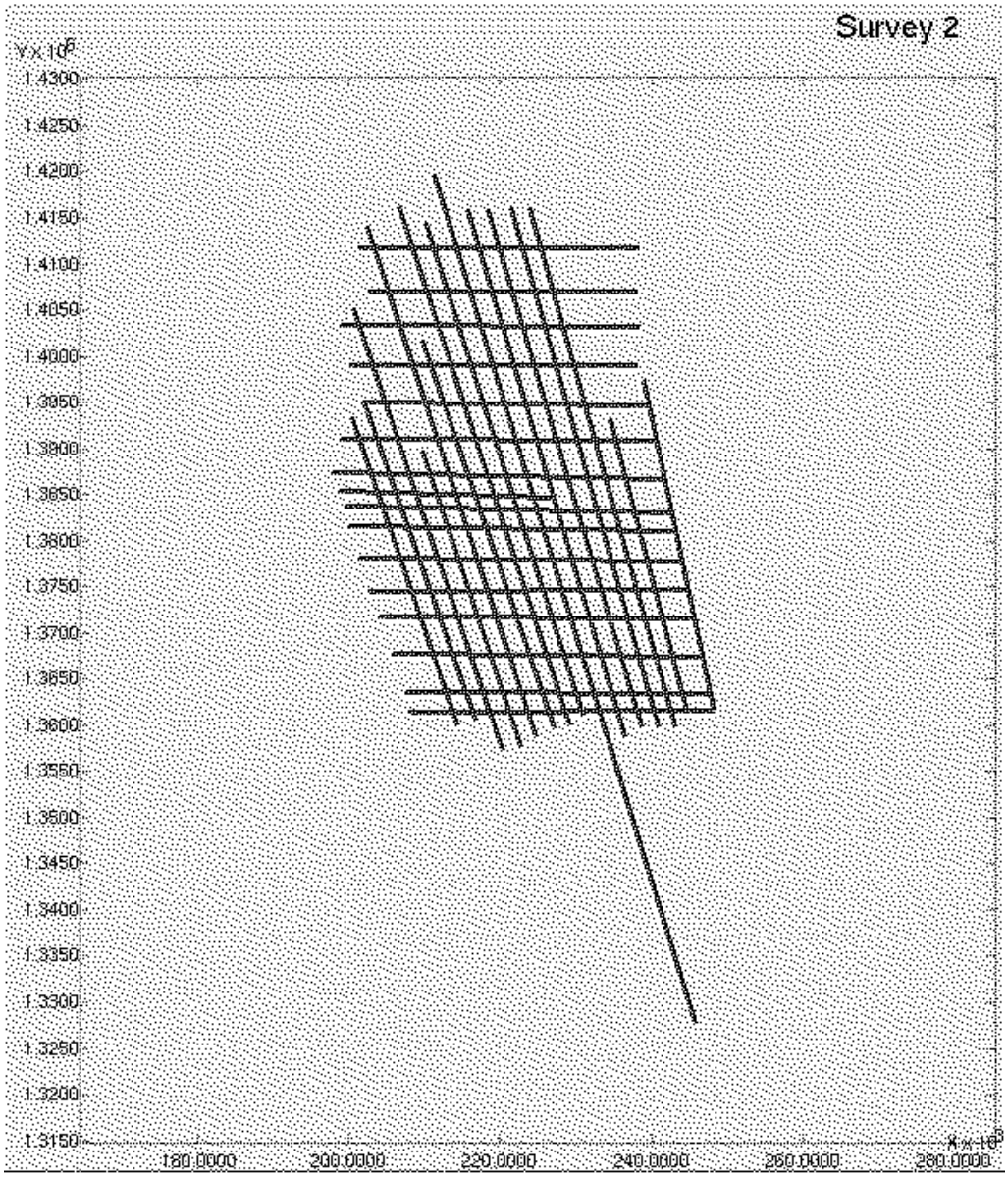

Figure 1: Example of a seismic survey

We use the following graph model to represent the problem. Survey lines are represented by a set of pairs of vertices. Each pair represents a survey line to be shot (the two vertices represent the two line endpoints). The two vertices in the pair $\{x, y\}$ can be connected by either

(a) one arc, (u, v) or (v, u) meaning that the corresponding line is only allowed to be shot in one direction; or

(b) by a pair of arcs, (u, v) and (v, u), meaning that the survey line can be shot in either direction.

The cost of shooting a line is taken to be constant and invariant to the direction in which the line is shot. Since every line has to be shot exactly once, the overall cost of shooting the lines is constant and can therefore be ignored. Hence, we take the cost of arcs connecting vertices in a pair to be zero.

The cost of an arc (x, y) that connects two pairs represents the time required to move the vessel from the final shooting position on the source line to the starting position on the destination line. Various approximation schemes can be used to estimate the amount of time required to change lines. A very basic scheme can be based solely on Euclidean distances between points, whereas more elaborate techniques can take into account vessel velocity, bearing and/or any potential obstacles.

There are two special vertices, s and t, that denote the starting and the required final position of the vessel (the latter is optional). The cost of arc (s, u) is taken to be the cost of moving the vessel from the initial position to point u. The cost of arc (v, t) is taken to be the cost of moving the vessel from point v to the final position. If no final position is specified, the cost of (v, t) is set to zero for all v. Finally, to close the tour, the vertices s and t are connected with the arc (t, s) of zero cost.

Given the above directed graph representation, the aim is to find a Hamilton cycle of minimum cost that visits each pair of vertices, including s and t exactly once. That is, once the cycle enters pair $\{u, v\}$ at u (or v), it must visit vertex v (or u) before leaving the pair.

A digraph H is *complete* if, for every pair x, y of distinct vertices, the arcs (x, y) and (y, x) are in D. A digraph $D = (V, A)$ is *weighted* if any arc of D is assigned a non-negative weight (called *cost* in this paper). More formally, SVP can be stated as follows: We are given a weighted complete digraph D, whose vertices are partitioned into pairs P. Each pair $\{u, v\} \in P$ is assigned a set F_{uv} such that $\emptyset \neq F_{uv} \subseteq \{(u, v), (v, u)\}$. Let $F = \{(u, v) \in A : \ (u, v) \in F_{uv}\}$. Every arc in F is assigned cost zero. We are required

to find a minimum cost Hamilton cycle that traverses one arc from F_{uv} for every pair $\{u, v\} \in P$.

Notice that the case when some arcs in F has to be of non-zero cost can be easily transformed into the case when all arcs in F are of cost zero by adding the cost of every arc $(u, v) \in F$ to the costs of all arcs of the form (x, u).

The Stacker Crane Problem (SCP) studied in [4, 6, 7] is a special case of SVP. In SCP, F_{uv} consists of one arc for every pair $\{u, v\} \in P$. To see that SCP generalizes the Asymmetric Travelling Salesman Problem (TSP) it suffices to contract all arcs of F.

2 Solving SVP

SVP is a new problem, which clearly generalizes the Asymmetric TSP and, thus, the Symmetric TSP. As with other extensions of TSP, the most practical approach for obtaining near-optimal or optimal solutions in the case of instances of moderate size seems to apply a transformation from the problem in hand to TSP and to subsequently use already developed exact algorithms or heuristics for TSP. This approach has proved to be successful for the so-called Generalized TSP [2, 3, 9].

The Asymmetric Generalized TSP can be transformed into the Asymmetric TSP [12]. In [2, 3], the resulting instances of the Asymmetric TSP were subsequently transformed into instances of the Symmetric TSP. This allows one to apply standard exact and heuristic methods to solving the transformed problem. Transformation into the Symmetric TSP is a frequently-used technique for tackling the Asymmetric variant of the problem [7, 8, 10].

Consider SVP. In order to enforce the requirement that a Hamilton cycle has to traverse one arc in F_{uv} for each pair $\{u, v\} \in P$, we apply a transformation which results in a weighted undirected complete graph. Solving the Symmetric TSP on the transformed graph provides a solution to the original problem.

The transformation replaces each pair $\{u, v\} \in P$ with a graph. Depending on whether there is one, or two, arcs of F connecting the vertices in the pair, two different transformations are used.

1. When the line $\{u, v\}$ is directed, a single arc (u, v) of F_{uv} is replaced with the edge $\{u, v\}$. A new vertex x is inserted into the edge. The costs of the two new edges, $\{u, x\}$ and $\{x, v\}$, are taken to be zero.

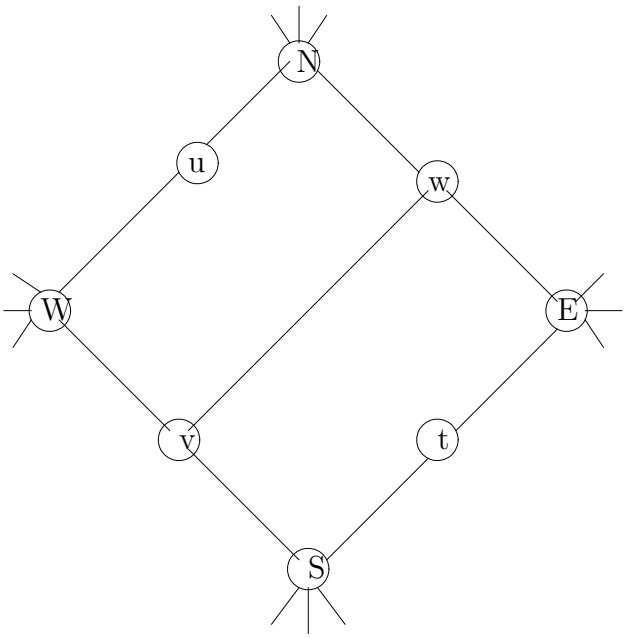

Figure 2: Diamond graph

2. When the line $\{u, v\}$ is undirected (that is, $|F_{uv}| = 2$), the two vertices are replaced with the so-called diamond graph depicted in Figure 2.

   The diamond graph can be traversed in two possible ways, N-S and W-E (see Chapter 19 in [13]). These correspond to traversing the original pair of vertices, $\{u, v\}$ via arcs (u, v) and (v, u), respectively. To make the cost of the tour consistent with the original graph:

   - We set the cost of edges incident to W to be the same as the cost of the corresponding original arcs entering vertex v;
   - Cost of edges incident to E are taken to be the same as cost of arcs leaving u;
   - Cost of edges incident to N are taken to be the same as cost of arcs entering u;
   - Cost of edges incident to S are taken to be the same as cost of arcs leaving v;
   - Since arcs (u, v) and (v, u) have zero cost, all edges inside the diamond graph have their costs set to 0.

Inst.	Size	concorde/opt	concorde/lk	neto/lk	NN
1	56	495773.7	508628.9	519517.8	503119.4
2	56	345077.6	361656.7	414531.7	431499.2
3	64	549270.6	575649.7	603474.6	633979.2
4	64	389559.8	409760.2	424234.3	431809.2
5	22	447343.8	458339.9	477562.0	491964.3
6	22	306119.3	309956.8	308386.9	357591.0
7	132	858549.8	923923.0	972181.4	1202892.4
8	132	745390.0	849419.2	FAILED	1048438.6
9	70	236887.9	245084.6	254901.8	419049.9

Table 1: Absolute solution quality (weight of the solution)

Finally, we complement the graph to a complete graph by adding all missing edges and setting their costs to a large constant, which ensures such edges can never be chosen as part of an optimal tour.

3 Computational Experience

Our experiments were performed on nine real-world instances supplied by Veritas DGC Ltd. Table 1 provides the sizes of the instances. In every instance of Table 1 all lines but one are undirected.

We used a nearest neighbor algorithm (NN) adapted to SVP to find a 'good' feasible solution for the instances. NN is of interest because this approach is somewhat similar to what people with no knowledge of combinatorial optimization would likely do to approximately solve SVP.

CONCORDE [1] is the most developed and popular software package devoted to solving instances of the Symmetric TSP. We applied the diamond transformation to the nine instances of SVP and then used the branch-and-cut solver of CONCORDE (concorde/opt), its chained Lin-Kernighan local search algorithm (concorde/lk; see [7, 14]) as well as Neto's Lin-Kernighan algorithm implementation [11] (neto/lk). The results are shown in Table 1. The data in this form are difficult to analyze.

Table 2 shows how each algorithm performed compared to NN. The numbers show percentage improvement in solution quality compared to NN. Negative numbers mean that the algorithm has performed worse than the NN. It can be seen from Table 2 that concorde/opt and both implementations of Lin-Kernighan offer a substantial improvement in solution quality

Inst.	Size	concorde/opt	concorde/lk	neto/lk
1	56	1.46%	-1.10%	-3.26%
2	56	20.03%	16.19%	3.93%
3	64	13.36%	9.20%	4.81%
4	64	9.78%	5.11%	1.75%
5	22	9.07%	6.83%	2.93%
6	22	14.39%	13.32%	13.76%
7	132	28.63%	23.19%	19.18%
8	132	28.90%	18.98%	FAILED
9	70	43.47%	41.51%	39.17%

Table 2: Solution quality relative to NN (% improvement over NN)

over NN. In some cases solution quality is improved by over 40%.

In one case neto/lk fails to find a feasible solution altogether (in Tables 1-4 this case is denoted by FAILED). This is due to the fact that unfortunately neto/lk managed to produce only a tour of the Symmetric TSP that contains edges of very large cost and, thus, does not correspond to any feasible solution of SVP. The other heuristic algorithm, concorde/lk performed more consistently in our experience, producing feasible solutions to all nine instances.

It is interesting to note that on Instance 1 both implementations of Lin-Kernighan performed somewhat worse than NN, by 1.1%-3.3%. One can see that this is due to NN performing very well in this instance, rather than Lin-Kernighan performing poorly. Notice that the heuristics of neto/lk and concorde/lk that produce initial solutions are not NN (they are different versions of the greedy algorithm).

Table 3 shows execution time of each solver on each of the test problems. The times are in seconds and have been obtained on a PC equipped with one Athlon XP 1900+ (1.6GHz) CPU and 512MB of RAM.

NN is the fastest algorithm. It found solutions to each of the test problems within few hundredths of a second (note that 0.016s is the smallest measurable interval on the test platform). Both implementations of the Lin-Kernighan algorithm require similar amounts of time, taking between 0.1s and 1.6s. As expected, CONCORDE's branch-and-cut solver, concorde/opt, is much slower than all other algorithms on test. Being an exact algorithm, it may require exponential execution time in the worst case. On the nine instances tested in this study, the algorithm requires between 0.5s and four

Inst.	Size	concorde/opt	concorde/lk	neto/lk	NN
1	56	1.372	0.388	0.388	<0.016
2	56	1.403	0.451	0.373	0.016
3	64	223.420	0.513	0.388	0.016
4	64	9.358	0.514	0.498	<0.016
5	22	0.561	0.170	0.108	<0.016
6	22	7.529	0.186	0.171	<0.016
7	132	204.622	1.358	1.311	0.016
8	132	99.639	1.529	FAILED	0.016
9	70	2.091	0.701	0.436	0.016

Table 3: Execution time (seconds)

minutes of execution time.

4 Conclusions

We have seen that the suggested transformation of SVP into the Symmetric TSP allows one to solve some practical SVP instances of moderate size without investing much effort on developing special SVP solvers.

Certainly, the use of CONCORDE's exact solver appears to be the best option for practical solution of SVP instances of moderate size. However, for large size instances or when the data are not exact (as in many industrial applications), a combination of CONCORDE's LK algorithm and NN seems to be a good option, too.

Acknowledgements We thank Sam Borman, Damian Hite and Larry Scott from Veritas DGC Ltd for several discussions on the topic.

References

[1] D. Applegate, R.E. Bixby, V. Chvátal and W. Cook, On the Solution of Traveling Salesman Problems, Doc. Math. J. DMV, Extra Vol. ICM Berlin 1998, III (1998) 645-656. The `Concorde` code is currently available from `http://www.math.princeton.edu/tsp/concorde.html`

[2] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo and A. Zverovitch, Process planning for rotational parts and the generalized Traveling Salesman Problem. International Journal of Production Research 41 (2003) 2581-2596.

[3] D. Ben-Arieh, G. Gutin, M. Penn, A. Yeo and A. Zverovitch, Transformations of generalized ATSP into ATSP. Operations Research Letters 31 (2003) 357–365.

[4] G.N. Frederickson, M.S. Hecht and C.E. Kim, Approximation algorithms for some routing problems. SIAM J. Comp. 7 (1978), 178-193.

[5] G. Gutin, A. Yeo and A. Zverovich, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP. Discrete Appl. Math. 117 (2002) 81-86.

[6] R. Hassin and S. Khuller, z-Approximations. J. Algorithms 41 (2001), 429-442.

[7] D.S. Johnson, G. Gutin, L. McGeoch, A. Yeo, X. Zhang and A. Zverovitch, Experimental Analysis of Heuristics for ATSP. in: The Traveling Salesman Problem and its Variations (G. Gutin and A. Punnen, eds.), Kluwer, Dordrecht, 2002.

[8] D.S. Johnson and L. McGeoch, Experimental Analysis of Heuristics for STSP. in: The Traveling Salesman Problem and its Variations (G. Gutin and A. Punnen, eds.), Kluwer, Dordrecht, 2002.

[9] G. Laporte and F. Semet, Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem. INFOR (37) 1999 114–120.

[10] D. Naddef, Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP. in: The Traveling Salesman Problem and its Variations (G. Gutin and A. Punnen, eds.), Kluwer, Dordrecht, 2002.

[11] D. Neto, http://www.cs.toronto.edu/~neto/research/lk/

[12] C.E. Noon and J.C. Bean, An efficient transformation of the generalized traveling salesman problem, INFOR 31 (1993) 39–44.

[13] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization. Dover Publ., N.Y., 1998.

[14] C. Rego and F. Glover, Local Search and Metaheuristics. in: The Traveling Salesman Problem and its Variations (G. Gutin and A. Punnen, eds.), Kluwer, Dordrecht, 2002.