1. The file format

Each file consists of a specification part and of a data part. The specification part contains information on the file format and on its contents. The data part contains explicit data.

1.1 The specification part

All entries in this section are of the form <keyword> : <value>, where <keyword> denotes an alphanumerical keyword and <value> denotes alphanumerical or numerical data. The terms <string>, <integer> and <real> denote character string, integer or real data, respectively. The order of specification of the keywords in the data file is arbitrary (in principle), but must be consistent, i.e., whenever a keyword is specified, all necessary information for the correct interpretation of the keyword has to be known. Below we give a list of all available keywords.

1.1.1 NAME : <string>

Identifies the data file.

1.1.2 TYPE : <string>

Specifies the type of the data. Possible types are

- TSP   Data for a symmetric traveling salesman problem
- ATSP  Data for an asymmetric traveling salesman problem
- SOP   Data for a sequential ordering problem
- HCP   Hamiltonian cycle problem data
- CVRP  Capacitated vehicle routing problem data
- TOUR  A collection of tours

1.1.3 COMMENT : <string>

Additional comments (usually the name of the contributor or creator of the problem instance is given here).

1.1.4 DIMENSION : <integer>

For a TSP or ATSP, the dimension is the number of its nodes. For a CVRP, it is the total number of nodes and depots. For a TOUR file it is the dimension of the corresponding problem.

1.1.5 CAPACITY : <integer>

Specifies the truck capacity in a CVRP.

1.1.6 EDGE_WEIGHT_TYPE : <string>

Specifies how the edge weights (or distances) are given. The values are

- EXPLICIT Weights are listed explicitly in the corresponding section
- EUC_2D   Weights are Euclidean distances in 2-D
- EUC_3D   Weights are Euclidean distances in 3-D
MAX_2D  Weights are maximum distances in 2-D
MAX_3D  Weights are maximum distances in 3-D
MAN_2D  Weights are Manhattan distances in 2-D
MAN_3D  Weights are Manhattan distances in 3-D
CEIL_2D  Weights are Euclidean distances in 2-D rounded up
GEO  Weights are geographical distances
ATT  Special distance function for problems att48 and att532
XRAY1  Special distance function for crystallography problems (Version 1)
XRAY2  Special distance function for crystallography problems (Version 2)
SPECIAL  There is a special distance function documented elsewhere

1.1.7  EDGE_WEIGHT_FORMAT : <string>

Describes the format of the edge weights if they are given explicitly. The values are

FUNCTION  Weights are given by a function (see above)
FULL MATRIX  Weights are given by a full matrix
UPPER_ROW  Upper triangular matrix (row-wise without diagonal entries)
LOWER_ROW  Lower triangular matrix (row-wise without diagonal entries)
UPPER_DIAG_ROW  Upper triangular matrix (row-wise including diagonal entries)
LOWER_DIAG_ROW  Lower triangular matrix (row-wise including diagonal entries)
UPPER_COL  Upper triangular matrix (column-wise without diagonal entries)
LOWER_COL  Lower triangular matrix (column-wise without diagonal entries)
UPPER_DIAG_COL  Upper triangular matrix (column-wise including diagonal entries)
LOWER_DIAG_COL  Lower triangular matrix (column-wise including diagonal entries)

1.1.7  EDGE_DATA_FORMAT : <string>

Describes the format in which the edges of a graph are given, if the graph is not complete. The values are

EDGE_LIST  The graph is given by an edge list
ADJ_LIST  The graph is given as an adjacency list

1.1.9  NODE_COORD_TYPE : <string>

Specifies whether coordinates are associated with each node (which, for example may be used for either graphical display or distance computations). The values are

TWOD_COORDS  Nodes are specified by coordinates in 2-D
THREED_COORDS  Nodes are specified by coordinates in 3-D
NO_COORDS  The nodes do not have associated coordinates

The default value is NO_COORDS.

1.1.10  DISPLAY_DATA_TYPE : <string>

Specifies how a graphical display of the nodes can be obtained. The values are

COORD_DISPLAY  Display is generated from the node coordinates
TWOD_DISPLAY  Explicit coordinates in 2-D are given
NO_DISPLAY  No graphical display is possible

The default value is COORD_DISPLAY if node coordinates are specified and NO_DISPLAY otherwise.
1.1.11 EOF:
Terminates the input data. This entry is optional.

1.2 The data part

Depending on the choice of specifications some additional data may be required. These
data are given in corresponding data sections following the specification part. Each data
section begins with the corresponding keyword. The length of the section is either implicitly
known from the format specification, or the section is terminated by an appropriate end-
of-section identifier.

1.2.1 NODE_COORD_SECTION:

Node coordinates are given in this section. Each line is of the form

\(<\text{integer}>\ <\text{real}>\ <\text{real}>\)

if NODE_COORD_TYPE is TWOD_COORDS, or

\(<\text{integer}>\ <\text{real}>\ <\text{real}>\ <\text{real}>\)

if NODE_COORD_TYPE is THREED_COORDS. The integers give the number of the respective
nodes. The real numbers give the associated coordinates.

1.2.2 DEPOT_SECTION:

Contains a list of possible alternate depot nodes. This list is terminated by a \(-1\).

1.2.3 DEMAND_SECTION:

The demands of all nodes of a CVRP are given in the form (per line)

\(<\text{integer}>\ <\text{integer}>\)

The first integer specifies a node number, the second its demand. The depot nodes must
also occur in this section. Their demands are 0.

1.2.4 EDGE_DATA_SECTION:

Edges of a graph are specified in either of the two formats allowed in the EDGE_DATA_FORMAT
entry. If the type is EDGE_LIST, then the edges are given as a sequence of lines of the form

\(<\text{integer}>\ <\text{integer}>\)

each entry giving the terminal nodes of some edge. The list is terminated by a \(-1\).

If the type is ADJ_LIST, the section consists of a list of adjacency lists for nodes. The
adjacency list of a node \(x\) is specified as

\(<\text{integer}>\ <\text{integer}>\ldots <\text{integer}>\ -1\)

where the first integer gives the number of node \(x\) and the following integers (terminated
by \(-1\) ) the numbers of nodes adjacent to \(x\). The list of adjacency lists is terminated by
an additional \(-1\).
1.2.5  FIXED_EDGES_SECTION:

In this section, edges are listed that are required to appear in each solution to the problem. The edges to be fixed are given in the form (per line)

\(<\text{integer}>\ \text{<integer>}\)

meaning that the edge (arc) from the first node to the second node has to be contained in a solution. This section is terminated by a -1.

1.2.6  DISPLAY_DATA_SECTION:

If DISPLAY_DATA_TYPE is TWOD_DISPLAY, the 2-dimensional coordinates from which a display can be generated are given in the form (per line)

\(<\text{integer}>\ \text{<real>}\ \text{<real>}\)

The integers specify the respective nodes and the real numbers give the associated coordinates.

1.2.7  TOUR_SECTION:

A collection of tours is specified in this section. Each tour is given by a list of integers giving the sequence in which the nodes are visited in this tour. Every such tour is terminated by a -1. An additional -1 terminates this section.

1.2.8  EDGE_WEIGHT_SECTION:

The edge weights are given in the format specified by the EDGE_WEIGHT_FORMAT entry. At present, all explicit data is integral and is given in one of the (self-explanatory) matrix formats. with implicitly known lengths.
2. The distance functions

For the various choices of \texttt{EGDE\_WEIGHT\_TYPE}, we now describe the computations of the respective distances. In each case we give a (simplified) C-implementation for computing the distances from the input coordinates. All computations involving floating-point numbers are carried out in double precision arithmetic. The integers are assumed to be represented in 32-bit words. Since distances are required to be integral, we round to the nearest integer (in most cases). Below we have used the rounding function “\texttt{nint}”.

2.1 Euclidean distance ($L_2$-metric)

For edge weight type \texttt{EUC\_2D} and \texttt{EUC\_3D}, floating point coordinates must be specified for each node. Let $x[i]$, $y[i]$, and $z[i]$ be the coordinates of node $i$.

In the 2-dimensional case the distance between two points $i$ and $j$ is computed as follows:

\[
\begin{align*}
xd &= x[i] - x[j]; \\
yd &= y[i] - y[j]; \\
dij &= \text{nint}(\sqrt{xd\times xd + yd\times yd});
\end{align*}
\]

In the 3-dimensional case we have:

\[
\begin{align*}
xd &= x[i] - x[j]; \\
yd &= y[i] - y[j]; \\
zd &= z[i] - z[j]; \\
dij &= \text{nint}(\sqrt{xd\times xd + yd\times yd + zd\times zd});
\end{align*}
\]

where $\sqrt{\cdot}$ is the C square root function.

2.2 Manhattan distance ($L_1$-metric)

Distances are given as Manhattan distances if the edge weight type is \texttt{MAN\_2D} or \texttt{MAN\_3D}. They are computed as follows.

2-dimensional case:

\[
\begin{align*}
xd &= \text{abs}(x[i] - x[j]); \\
yd &= \text{abs}(y[i] - y[j]); \\
dij &= \text{nint}(xd + yd);
\end{align*}
\]

3-dimensional case:

\[
\begin{align*}
xd &= \text{abs}(x[i] - x[j]); \\
yd &= \text{abs}(y[i] - y[j]); \\
zd &= \text{abs}(z[i] - z[j]); \\
dij &= \text{nint}(xd + yd + zd);
\end{align*}
\]

2.3 Maximum distance ($L_{\infty}$-metric)

Maximum distances are computed if the edge weight type is \texttt{MAX\_2D} or \texttt{MAX\_3D}.

2-dimensional case:

\[
\begin{align*}
xd &= \text{abs}(x[i] - x[j]); \\
yd &= \text{abs}(y[i] - y[j]); \\
dij &= \text{max}(\text{nint}(xd), \text{nint}(yd));
\end{align*}
\]
3-dimensional case:
\[
\begin{align*}
xd &= \text{abs}( x[i] - x[j] ); \\
yd &= \text{abs}( y[i] - y[j] ); \\
zd &= \text{abs}( z[i] - z[j] ); \\
dij &= \max( \text{nint}( xd ), \text{nint}( yd ), \text{nint}( zd ) );
\end{align*}
\]

2.4 Geographical distance

If the traveling salesman problem is a geographical problem, then the nodes correspond to points on the earth and the distance between two points is their distance on the idealized sphere with radius 6378.388 kilometers. The node coordinates give the geographical latitude and longitude of the corresponding point on the earth. Latitude and longitude are given in the form DDD-MM where DDD are the degrees and MM the minutes. A positive latitude is assumed to be “North”, negative latitude means “South”. Positive longitude means “East”, negative latitude is assumed to be “West”. For example, the input coordinates for Augsburg are 48.23 and 10.53, meaning 48°23’ North and 10°53’ East.

Let \( x[i] \) and \( y[i] \) be coordinates for city \( i \) in the above format. First the input is converted to geographical latitude and longitude given in radians.

\[
\begin{align*}
\text{PI} &= 3.141592; \\
\text{deg} &= \text{nint}( x[i] ); \\
\text{min} &= x[i] - \text{deg}; \\
\text{latitude}[i] &= \text{PI} \times (\text{deg} + 5.0 \times \text{min} / 3.0) / 180.0; \\
\text{deg} &= \text{nint}( y[i] ); \\
\text{min} &= y[i] - \text{deg}; \\
\text{longitude}[i] &= \text{PI} \times (\text{deg} + 5.0 \times \text{min} / 3.0) / 180.0;
\end{align*}
\]

The distance between two different nodes \( i \) and \( j \) in kilometers is then computed as follows:

\[
\begin{align*}
\text{RRR} &= 6378.388; \\
q1 &= \cos( \text{longitude}[i] - \text{longitude}[j] ); \\
q2 &= \cos( \text{latitude}[i] - \text{latitude}[j] ); \\
q3 &= \cos( \text{latitude}[i] + \text{latitude}[j] ); \\
dij &= (\text{int})( \text{RRR} \times \text{acos}( 0.5 \times (q1 + q2 - (1.0 - q1) \times q3) ) + 1.0);
\end{align*}
\]

The function “acos” is the inverse of the cosine function.

2.5 Pseudo-Euclidean distance

The edge weight type ATT corresponds to a special “pseudo-Euclidean” distance function. Let \( x[i] \) and \( y[i] \) be the coordinates of node \( i \). The distance between two points \( i \) and \( j \) is computed as follows:

\[
\begin{align*}
xd &= x[i] - x[j]; \\
yd &= y[i] - y[j]; \\
rij &= \text{sqrt}( (xd \times xd + yd \times yd) / 10.0 ); \\
tij &= \text{nint}( rij ); \\
\text{if } (tij < rij) \text{ dij = tij + 1;} \\
\text{else dij = tij;}
\end{align*}
\]
2.6 Ceiling of the Euclidean distance

The edge weight type CEIL_2D requires that the 2-dimensional Euclidean distances is rounded up to the next integer.

2.7 Distance for crystallography problems

We have included into TSPLIB the crystallography problems as described in [1]. These problems are not explicitly given but subroutines are provided to generate the 12 problems mentioned in this reference and subproblems thereof (see section 3.2).
To compute distances for these problems the movement of three motors has to be taken into consideration. There are two types of distance functions: one that assumes equal speed of the motors (XRAY1) and one that uses different speeds (XRAY2). The corresponding distance functions are given as FORTRAN implementations (files deq.f, resp. duneq.f) in the distribution file.
For obtaining integer distances, we propose to multiply the distances computed by the original subroutines by 100.0 and round to the nearest integer.
We list our modified distance function for the case of equal motor speeds in the FORTRAN version below.

```
INTEGER FUNCTION ICOST(V,W)
  INTEGER V, W
  DOUBLE PRECISION DMIN1, DMAX1, DABS
  DOUBLE PRECISION DISTP, DISTC, DISTT, COST
  DISTP = DMIN1(DABS(PHI(V) - PHI(W)), DABS(DABS(PHI(V) - PHI(W)) - 360.0E+0))
  DISTC = DABS(CHI(V) - CHI(W))
  DISTT = DABS(TWOTH(V) - TWOTH(W))
  COST = DMAX1(DISTP/1.00E+0, DISTC/1.00E+0, DISTT/1.00E+0)
C  *** Make integral distances ***
  ICOST = AINT(100.0E+0*COST + 0.5E+0)
RETURN
END
```

The numbers PHI(), CHI(), and TWOTH() are the respective x-, y-, and z-coordinates of the points in the generated traveling salesman problems. Note, that TSPLIB95 contains only the original distance computation without the above modification.

2.7 Verification

To verify correctness of the distance function implementations we give the length of some “canonical” tours 1, 2, 3, …, n.
The canonical tours for pcb442, gr666, and att532 have lengths 221.440, 423.710, and 309.636, respectively.
The canonical tour for the problem xray14012 (the 8th problem considered in [21]) with distance XRAY1 has length 15429.219. With distance XRAY2 it has the length 12943.294.