

# TRAVELLING SALESPERSON PROBLEM

GERRIT RENKER

## 1. PROBLEM DESCRIPTION

This is a classical *NP*-complete problem and has been studied so extensively in the past thirty years that we shall only discuss how to model the problem, which in its basic variant is described as follows. Given a set of cities (nodes in a complete graph) and a function which maps each two cities appearing as adjacent nodes into a distance value (label of the connecting edge), construct a Hamiltonian cycle such that the accumulated sum of distances reaches a minimal value. Table 1 (which is taken from [3, sec. 4.5]) presents instance data for the distances in a  $10 \times 10$  variant of the problem.

	A	B	C	D	E	F	G	H	I	J
A		24	18	22	31	19	33	25	30	26
B	15		19	27	26	32	25	31	28	18
C	22	23		23	16	29	27	18	16	27
D	24	31	18		19	13	28	9	19	27
E	23	18	34	20		31	24	15	25	8
F	24	12	17	15	10		11	16	21	31
G	28	15	27	35	19	18		21	21	19
H	13	24	18	13	13	22	25		29	24
I	17	21	18	24	27	24	34	31		18
J	18	19	29	16	23	17	18	31	23	

TABLE 1. Instance data for the  $10 \times 10$  travelling salesperson problem

## 2. Z MODEL

**2.1. Data modelling.** As usual, we begin with constants that are global to the specification. The number  $n$  determines the size of the problem instance. The set *cities* is then defined as an Integer range.

$$\begin{array}{|l} n : \mathbb{N} \\ \text{cities} : \mathbb{F} \mathbb{N} \\ \hline \text{cities} = 1 \dots n \end{array}$$

**2.2. Instance data.** The matrix of given distances (cf. table 1) is specified as a function of global scope.

$$\begin{array}{|l} \text{distance} : (\text{cities} \times \text{cities}) \rightarrow \mathbb{N} \end{array}$$

---

Date: February 2004.

**2.3. Problem constraints.** We here make use of the `circuit` global constraint from our toolkit, where `route` is our main decision variable. We thus get the following, succinct schema.

<i>TSP</i>
<code>route : seq cities</code>
<code>circuit route</code>

**2.4. Optimisation part.** The schema is according to our template for optimisation problems, with the minor addition that we first have to convert the sequence (`route`) of `cities` into numerical distance values, which we achieve by means of the function `evaluate`. The result of this recursively defined function is a sequence of distance values corresponding to each two consecutive cities en route. We map the pathological case of less than three cities into the value  $\langle 0 \rangle$  as there is nothing to optimise. In all other cases a sequence is constructed whose first element contains the first distance value, concatenated with evaluating the remainder of the sequence. The latter is referred to via the sequence extraction operator  $\upharpoonright$  [4, p. 118].

<i>TSP_Optimisation_Part</i>
<code>evaluate : seq <math>\mathbb{N}</math> <math>\rightarrow</math> seq <math>\mathbb{N}</math></code>
<code>objective : TSP <math>\rightarrow</math> <math>\mathbb{N}</math></code>
<code>solution : TSP</code>
$\forall \sigma : \text{seq } \mathbb{N}; \text{len} : \mathbb{N} \mid \text{len} = \#\sigma \bullet$ $\text{evaluate}(\sigma) = \text{if } \text{len} \leq 2 \text{ then } \langle 0 \rangle \text{ else } \langle \text{distance}(\sigma \ 1, \sigma \ 2) \rangle \frown \text{evaluate}((3 \dots \text{len}) \upharpoonright \sigma)$
$\forall \text{tsp} : \text{TSP} \bullet$ $\text{objective}(\text{tsp}) = \text{sumseq}(\text{evaluate}(\text{tsp}.\text{route}))$ $\text{objective}(\text{solution}) = \text{min}(\text{objective}(\text{TSP}))$

In the definition of the `objective` function we sum all (possibly repeating) range values of `distance`, by applying the `sumseq` operator (cf. Appendix). Lastly, we specify that a `solution` would minimize the value of the `objective` function.

### 3. REFERENCES

The problem is discussed in many general textbooks, with respect to the constraint solving perspective there is for instance [1], which proposes optimized algorithms to solve small instances of the problem. It is worthwhile to compare the Z model of section 2 with the ALICE program in [3, p. 111]. A comprehensive TSP benchmark library is TSPLIB, at <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>

Closely related to the TSP is the vehicle routing problem (VRP), which in fact is a generalisation of the TSP; a TSP can be regarded as a VRP with only one vehicle. There are many variations of both problem types, Ilog code for solving several VRP/TSP problems is e.g. documented in [2, chap. 23].

### REFERENCES

- [1] Yves Caseau and François Laburthe. Solving Small TSPs with Constraints. In Lee Naish, editor, *Proceedings of the Fourteenth International Conference on Logic Programming (ICLP-97)*, pages 316–330. MIT Press, 1997.
- [2] ILOG, France. *Ilog Solver 4.4, User's Manual*, May 1999.
- [3] Jean-Louis Laurière. A Language and a Program for Stating and Solving Combinatorial Problems. *Artificial Intelligence*, 10(1):29–127, February 1978.
- [4] J. M. Spivey. *The Z Notation: A Reference Manual*. J. M. Spivey, Oriel College, Oxford OX1 4EW, second edition, 1998. First published 1992 by Prentice Hall, current version on <http://spivey.oriel.ox.ac.uk/mike/zrm/>.