TSP in Spreadsheets – a Guided Tour

Rasmus Rasmussen

Abstract

The travelling salesman problem (TSP) is a well-known business problem, and variants like the *m*aximum benefit TSP or the price collecting TSP may have numerous economic applications. We are looking at several different variants of TSP; all solved in spreadsheets, not using tailored solvers for TSP. As these problems are NP hard, solving those using standard LP/MIP solvers has been regarded feasible only for very small sized problems. However, a careful consideration of the spreadsheet layout may facilitate efficient software utilisation. For real world problems this can have considerable effects, and with the recent advancements in solver engines, problems previously regarded as big are now easily solvable in spreadsheets. This paper shows you how; and how the flexibility of spreadsheets makes it a convenient tool solving many variants of TSP, where tailored solvers simply would not fit.

JEL classification: C61, Z00

1. Introduction

After a formal statement of the problem, three different spreadsheet models will be illustrated. The flexibility of spreadsheets will also be demonstrated, as will how spreadsheet layout may help in making an efficient problem formulation, in addition to helping to clearly communicate and display the solution. The direct permutation approach is presented first, applying integer variables to describe the sequence of the visits. The direct permutation approach fits small problems well, and requires very little work after data has been obtained. No constraints to eliminate subtours are needed, but the problem is non-linear and non-smooth, requiring heuristic solvers. Second, a network formulation is presented, where binary variables are used to make a linear formulation of the problem. An efficient spreadsheet layout is presented for non-complete graphs. Thirdly an assignment formulation is presented, applying a spreadsheet layout more suitable for complete graphs.

Variants of TSP not fitting tailored TSP software are also solved. In addition some confining side effects of common subtour eliminating constraints are discussed, particularly when multiple visits are required.

2. The standard TSP

Travelling salesman problems (TSP) are easy to describe: a salesman needs to visit all his customers located in different cities in his region, and he would like to find the cheapest tour that will assure that all cities have been visited. Unfortunately TSP is not so easy to formulate, and relatively hard to solve. When making a mathematical formulation of these problems we will for the most part use a network framework. The cities are then called nodes, and the roads connecting the cities are called arcs. See Gutin and Punnen (2007) for a full treatment of TSP and its variants.

The set of nodes to be visited are defined as $N = \{1, 2, ..., n\}$ where n is the total number of nodes (referred to as the *size* of a TSP), and the set of arcs connecting the nodes is defined as $A = \{(i,j) : i, j \in N,$

$i \neq j\}$, where the pair (*i,j*) indicates the arc between node *i* and *j*. A standard assumption in TSP is to assume *direct links between every pair of nodes*, usually referred to as a *complete graph*. The graph consisting of the nodes *N* and arcs *A* is then *connected*; there is a *connection* or path from any node to any other node in the graph. The basic standard assumption is to restrict the number of visits to exactly one for each node. Why the salesman is not allowed to visit a node more than once is not obvious. One can speculate that such a requirement makes it easier to develop solution procedures, thereby fitting the problem to the tools at hand. A common definition of the set of decision variables is $X \equiv \{x_{ij} : i, j \in N, i \neq j\}$ where $x_{ij} = 1$ if the salesman travels from node *i* to *j* (node *i* is visited immediately before node *j*), and *0* otherwise. The cost matrix is defined as $C = \{c_{ij} : i, j \in N, i \neq j\}$ and usually assumed to be positive, where c_{ij} represents the cost of traversing from node *i* to node *j*. In standard TSP a common assumption is that the square cost matrix is symmetric, $c_{ij} = c_{ji}$, the cost is the same in both directions. Another standard assumption is to assume the triangle inequality; $c_{ij} + c_{jk} \geq c_{ik} \ \forall i, j, k \in N$, the direct connection between two nodes is always the cheapest.

One basic assumption in TSP is to assume that the salesman has to return to the node where he starts the tour; this node is usually referred to as the *base city* or *depot*. This assumption is called a *closed tour*. For a closed tour any node can be selected as the starting node, but for practical reasons node 1 is set to be the starting node. *Node 1 is then the base city or depot*.

For a standard TSP there is always a feasible solution (as a complete graph is always connected), and we can choose any node to start (as the tour is closed and all nodes are visited). There are always alternative optimal solutions; the tour can go in either direction (as the costs are symmetric). And in the optimal tour(s) every node is visited only once (because of the triangle inequality, and the objective is always minimisation).

3. Variants of TSP

Quite a lot of real life problems do not fit these assumptions. Often we must allow for the set *A* not being complete, in cases where some nodes do not have direct links to all other nodes. Graphs that are not complete are no longer guaranteed to be *connected,* and for *disconnected graphs* there is *no feasible solution.* In real life we also have to allow for $c_{ij} \neq c_{ji}$, the cost of travelling from node *i* to *j* may not be the same as travelling from *j* to node *i.* This represents the *asymmetric travelling salesman problem (ATSP),* and implies *directed arcs.* Similarly it is not always cheapest to travel the direct link from node *i* to node *k,* sometimes it may be cheaper to travel via node *j.* Thus we must allow for the triangle inequality not to apply. The basic standard assumption to restrict the visits to exactly one for each node may also be skipped; TSP with multiple visits is referred to as TSPM, as in Gutin and Punnen (2007).

Of course the reason for the salesman to make the tour is to derive some benefit from visiting the nodes. Then let $B = \{b_j : j \in N\}$, where b_j is the benefit from visiting node *j.* For such problems we have the *maximum benefit travelling salesman problem (MBTSP);* see Malandraki and Daskin (1993). Another variant is the price collecting TSP (or PCTSP), see Gutin (2007).

Sometimes the salesman does not have to return to the base, and relaxing such a requirement is called an *open tour*. For an open tour it may be advantageous to be able to select the ending node as part of the problem solution, but this may increase the problem size for some types of formulations, except for the direct permutation approach.

There is a wide selection of literature on these problems, and several variants of problem formulations. We will group the formulations in two classes: the *assignment formulations* and the *flow formulations*. Further, in each group the models vary according to which assumptions are made, most notably whether a complete graph is assumed.

4. Assignment formulation of TSP

For the closed tour an assignment formulation could be of the following form:

$$Minimize \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ , \ \forall \, j \in N \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \ , \ \forall \, i \in N \qquad (3)$$

$$x_{ij} \in \{0,1\} \ , \ \forall \, i, j \in N \qquad (4)$$

In addition *subtour elimination constraints* (SECs) are needed. Constraints (2) and (3) are the standard assignment constraints. The objective in (1) will minimise the total cost along all the arcs used to complete the tour. However, as written this formulation assumes a complete graph, and if the data are being arranged in a square matrix will also include the diagonal. For a complete graph the only arcs that do not exist are related to the *self-loop variable* $x_{i,i}$ (along the diagonal). Therefore it usually is more convenient to exclude these variables by a new constraint (5), instead of excluding them in the definition of the *set X*. This convenience comes at the cost of increased problem size (both in terms of variables and constraints). For a complete graph the following constraint will fix the diagonal in a square *n* matrix of the binary variables x_{ij} equal to zero:

$$x_{i,i} = 0 \ \ \forall \ \ i \in \ N \qquad (5)$$

A different approach to rectify this, and allow for instances of non-complete graphs; is to set the cost c_{ij} sufficiently large for non-existing arcs, thereby preventing them from entering the final solution. However this is not a foolproof trick. In a *connected graph* there is a path from any node to any other node in the graph, and a complete graph is always connected, and thus has a feasible solution. Non-complete graphs may not be connected (*disconnected*), and will as such have no feasible solution. A high cost for non-existing arcs is then no guarantee for theses arcs to be excluded in the final solution. Therefore another strategy is to set a new parameter: $e_{ij} = 1$ if node *i* is directly connected to node *j*, otherwise 0; and replace constraint (5) with (6):

$$x_{ij} \le e_{ij} \ , \ \forall \, i, j \in N \qquad (6)$$

This formulation does not require a complete graph and allows for asymmetric costs and also for the triangle inequality not to apply, but unfortunately it has some flaws. If the assumption of a complete graph is not satisfied (and therefore constraint (6) is required), then a feasible solution for a closed tour may require some nodes to be visited twice, breaking the '=' requirement in constraint (2).

The limitation of visiting each node exactly once may also cause difficulties even for problems with a complete graph, if the triangle inequality is not satisfied. This requirement will effectively prohibit hub-like solutions, even when such solutions are the most cost-effective. A problem formulation that excludes such possible optimal solutions is generally not recommended.

5. Flow formulation of TSP

A flow formulation of the closed tour, that explicitly considers valid connections only, can be made after redefining $C = \{c_{i,j} : (i, j) \in A\}$ and $X \equiv \{x_{i,j} : (i, j) \in A\}$. This formulation will thus work even when the graph is not complete:

$$Minimize \sum_{(i,j)\in A} c_{i,j} \cdot x_{i,j} \qquad (7)$$

$$\sum_{i\,:\,(i,j)\in A} x_{i,j} \geq 1 \quad \forall \quad j \in N \qquad (8)$$

$$\sum_{i\,:\,(i,k)\in A} x_{i,k} = \sum_{j\,:\,(k,j)\in A} x_{k,j} \quad \forall\ k \ \in \ N \qquad (9)$$

$$x_{i,j} \in \{0,1\} \quad \forall \ (i, j) \in A \qquad (10)$$

Of course SECs are also required. The objective (7) will minimise the total cost of the tour, only considering valid arcs. Constraint (8) states that the salesman has to arrive each node at least once. Constraint (9) states that the salesman has to leave each node as many times as he arrive the node. By using '≥' instead of '=' in (8), we avoid the possibility of making the problem infeasible for non-complete graphs where some nodes need to be visited twice, and we do not exclude 'hub-like' optimal solutions if the triangle inequality does not apply.

6. Flow formulation of open tour TSP

For the open tour formulation we add the parameters $D = \{d_i : i \in N\}$ where d_i is the 'net demand' in node *i*; and $d_i = -1$ for the start node (the base city is numbered node 1); $d_i = +1$ for the end node, for the transit or intermediate nodes $d_i = 0$. The open tour formulation can then be stated as:

$$Minimize \sum_{(i,j)\in A} c_{i,j} \cdot x_{i,j} \qquad (11)$$

$$\sum_{i\,:\,(i,j)\in A} x_{i,j} \geq 1 \quad \forall\, j > 1 \ \in N \qquad (12)$$

$$\sum_{i\,:\,(i,k)\in A} x_{i,k} - \sum_{j\,:\,(k,j)\in A} x_{k,j} = d_k \quad \forall\ k \ \in \ N \qquad (13)$$

$$x_{i,j} \in \{0,1\} \quad \forall \ (i, j) \in A \qquad (14)$$

We must add SECs to complete the TSP formulation. The objective in (11) is identical to (7). Constraint (12) is similar to (8), except that we do not require the salesman to arrive/return to the starting node 1. Constraints (13) require the salesman to leave the starting node one more time than entering, enter the stopping node one more time than leaving, and leave any intermediate node as often as arriving the node. By removing constraint (12) we have the common *shortest path problem*. If the end node is not specified, the d_i parameters may be converted to binary variables (except for the start node), requiring their sum to equal 1.

7. Subtour eliminating constraints (SECs)

A key part of a TSP is to make sure the tour is continuous, that the arcs are linked from the base city all the way to every city visited. Without such constraints we quite often will get solutions containing degenerate tours between intermediate nodes and not connected to the base city. The originally SECs was formed in 1954 by Dantzig-Fulkerson-Johnson (DFJ) (see Dantzig, Fulkerson and Johnson, 1954):

$$\sum_{i\in S}\sum_{j\in S} x_{i,j} \le |S|-1, \quad \forall\ S \subseteq N \setminus \{1\},\ S \ne \varnothing \qquad (15)$$

Unfortunately this introduces an exponential number of constraints, and becomes impractical even for small sized problems. A different SEC proposed in 1960 by Miller-Tucker-Zemlin (MTZ) (see Miller, Tucker and Zemlin, 1960) introduces only a maximum[1] of *(n−2)*^{2} constraints, at the disadvantage of a weak LP relaxation:

$$u_i - u_j + 1 \le (n-1)\left(1-x_{i,j}\right) \quad \forall (i,j) \in A,\ :i,j \ne 1 \qquad (16)$$

In (16) a new set of variables $U = \{u_i : i \in N, i \ne 1\}$ is required. The u_i are arbitrary real numbers, but can be ranked to non-negative integers, representing the sequence the nodes are being visited. For convenience we may add $u_1 \equiv 1$ (node 1 is the base city), and limit the range of u_i, thus helping the optimisation software (see also Pataki (2003))*:*

$$2 \le u_i \le n \quad \forall\ i > 1\ \in N \qquad (17)$$

The MTZ SECs will be used in this paper, and have the following properties:

- node 1 is required to be the base city;
- they make sure that every city visited belongs to a tour connected to the base city, thereby eliminating subtours;
- they allow nodes to be visited more than once (unless other constraints prevent such a solution);
- they do not require all nodes being visited (unless other constraints make such requirements);
- they allow unidirectional arcs to be utilised in both directions on the same tour.

For a closed tour visiting all nodes the base node can always be chosen arbitrarily. A fundamental weakness of MTZ SECs is that feasibility and final solution may depend on which node is selected as the base city. The MTZ SECs may fail to find a feasible solution even if such exists, and they may fail to find the global optimal solution. Problems with feasibility may occur in non-complete graphs, where all feasible solutions require some nodes to be visited twice. Problems finding the global optimal solution may occur in complete graphs where the triangle inequality does not apply, and where the global optimal solution requires some nodes to be visited more than once. It is therefore important to be aware of these two situations where applying the MTZ SECs may make the final solution sensitive as to which node is selected as the starting node. They will never fail if the global optimal solution visits each node only once.

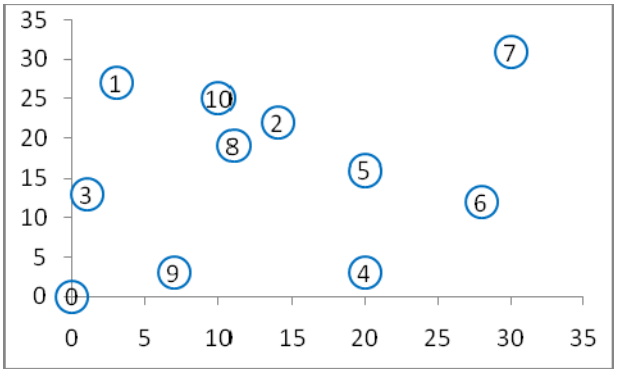
8. Closed TSP in a complete graph

As an example of a TSP in a complete graph we shall use the following example. A supply ship is serving 10 oil rigs at sea. The base is located at coordinates (0,0), and the rigs are located as displayed in Figure

[1] In a square *n×n* matrix; the first row, first column and the diagonal are excluded.

1. Assuming open sea the distances between any pair of nodes (oil rigs) can be calculated as straight lines (ignoring the fact that the sea level is not flat). This is a standard symmetric TSP with a complete graph where the triangle inequality applies. Data is taken from Ragsdale (2001).

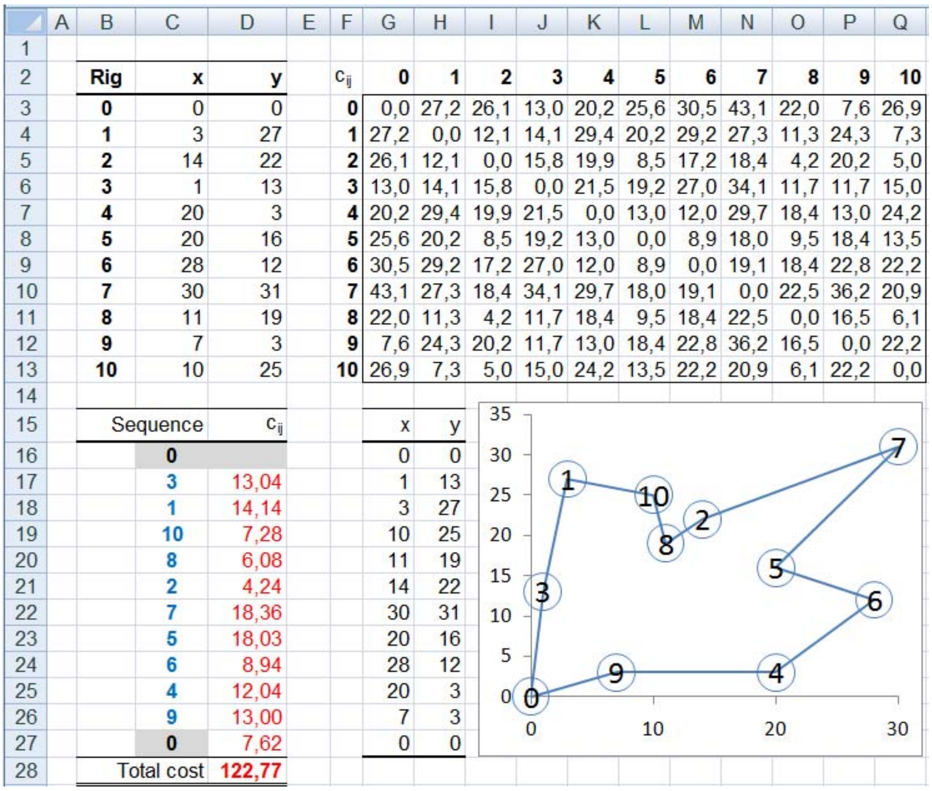
Figure 1 Locations of the oil rigs to visit



9. A direct permutation approach

In this simple form the problem is to find the order for each node in the sequence of the tour that minimises the total distance (cost). If the supply ship takes the tour based on the rig numbers: 0-1-2-...-9-10-0; the total distance is 205.67. We seek the order or permutation that minimises the total distance. This direct approach is very easy to implement in spreadsheets, as displayed in Figure 2.

Figure 2 Spreadsheet for direct permutation of TSP



Rig	x	y
0	0	0
1	3	27
2	14	22
3	1	13
4	20	3
5	20	16
6	28	12
7	30	31
8	11	19
9	7	3
10	10	25

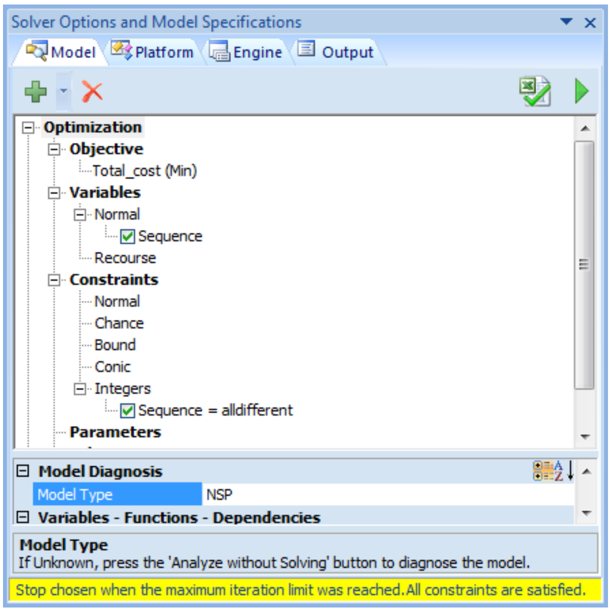
c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0,0	27,2	26,1	13,0	20,2	25,6	30,5	43,1	22,0	7,6	26,9
1	27,2	0,0	12,1	14,1	29,4	20,2	29,2	27,3	11,3	24,3	7,3
2	26,1	12,1	0,0	15,8	19,9	8,5	17,2	18,4	4,2	20,2	5,0
3	13,0	14,1	15,8	0,0	21,5	19,2	27,0	34,1	11,7	11,7	15,0
4	20,2	29,4	19,9	21,5	0,0	13,0	12,0	29,7	18,4	13,0	24,2
5	25,6	20,2	8,5	19,2	13,0	0,0	8,9	18,0	9,5	18,4	13,5
6	30,5	29,2	17,2	27,0	12,0	8,9	0,0	19,1	18,4	22,8	22,2
7	43,1	27,3	18,4	34,1	29,7	18,0	19,1	0,0	22,5	36,2	20,9
8	22,0	11,3	4,2	11,7	18,4	9,5	18,4	22,5	0,0	16,5	6,1
9	7,6	24,3	20,2	11,7	13,0	18,4	22,8	36,2	16,5	0,0	22,2
10	26,9	7,3	5,0	15,0	24,2	13,5	22,2	20,9	6,1	22,2	0,0

Sequence	c_{ij}	x	y
0		0	0
3	13,04	1	13
1	14,14	3	27
10	7,28	10	25
8	6,08	11	19
2	4,24	14	22
7	18,36	30	31
5	18,03	20	16
6	8,94	28	12
4	12,04	20	3
9	13,00	7	3
0	7,62	0	0
Total cost	122,77		

Table 1: Formulas for spreadsheet in Figure 2

Cell	Formula	Copied to	Name / Task
G3	=SQRT((INDEX($C$3:$D$13;$F3+1;1) -INDEX($C$3:$D$13;G$2+1;1))^2 +(INDEX($C$3:$D$13;$F3+1;2) -INDEX($C$3:$D$13;G$2+1;2))^2)	G3:Q13	Calculate Eucledian distances between any pair of nodes
D17	=INDEX($G$3:$Q$13;C16+1;C17+1)	D18:D27	Cost on a leg
D28	=SUM(D17:D27)	-	*Total cost*
C17:C26			*Sequence*
G16	=INDEX($B$3:$D$13;$C16+1;2)	G17:G27	A visited node's x-coordinate
H16	=INDEX($B$3:$D$13;$F16+1;3)	H17:H27	A visited node's y-coordinate

Figure 3 Solver settings for the spreadsheet in Figure 2



Optimization
Objective
Total_cost (Min)
Variables
Normal
☑ Sequence
Recourse
Constraints
Normal
Chance
Bound
Conic
Integers
☑ Sequence = alldifferent
Parameters

Model Diagnosis

Model Type	NSP

Variables - Functions - Dependencies

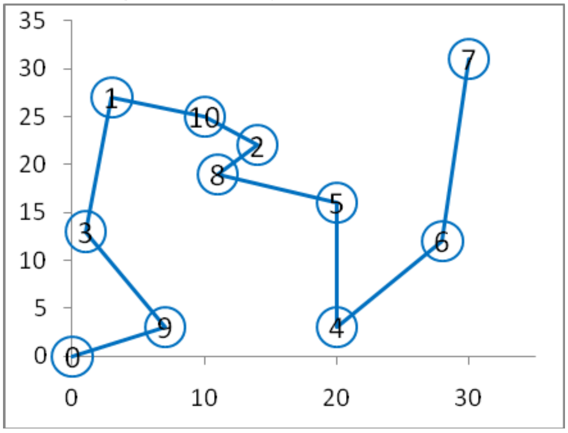
Model Type
If Unknown, press the 'Analyze without Solving' button to diagnose the model.

Stop chosen when the maximum iteration limit was reached. All constraints are satisfied.

The spreadsheet is organised in two parts. The upper part holds a table of the coordinates for the nodes, and a corresponding table calculating the distances. The lower part holds a table of the tour sequence and the cost of each leg, and a corresponding table with the coordinates of each leg, to facilitate a plot of the tour. The table of the tour sequence starts at the base. Note that node number 0 is used for the depot in this example, to facilitate use of the *Alldifferent* constraint in Solver. (A trial version of Solver is available at www.solver.com.) The problem is to select which node to go to next in the sequence (heading 'Sequence' in Figure 2). The last leg has to return to the base. The minimum total distance/cost of 122.77 is achieved by the tour sequence 0-9-4-6-5-7-2-8-10-1-3-0 (or reverse). To model an open tour simply delete row 27 in the sheet. Figure 4 displays the optimal open tour, which has a cost of 103.58.

The scatter plot consists of two series. One series is a plot of the nodes (C3:D13 in Figure 2), with markers but no line. The second series is the tour (G13:H27 in Figure 2), with no markers and a line.

Figure 4 The open TSP solution



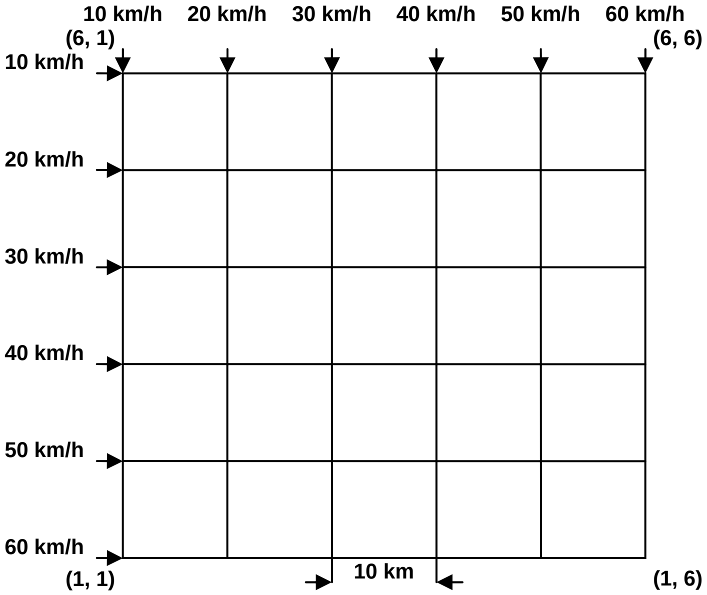
Both the spreadsheet and the Solver settings are very simple. We have 10 decision variables (number of nodes less the depot), named '*Sequence*' in the spreadsheet. The objective in the Solver settings is to minimise the value in the cell named '*Total_cost*', and the only constraint is that the variables must be all different. The **alldifferent** constraint sets the variables to integers ranging from 1 to the number of variables, and all are different. (This type of constraint is not available in the Standard Solver that ships with Excel prior to Excel 2010, but is introduced in the educational version of Solver, included in many textbooks.)

The use of the **Index** function in Excel to look up the cost at each leg makes the objective function non-smooth, because the decision variables are used as arguments in the Index function. An integer non-smooth problem is not easy to solve, and is definitely not the preferred form for large problems. In this case the *Premium Solver Platform* (PSP) selects the *OptQuest* solver engine, and Solver spends less than two seconds in finding the optimal tour (the *AutoStop* option for *OptQuest* was increased from 100 to 1000 iterations to avoid a premature ending). As this solver engine applies heuristics, it cannot guarantee that a global optimal solution has been found. When such problems become large, this non-linear approach is no longer efficient. We will therefore introduce the linear formulation, which will be applied in the rest of the paper. Also note that the direct permutation approach does not allow for multiple visits.

10. TSP in a non-complete graph, flow formulation

As an introductory example for a non-complete graph the 'Gridspeed' puzzle will be used, taken from Chlond (2008). Figure 5 presents the puzzle, based on a rectangular grid street plan, where the distance between any two intersections is 10 kilometres. (I have taken the liberty to transform the data to the metric system.) The speed along all north-south streets and all east-west avenues is constant. However the speed on the north-south streets is highest on the east end of the grid, and for the avenues east-west the speed is highest in the south end of the grid. The fastest area is therefore at the south-east edges of the grid, and slowest in north-west.

Figure 5 Street plan



One puzzle related to Figure 5 is to find the fastest route from intersection (6,1) (north-west) to intersection (1,1) (south-west), but visiting each intersection at least once. The original problem is to visit each intersection once and only once. However this is more restricted than required. Since it obviously will take more time to visit an intersection more than once, and we want to spend as short time as possible on the tour, it is sufficient to use the requirement to visit each intersection at least once.

It is necessary to transform the problem by numbering the intersections and calculate the travelling time between each (directly connected node), to facilitate a mathematical formulation. The numbered intersections are the nodes, and the lines connecting the nodes are the arcs. The travelling time (in minutes) along each arc is calculated as shown in Figure 6.

Figure 6 Relabelled street plan

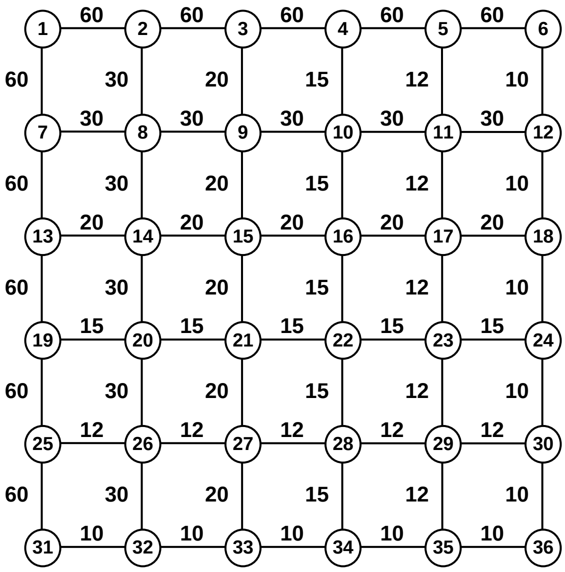
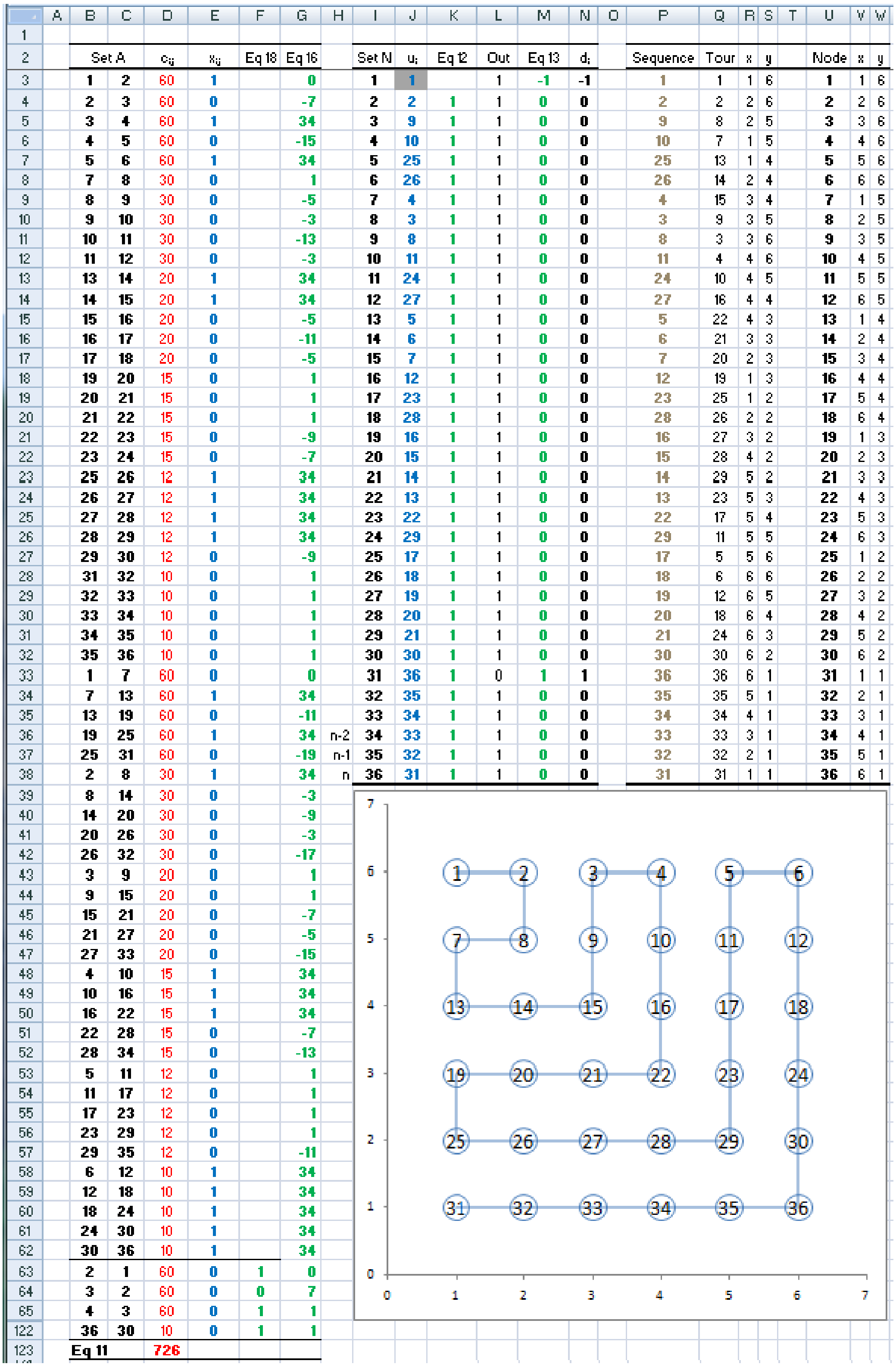


Figure 7 Spreadsheet of open TSP, non-complete graph (rows 66-121 are hidden)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1																							
2		Set A		c_{ij}	x_{ij}	Eq 18	Eq 16		Set N	u_i	Eq 12	Out	Eq 13	d_i		Sequence	Tour	x	y		Node	x	y
3		1	2	60	1		0		1	1		1	-1	-1		1	1	1	6		1	1	6
4		2	3	60	0		-7		2	2	1	1	0	0		2	2	2	6		2	2	6
5		3	4	60	1		34		3	9	1	1	0	0		9	8	2	5		3	3	6
6		4	5	60	0		-15		4	10	1	1	0	0		10	7	1	5		4	4	6
7		5	6	60	1		34		5	25	1	1	0	0		25	13	1	4		5	5	6
8		7	8	30	0		1		6	26	1	1	0	0		26	14	2	4		6	6	6
9		8	9	30	0		-5		7	4	1	1	0	0		4	15	3	4		7	1	5
10		9	10	30	0		-3		8	3	1	1	0	0		3	9	3	5		8	2	5
11		10	11	30	0		-13		9	8	1	1	0	0		8	3	3	6		9	3	5
12		11	12	30	0		-3		10	11	1	1	0	0		11	4	4	6		10	4	5
13		13	14	20	1		34		11	24	1	1	0	0		24	10	4	5		11	5	5
14		14	15	20	1		34		12	27	1	1	0	0		27	16	4	4		12	6	5
15		15	16	20	0		-5		13	5	1	1	0	0		5	22	4	3		13	1	4
16		16	17	20	0		-11		14	6	1	1	0	0		6	21	3	3		14	2	4
17		17	18	20	0		-5		15	7	1	1	0	0		7	20	2	3		15	3	4
18		19	20	15	0		1		16	12	1	1	0	0		12	19	1	3		16	4	4
19		20	21	15	0		1		17	23	1	1	0	0		23	25	1	2		17	5	4
20		21	22	15	0		1		18	28	1	1	0	0		28	26	2	2		18	6	4
21		22	23	15	0		-9		19	16	1	1	0	0		16	27	3	2		19	1	3
22		23	24	15	0		-7		20	15	1	1	0	0		15	28	4	2		20	2	3
23		25	26	12	1		34		21	14	1	1	0	0		14	29	5	2		21	3	3
24		26	27	12	1		34		22	13	1	1	0	0		13	23	5	3		22	4	3
25		27	28	12	1		34		23	22	1	1	0	0		22	17	5	4		23	5	3
26		28	29	12	1		34		24	29	1	1	0	0		29	11	5	5		24	6	3
27		29	30	12	0		-9		25	17	1	1	0	0		17	5	5	6		25	1	2
28		31	32	10	0		1		26	18	1	1	0	0		18	6	6	6		26	2	2
29		32	33	10	0		1		27	19	1	1	0	0		19	12	6	5		27	3	2
30		33	34	10	0		1		28	20	1	1	0	0		20	18	6	4		28	4	2
31		34	35	10	0		1		29	21	1	1	0	0		21	24	6	3		29	5	2
32		35	36	10	0		1		30	30	1	1	0	0		30	30	6	2		30	6	2
33		1	7	60	0		0		31	36	1	0	1	1		36	36	6	1		31	1	1
34		7	13	60	1		34		32	35	1	1	0	0		35	35	5	1		32	2	1
35		13	19	60	0		-11		33	34	1	1	0	0		34	34	4	1		33	3	1
36		19	25	60	1		34	n-2	34	33	1	1	0	0		33	33	3	1		34	4	1
37		25	31	60	0		-19	n-1	35	32	1	1	0	0		32	32	2	1		35	5	1
38		2	8	30	1		34	n	36	31	1	1	0	0		31	31	1	1		36	6	1
39		8	14	30	0		-3																
40		14	20	30	0		-9																
41		20	26	30	0		-3																
42		26	32	30	0		-17																
43		3	9	20	0		1																
44		9	15	20	0		1																
45		15	21	20	0		-7																
46		21	27	20	0		-5																
47		27	33	20	0		-15																
48		4	10	15	1		34																
49		10	16	15	1		34																
50		16	22	15	1		34																
51		22	28	15	0		-7																
52		28	34	15	0		-13																
53		5	11	12	0		1																
54		11	17	12	0		1																
55		17	23	12	0		1																
56		23	29	12	0		1																
57		29	35	12	0		-11																
58		6	12	10	1		34																
59		12	18	10	1		34																
60		18	24	10	1		34																
61		24	30	10	1		34																
62		30	36	10	1		34																
63		2	1	60	0	1	0																
64		3	2	60	0	0	7																
65		4	3	60	0	1	1																
122		36	30	10	0	1	1																
123		Eq 11		726																			

Table 2 Formulas for the spreadsheet in Figure 7

Cell	Formula	Copied to	Name / Task
B63	=C3	B64:B122	Reverse arcs: stop→start
C63	=B3	C64:C122	Reverse arcs: start→stop
D63	=D3	D64:D122	Reverse arcs: copy costs
D123	=SUMPRODUCT(D3:D122;E3:E122)	-	*Eq 11*
F63	=E3+E63	F64:F122	*Eq 18* (LHS)
G3	=IF(OR(B3=1;C3=1);0;INDEX($J$3:$J$38;B3) -INDEX($J$3:$J$38;C3) +($I$37*E3))	G4:G122	*Eq 16* (LHS)
K4	=SUMIF($C$3:$C$122;$I$3:$I$38;$E$3:$E$122)	K5:K38	*Eq 12* (LHS)
L3	=SUMIF($B$3:$B$122;$I$3:$I$38;$E$3:$E$122)	L4:L38	First part of Eq 13
M3	=K3-L3	M4:M38	*Eq 13* (LHS)
E3:E122			*Var_x*
I36			*Param_n_2*
I38			*Param_n*
J4:J38			*Var_u*
N3:N38			*Param_d*
P3	=RANK(J3;$J$3:$J$38;1)	P4:P38	The rank of a node
Q3	=MATCH(U3;$P$3:$P$38;0)	Q4:Q38	Visiting sequence
R3	=INDEX($U$3:$W$38;$Q3;2)	R4:R38	A visited node's x-coordinate
S3	=INDEX($U$3:$W$38;$Q3;3)	S4:S38	A visited node's y-coordinate

11. The open tour

We will first have a look at the open tour variant of the puzzle. An efficient layout of this network in a spreadsheet would be to organise the problem in two tables, one table for the arcs and the binary decision variables, and another table for the nodes and the continuous variables (as in Ragsdale, 2001). This will facilitate the entry of the equations (11) – (14), (16) and (17), and also make a solution easy to understand. Once the data has been entered in the spreadsheet, the model can easily be built around the data. Notice that for non-directed arcs it is sufficient to enter them in one direction, and use simple formulas to mirror the other direction. A third table has been added to the spreadsheet to facilitate a plot of the tour, which of course is not needed for solving the problem, but handy for displaying the solution.

A new constraint has been added, to speed up the solution process:

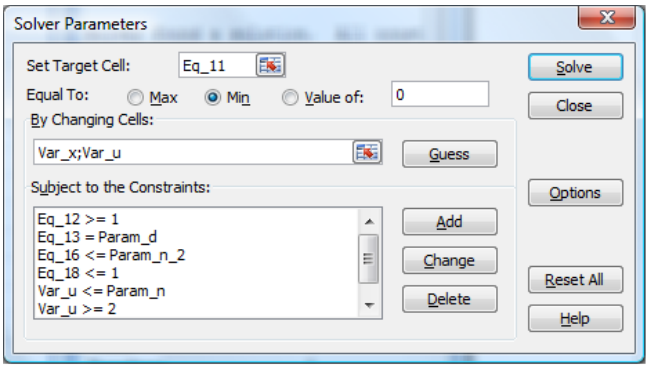
$$x_{i,j} + x_{j,i} \le 1 \ \ \forall \ (i,j) \in A \qquad (18)$$

Constraint (18) simply states that no arc will be used in both directions, which is quite obvious for this problem. Such bounds on the variables are very helpful for the optimisation process, particularly so for binary variables. However they should be used with care, since adding them can make some problems infeasible.

In Figure 7 the table for the arcs is listed first; then the table for the nodes and finally the table for facilitating a plot of the tour. The first half of the arcs are listed (all arcs in one direction) and a few of the rest, together with the objective. (The rest of the arcs are in the hidden rows 66–121.) We see that the optimal value of the objective (11) is 726; the fastest open tour from node 1 to node 31 takes a minimum of 726 minutes. The formulas in the spreadsheet are displayed in Table 2, and the Solver settings are listed in Figure 8.

The first three rows in Table 2 are purely for easy data entry. The optimisation model consists of the next six rows. The following five rows are used for naming some key cells, making the model easier to read. The last four rows facilitate a plot of the solution, assuming only *n* legs in the tour (each node is visited only once). The scatter plot in Figure 7 consists of two data series. One series is the *xy* coordinates of the nodes (in column V and W), with no line, and a circle (size 20) as marker.[2] The second series is the *xy* coordinates of the tour (in column R and S), with no marker and a line. For a closed tour a final leg is added at the end (by referring to the first leg in column Q).

Figure 8 Solver settings for Figure 7


Solver Parameters

Set Target Cell: Eq_11

Equal To: ○ Max ● Min ○ Value of: 0

By Changing Cells:

Var_x;Var_u

Subject to the Constraints:

Eq_12 >= 1
Eq_13 = Param_d
Eq_16 <= Param_n_2
Eq_18 <= 1
Var_u <= Param_n
Var_u >= 2

The Standard Solver Parameter Dialog Box displayed in Figure 8 has a scroll bar to display the constraints not fitting the fixed size of the box. Here constraint (14) is not displayed; this is the declaration of the x_{ij} variable being binary. Observe that constraint (14) and (17) is entered directly in Solver, involving no formulas in the spreadsheet. Constraints (17) are the last two visible constraints in the Solver Parameter Dialog Box.

This model has 120 binary variables and 35 continuous variables, 155 bounds on the variables and 191 constraints. The number of constraints can be reduced by 4 if we group the four arcs connected to node 1 and list them first, then not include them in constraints (16). In the spreadsheet the formula for (16) include these arcs, but fix their value to 0, thereby satisfying the constraint.

Excel and the Standard Solver take less than five seconds to find the optimal solution for the open tour. (The solution time will of course depend on the version of Excel, the operating system, and the computer.) This spreadsheet design is quite versatile for many types of network problems. If we drop (12), (16), (17) and (18) we have the shortest path problem. (We may then delete column F, G, J and P – W.)

12. A closed TSP in a non-complete graph, assignment formulation

We will now rephrase the problem to a closed tour, requiring the salesman to return to the base. We will implement the assignment formulation and compare it with a flow formulation (not shown). We will also demonstrate an efficient layout for the spreadsheet of a TSP in a complete graph, even though this particular example is non-complete. For a TSP in a complete graph, it is more efficient to group the problem in three tables; one table for the cost matrix and the objective (1), a second table for the x_{ij}

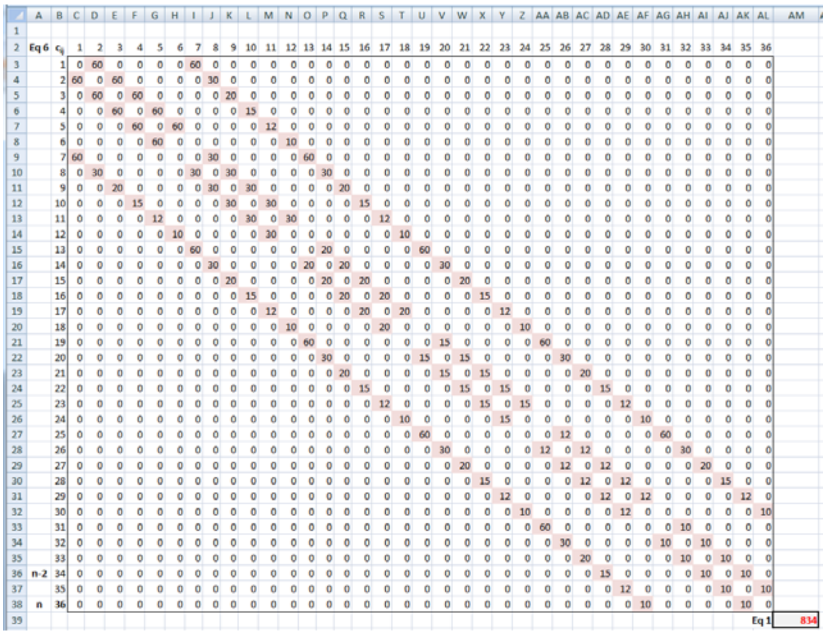
[2] The labels of the scatter plot were made by the XY Chart Labeler add-in for Excel, free at www.appspro.com.

binary decision variables and constraint (2) and (3), and a third table for the SECs and the related u_i variables. For convenience a table of the coordinates of the nodes can be added to facilitate a plot of the tour. For large problems these matrixes may be entered in different sheets in the workbook. (It is more effective though for Solver to have the objective, constraints and variables in one sheet. The cost matrix and the plot data can be stored in a separate sheet.)

A matrix layout of the costs is very efficient for a complete graph, when there are direct links between any node to every other node. However, it is also very common to use the same approach for non-complete graphs, maybe because it is considered handier for data entry, at least in the preferred software tools most commonly used.

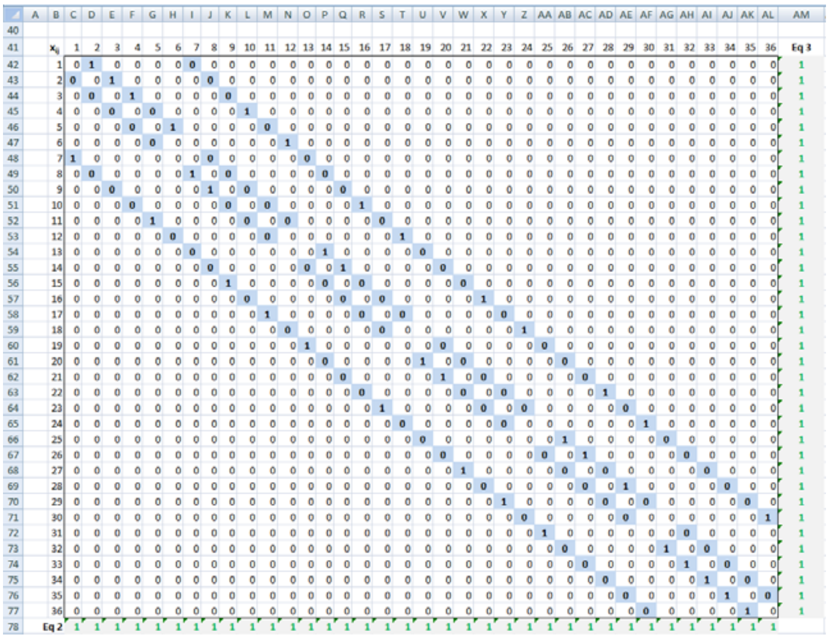
Unfortunately this convenience has a trade off. Adding enormous amount of non-existing variables and correcting this by adding an equal amount of non-existing constraints, makes a substantial burden on the software. In contrast, entering the data in a spreadsheet may actually be easier for non-complete graph situations, using only two tables instead of three.

Figure 9 Matrix of costs for TSP non-complete graph, assignment formulation



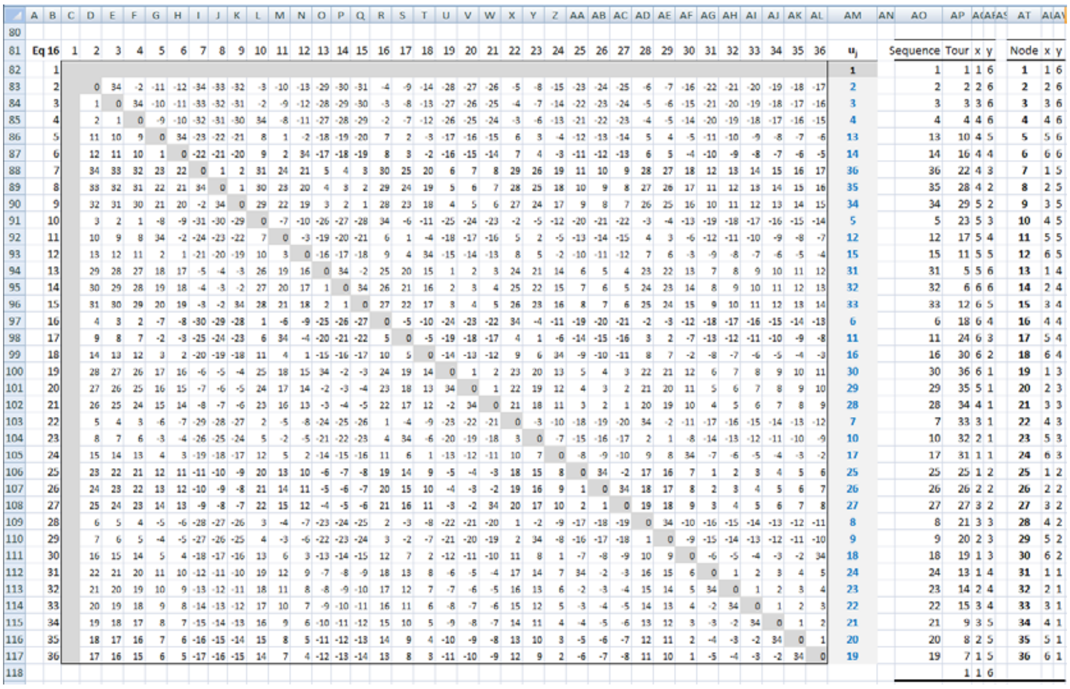
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1																																							
2	Eq 6	c_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	
3		1	0	60	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4		2	60	0	60	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5		3	0	60	0	60	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6		4	0	0	60	0	60	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7		5	0	0	0	60	0	60	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8		6	0	0	0	0	60	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9		7	60	0	0	0	0	0	0	30	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10		8	0	30	0	0	0	0	30	0	30	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11		9	0	0	20	0	0	0	0	30	0	30	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12		10	0	0	0	15	0	0	0	0	30	0	30	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
13		11	0	0	0	0	12	0	0	0	0	30	0	30	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14		12	0	0	0	0	0	10	0	0	0	0	30	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15		13	0	0	0	0	0	0	60	0	0	0	0	0	0	20	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16		14	0	0	0	0	0	0	0	30	0	0	0	0	20	0	20	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17		15	0	0	0	0	0	0	0	0	20	0	0	0	0	20	0	20	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18		16	0	0	0	0	0	0	0	0	0	15	0	0	0	0	20	0	20	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19		17	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	20	0	20	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	
20		18	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	20	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	
21		19	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	15	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	
22		20	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	15	0	15	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	
23		21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	15	0	15	0	0	0	0	20	0	0	0	0	0	0	0	0	0	
24		22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	15	0	15	0	0	0	0	15	0	0	0	0	0	0	0	0	
25		23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	15	0	15	0	0	0	0	12	0	0	0	0	0	0	0	
26		24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	15	0	0	0	0	0	0	10	0	0	0	0	0	0	
27		25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	12	0	0	0	0	60	0	0	0	0	0	
28		26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	12	0	12	0	0	0	0	30	0	0	0	0	
29		27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	12	0	12	0	0	0	0	20	0	0	0	
30		28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	12	0	12	0	0	0	0	15	0	0	
31		29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	12	0	12	0	0	0	0	12	0	
32		30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	12	0	0	0	0	0	0	10	
33		31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0	10	0	0	0	0	
34		32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	10	0	10	0	0	0	
35		33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	10	0	10	0	0	
36	n-2	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	10	0	10	0	
37		35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	10	0	10	
38	n	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	10	0	
39																																						Eq 1	834

Figure 10 Matrix of binary variables for TSP non-complete graph, assignment formulation



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
40																																							
41		x_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	Eq 3
42		1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
43		2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
44		3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
45		4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
46		5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
47		6	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
48		7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
49		8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
50		9	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
51		10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
52		11	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
53		12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
54		13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
55		14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
56		15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
57		16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
58		17	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
59		18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
60		19	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
61		20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
62		21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
63		22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
64		23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
65		24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
66		25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
67		26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
68		27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
69		28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
70		29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
71		30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
72		31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
73		32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
74		33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
75		34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
76		35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
77		36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
78		Eq 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

Figure 11 Matrix of SECs and plot data for TSP non-complete graph, assignment formulation



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	A(	AF	AS	AT	AI	AV
80																																																
81	Eq 16		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	u_j		Sequence	Tour	x	y		Node	x	y
82		1																																					1		1	1	1	6		1	1	6
83		2		0	34	-2	-11	-12	-34	-33	-32	-3	-10	-13	-29	-30	-31	-4	-9	-14	-28	-27	-26	-5	-8	-15	-23	-24	-25	-6	-7	-16	-22	-21	-20	-19	-18	-17	2		2	2	2	6		2	2	6
84		3		1	0	34	-10	-11	-33	-32	-31	-2	-9	-12	-28	-29	-30	-3	-8	-13	-27	-26	-25	-4	-7	-14	-22	-23	-24	-5	-6	-15	-21	-20	-19	-18	-17	-16	3		3	3	3	6		3	3	6
85		4		2	1	0	-9	-10	-32	-31	-30	34	-8	-11	-27	-28	-29	-2	-7	-12	-26	-25	-24	-3	-6	-13	-21	-22	-23	-4	-5	-14	-20	-19	-18	-17	-16	-15	4		4	4	4	6		4	4	6
86		5		11	10	9	0	34	-23	-22	-21	8	1	-2	-18	-19	-20	7	2	-3	-17	-16	-15	6	3	-4	-12	-13	-14	5	4	-5	-11	-10	-9	-8	-7	-6	13		13	10	4	5		5	5	6
87		6		12	11	10	1	0	-22	-21	-20	9	2	34	-17	-18	-19	8	3	-2	-16	-15	-14	7	4	-3	-11	-12	-13	6	5	-4	-10	-9	-8	-7	-6	-5	14		14	16	4	4		6	6	6
88		7		34	33	32	23	22	0	1	2	31	24	21	5	4	3	30	25	20	6	7	8	29	26	19	11	10	9	28	27	18	12	13	14	15	16	17	36		36	22	4	3		7	1	5
89		8		33	32	31	22	21	34	0	1	30	23	20	4	3	2	29	24	19	5	6	7	28	25	18	10	9	8	27	26	17	11	12	13	14	15	16	35		35	28	4	2		8	2	5
90		9		32	31	30	21	20	-2	34	0	29	22	19	3	2	1	28	23	18	4	5	6	27	24	17	9	8	7	26	25	16	10	11	12	13	14	15	34		34	29	5	2		9	3	5
91		10		3	2	1	-8	-9	-31	-30	-29	0	-7	-10	-26	-27	-28	34	-6	-11	-25	-24	-23	-2	-5	-12	-20	-21	-22	-3	-4	-13	-19	-18	-17	-16	-15	-14	5		5	23	5	3		10	4	5
92		11		10	9	8	34	-2	-24	-23	-22	7	0	-3	-19	-20	-21	6	1	-4	-18	-17	-16	5	2	-5	-13	-14	-15	4	3	-6	-12	-11	-10	-9	-8	-7	12		12	17	5	4		11	5	5
93		12		13	12	11	2	1	-21	-20	-19	10	3	0	-16	-17	-18	9	4	34	-15	-14	-13	8	5	-2	-10	-11	-12	7	6	-3	-9	-8	-7	-6	-5	-4	15		15	11	5	5		12	6	5
94		13		29	28	27	18	17	-5	-4	-3	26	19	16	0	34	-2	25	20	15	1	2	3	24	21	14	6	5	4	23	22	13	7	8	9	10	11	12	31		31	5	5	6		13	1	4
95		14		30	29	28	19	18	-4	-3	-2	27	20	17	1	0	34	26	21	16	2	3	4	25	22	15	7	6	5	24	23	14	8	9	10	11	12	13	32		32	6	6	6		14	2	4
96		15		31	30	29	20	19	-3	-2	34	28	21	18	2	1	0	27	22	17	3	4	5	26	23	16	8	7	6	25	24	15	9	10	11	12	13	14	33		33	12	6	5		15	3	4
97		16		4	3	2	-7	-8	-30	-29	-28	1	-6	-9	-25	-26	-27	0	-5	-10	-24	-23	-22	34	-4	-11	-19	-20	-21	-2	-3	-12	-18	-17	-16	-15	-14	-13	6		6	18	6	4		16	4	4
98		17		9	8	7	-2	-3	-25	-24	-23	6	34	-4	-20	-21	-22	5	0	-5	-19	-18	-17	4	1	-6	-14	-15	-16	3	2	-7	-13	-12	-11	-10	-9	-8	11		11	24	6	3		17	5	4
99		18		14	13	12	3	2	-20	-19	-18	11	4	1	-15	-16	-17	10	5	0	-14	-13	-12	9	6	34	-9	-10	-11	8	7	-2	-8	-7	-6	-5	-4	-3	16		16	30	6	2		18	6	4
100		19		28	27	26	17	16	-6	-5	-4	25	18	15	34	-2	-3	24	19	14	0	1	2	23	20	13	5	4	3	22	21	12	6	7	8	9	10	11	30		30	36	6	1		19	1	3
101		20		27	26	25	16	15	-7	-6	-5	24	17	14	-2	-3	-4	23	18	13	34	0	1	22	19	12	4	3	2	21	20	11	5	6	7	8	9	10	29		29	35	5	1		20	2	3
102		21		26	25	24	15	14	-8	-7	-6	23	16	13	-3	-4	-5	22	17	12	-2	34	0	21	18	11	3	2	1	20	19	10	4	5	6	7	8	9	28		28	34	4	1		21	3	3
103		22		5	4	3	-6	-7	-29	-28	-27	2	-5	-8	-24	-25	-26	1	-4	-9	-23	-22	-21	0	-3	-10	-18	-19	-20	34	-2	-11	-17	-16	-15	-14	-13	-12	7		7	33	3	1		22	4	3
104		23		8	7	6	-3	-4	-26	-25	-24	5	-2	-5	-21	-22	-23	4	34	-6	-20	-19	-18	3	0	-7	-15	-16	-17	2	1	-8	-14	-13	-12	-11	-10	-9	10		10	32	2	1		23	5	3
105		24		15	14	13	4	3	-19	-18	-17	12	5	2	-14	-15	-16	11	6	1	-13	-12	-11	10	7	0	-8	-9	-10	9	8	34	-7	-6	-5	-4	-3	-2	17		17	31	1	1		24	6	3
106		25		23	22	21	12	11	-11	-10	-9	20	13	10	-6	-7	-8	19	14	9	-5	-4	-3	18	15	8	0	34	-2	17	16	7	1	2	3	4	5	6	25		25	25	1	2		25	1	2
107		26		24	23	22	13	12	-10	-9	-8	21	14	11	-5	-6	-7	20	15	10	-4	-3	-2	19	16	9	1	0	34	18	17	8	2	3	4	5	6	7	26		26	26	2	2		26	2	2
108		27		25	24	23	14	13	-9	-8	-7	22	15	12	-4	-5	-6	21	16	11	-3	-2	34	20	17	10	2	1	0	19	18	9	3	4	5	6	7	8	27		27	27	3	2		27	3	2
109		28		6	5	4	-5	-6	-28	-27	-26	3	-4	-7	-23	-24	-25	2	-3	-8	-22	-21	-20	1	-2	-9	-17	-18	-19	0	34	-10	-16	-15	-14	-13	-12	-11	8		8	21	3	3		28	4	2
110		29		7	6	5	-4	-5	-27	-26	-25	4	-3	-6	-22	-23	-24	3	-2	-7	-21	-20	-19	2	34	-8	-16	-17	-18	1	0	-9	-15	-14	-13	-12	-11	-10	9		9	20	2	3		29	5	2
111		30		16	15	14	5	4	-18	-17	-16	13	6	3	-13	-14	-15	12	7	2	-12	-11	-10	11	8	1	-7	-8	-9	10	9	0	-6	-5	-4	-3	-2	34	18		18	19	1	3		30	6	2
112		31		22	21	20	11	10	-12	-11	-10	19	12	9	-7	-8	-9	18	13	8	-6	-5	-4	17	14	7	34	-2	-3	16	15	6	0	1	2	3	4	5	24		24	13	1	4		31	1	1
113		32		21	20	19	10	9	-13	-12	-11	18	11	8	-8	-9	-10	17	12	7	-7	-6	-5	16	13	6	-2	-3	-4	15	14	5	34	0	1	2	3	4	23		23	14	2	4		32	2	1
114		33		20	19	18	9	8	-14	-13	-12	17	10	7	-9	-10	-11	16	11	6	-8	-7	-6	15	12	5	-3	-4	-5	14	13	4	-2	34	0	1	2	3	22		22	15	3	4		33	3	1
115		34		19	18	17	8	7	-15	-14	-13	16	9	6	-10	-11	-12	15	10	5	-9	-8	-7	14	11	4	-4	-5	-6	13	12	3	-3	-2	34	0	1	2	21		21	9	3	5		34	4	1
116		35		18	17	16	7	6	-16	-15	-14	15	8	5	-11	-12	-13	14	9	4	-10	-9	-8	13	10	3	-5	-6	-7	12	11	2	-4	-3	-2	34	0	1	20		20	8	2	5		35	5	1
117		36		17	16	15	6	5	-17	-16	-15	14	7	4	-12	-13	-14	13	8	3	-11	-10	-9	12	9	2	-6	-7	-8	11	10	1	-5	-4	-3	-2	34	0	19		19	7	1	5		36	6	1
118																																										1	1	6				

Figure 12 The plot of the closed TSP in a non-complete graph, assignment formulation

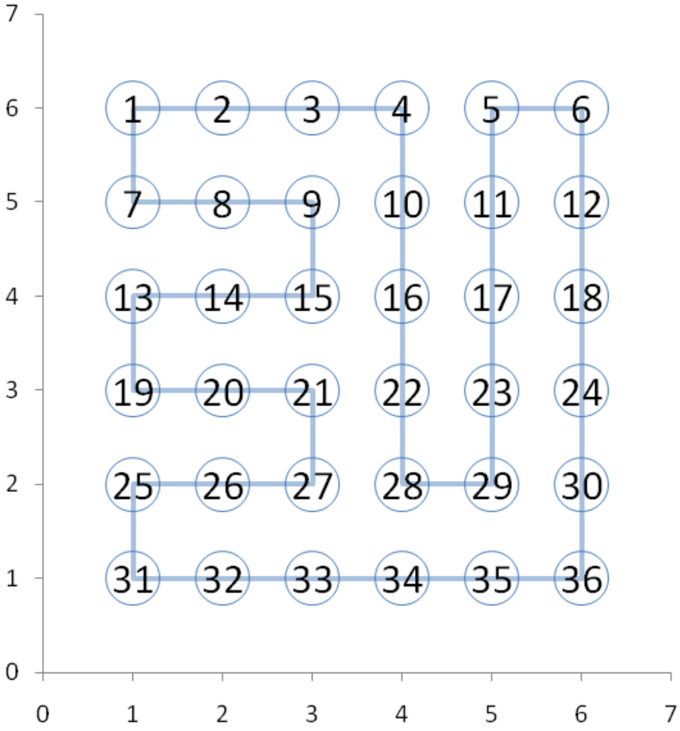
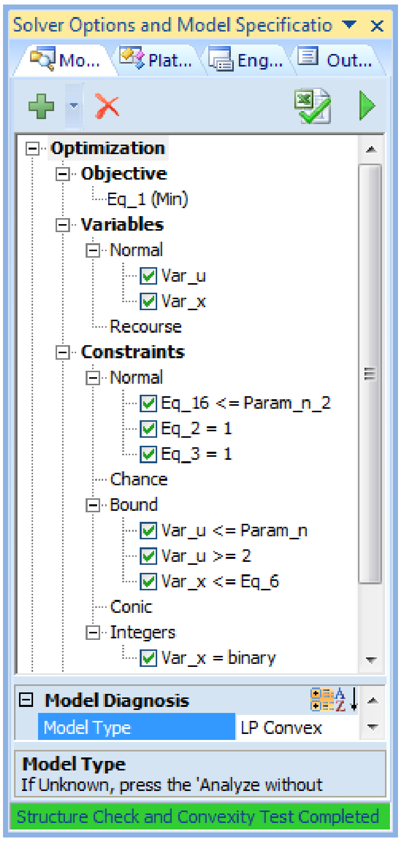


Figure 13 Solver settings for Figures 9 to 11


Solver Options and Model Specificatio

- Optimization
  - Objective
    - Eq_1 (Min)
  - Variables
    - Normal
      - ☑ Var_u
      - ☑ Var_x
    - Recourse
  - Constraints
    - Normal
      - ☑ Eq_16 <= Param_n_2
      - ☑ Eq_2 = 1
      - ☑ Eq_3 = 1
    - Chance
    - Bound
      - ☑ Var_u <= Param_n
      - ☑ Var_u >= 2
      - ☑ Var_x <= Eq_6
    - Conic
    - Integers
      - ☑ Var_x = binary

Model Diagnosis

Model Type: LP Convex

Model Type
If Unknown, press the 'Analyze without

Structure Check and Convexity Test Completed

Table 3: Formulas for spreadsheet in Figures 9 to 11

Cell	Formula	Copied to	Name / Task
AM39	=SUMPRODUCT(C3:AL38;C42:AL77)	-	*Eq 1*
AM42	=SUM(C42:AL42)	AM43:AM77	*Eq 3* (LHS)
C78	=SUM(C42:C77)	D78:AL78	*Eq 2* (LHS)
D83	=INDEX($AM$82:$AM$117;$B83) -INDEX($AM$82:$AM$117;D$81) +INDEX($C$42:$AL$77;$B83;D$81)*$B$37	D83:AL117	*Eq 15* (LHS)
C3:AL38			*Eq 6* (RHS)
C42:AL77			*Var_x*
B36			*Param_n_2*
B38			*Param_n*
AM83:AM117			*Var_u*
AO82	=RANK(AM82;$AM$82:$AM$117;1)	AO83:AO117	The rank of a node
AP82	=MATCH(AT82;$AO$82:$AO$117;0)	AP83:AP117	Visiting sequence
AP118	=AP82	-	Last leg, return to base
AQ82	=INDEX($AT$82:$AV$117;$AP82;2)	AQ83:AQ118	A visited node's x-coordinate
AR82	=INDEX($AT$82:$AV$117;$AP82;3)	AR83:AR118	A visited node's y-coordinate

Due to its size this formulation cannot be solved by the Standard Solver in Excel, which has a limit of 200 variables. We see from Figure 9 that the minimum time to complete a closed tour is 834, the value for the objective (1).

In Table 4 the model in Figure 7, a closed network version (not shown), and the closed assignment version in Figures 9 to 11 are compared. The solution time for the assignment formulation of the closed tour is more than 6.2 times the solution time for the flow formulation (using the Gurobi Solver Engine in RSP V9.04 for Excel).

Table 4: Key features of the models

Models of TSP	Figure 7	Not shown	Figure 9-11
Type, model	Open, network	Closed, network	Closed, assignment
Integer variables	155	155	1296
Continuous variables	35	35	35
Constraints	248	249	1297
Bounds on variables	70	70	1331
Solution time (seconds)	0.44	3.84	24.01
Time Standard Solver	≈5	≈Over night	Not solvable

Lessons learned from this small example are that formulation matters. Avoid using non-existing variables rectified by non-existing constraints. However, adding constraints may have a great impact, even when not changing the optimal solution. Constraints that are tightening the feasible space may speed up the solution time (or the contrary), but such constraints must not eliminate an otherwise optimal solution.

13. A closed MBTSP in a non-complete graph, flow formulation

The ultimate minimum cost for any problem is zero; simply do nothing. So there must be a reason for doing something (presumably there is a null alternative). Minimising costs can often turn out to be solving the wrong problem. Unless any revenues are completely unaffected by the decisions at hand we cannot be sure that minimising costs is a valid model that accurately represents the relevant characteristics of the problem.

Let us assume there are some revenues or benefits b_j by visiting node *j*; where $j \in N$. Also introduce the set $Y = \{y_j : j \in N, j \neq 1, y_j \in \{0,1\}\}$ where the binary decision variable $y_j = 1$ if the salesman visits node *j*, else *0*. Define the parameter $y_1 = 1$ as the requirement to visit/return to the depot (node 1), a consequence of our SECs.

The objective now is to maximise the total net benefit (total benefits minus total costs):

$$\text{maximize} \sum_{j \in N} b_j \cdot y_j - \sum_{(i,j) \in A} c_{i,j} \cdot x_{i,j} \qquad (19)$$

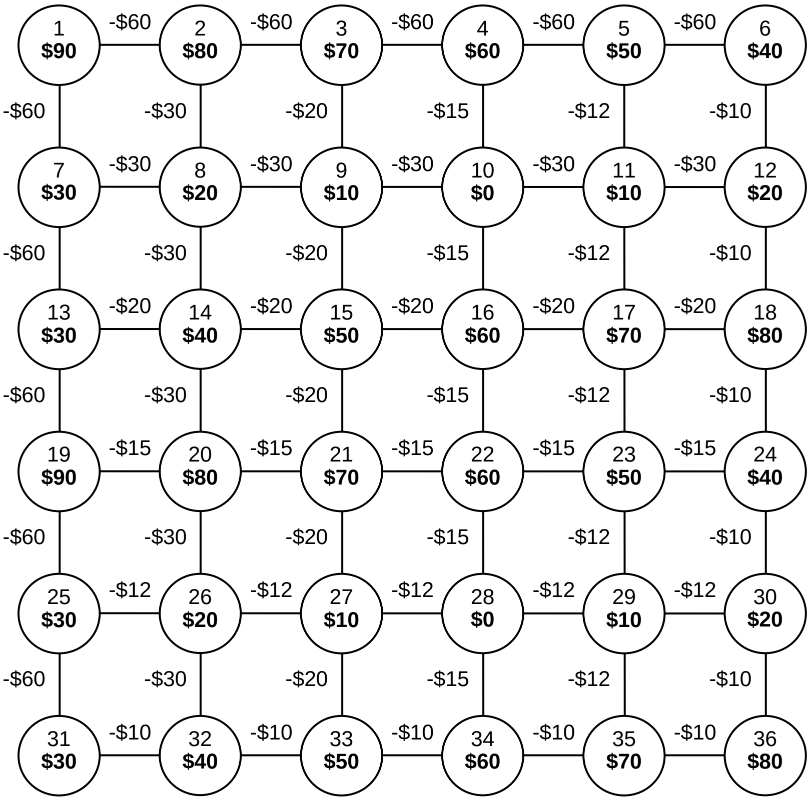
The salesman has to arrive (at least) once each node he decides to visit:

$$\sum_{i:(i,j) \in A} x_{i,j} \geq y_j \quad \forall \quad j \in N \qquad (20)$$

In addition the balance constraint (9) and binary conditions (10) apply, as well as the SECs (16) and the corresponding bounds (17). Here (19) and (20) replace the original objective (7) and constraint (8), when compared to the closed TSP flow formulation.

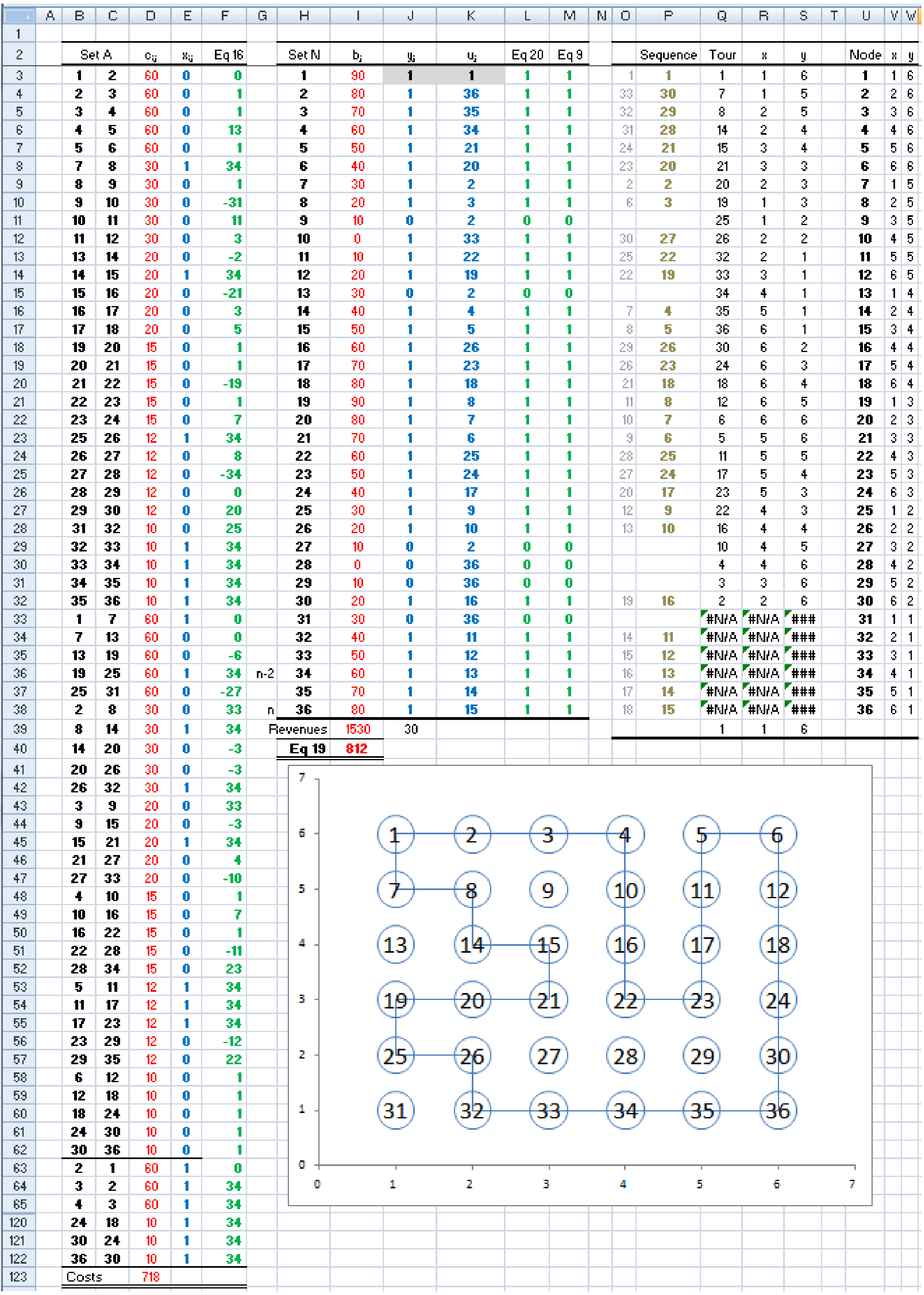
The revised problem is presented in Figure 14. The revenues are entered with $ symbols under each node number, indicating the revenue if the node is being visited. The cost is assumed to be 1$ for each minute of travel time, so we have a common unit of measurement in the objective. Observe that node 10 and node 28 have no revenues, and are therefore obvious candidates for no visits.

Figure 14 Relabelled street plan for MBTSP



This is a variation of the Price Collecting TSP. The common feature of these problems in this category is the combination of two kinds of decisions, the selection of some nodes, and the ordering of the nodes selected for the tour (see Gutin, 2007).

Figure 15 Closed MBTSP flow formulation in a spreadsheet, rows 66–119 are hidden

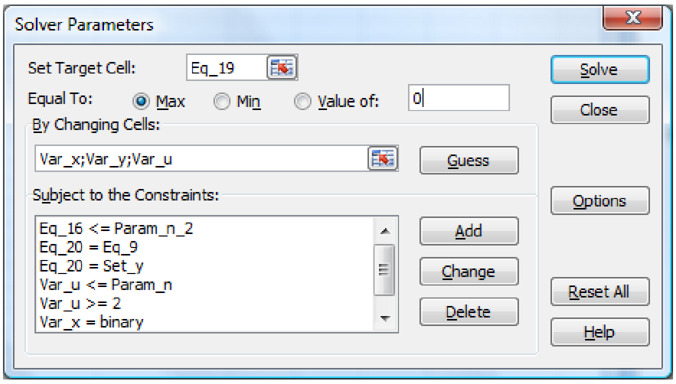


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1																							
2		Set A		c_{ij}	x_{ij}	Eq 16		Set N	b_j	y_j	u_j	Eq 20	Eq 9			Sequence	Tour	x	y		Node	x	y
3		1	2	60	0	0		1	90	1	1	1	1		1	1	1	1	6		1	1	6
4		2	3	60	0	1		2	80	1	36	1	1		33	30	7	1	5		2	2	6
5		3	4	60	0	1		3	70	1	35	1	1		32	29	8	2	5		3	3	6
6		4	5	60	0	13		4	60	1	34	1	1		31	28	14	2	4		4	4	6
7		5	6	60	0	1		5	50	1	21	1	1		24	21	15	3	4		5	5	6
8		7	8	30	1	34		6	40	1	20	1	1		23	20	21	3	3		6	6	6
9		8	9	30	0	1		7	30	1	2	1	1		2	2	20	2	3		7	1	5
10		9	10	30	0	-31		8	20	1	3	1	1		6	3	19	1	3		8	2	5
11		10	11	30	0	11		9	10	0	2	0	0				25	1	2		9	3	5
12		11	12	30	0	3		10	0	1	33	1	1		30	27	26	2	2		10	4	5
13		13	14	20	0	-2		11	10	1	22	1	1		25	22	32	2	1		11	5	5
14		14	15	20	1	34		12	20	1	19	1	1		22	19	33	3	1		12	6	5
15		15	16	20	0	-21		13	30	0	2	0	0				34	4	1		13	1	4
16		16	17	20	0	3		14	40	1	4	1	1		7	4	35	5	1		14	2	4
17		17	18	20	0	5		15	50	1	5	1	1		8	5	36	6	1		15	3	4
18		19	20	15	0	1		16	60	1	26	1	1		29	26	30	6	2		16	4	4
19		20	21	15	0	1		17	70	1	23	1	1		26	23	24	6	3		17	5	4
20		21	22	15	0	-19		18	80	1	18	1	1		21	18	18	6	4		18	6	4
21		22	23	15	0	1		19	90	1	8	1	1		11	8	12	6	5		19	1	3
22		23	24	15	0	7		20	80	1	7	1	1		10	7	6	6	6		20	2	3
23		25	26	12	1	34		21	70	1	6	1	1		9	6	5	5	6		21	3	3
24		26	27	12	0	8		22	60	1	25	1	1		28	25	11	5	5		22	4	3
25		27	28	12	0	-34		23	50	1	24	1	1		27	24	17	5	4		23	5	3
26		28	29	12	0	0		24	40	1	17	1	1		20	17	23	5	3		24	6	3
27		29	30	12	0	20		25	30	1	9	1	1		12	9	22	4	3		25	1	2
28		31	32	10	0	25		26	20	1	10	1	1		13	10	16	4	4		26	2	2
29		32	33	10	1	34		27	10	0	2	0	0				10	4	5		27	3	2
30		33	34	10	1	34		28	0	0	36	0	0				4	4	6		28	4	2
31		34	35	10	1	34		29	10	0	36	0	0				3	3	6		29	5	2
32		35	36	10	1	34		30	20	1	16	1	1		19	16	2	2	6		30	6	2
33		1	7	60	1	0		31	30	0	36	0	0				#N/A	#N/A	###		31	1	1
34		7	13	60	0	0		32	40	1	11	1	1		14	11	#N/A	#N/A	###		32	2	1
35		13	19	60	0	-6		33	50	1	12	1	1		15	12	#N/A	#N/A	###		33	3	1
36		19	25	60	1	34	n-2	34	60	1	13	1	1		16	13	#N/A	#N/A	###		34	4	1
37		25	31	60	0	-27		35	70	1	14	1	1		17	14	#N/A	#N/A	###		35	5	1
38		2	8	30	0	33	n	36	80	1	15	1	1		18	15	#N/A	#N/A	###		36	6	1
39		8	14	30	1	34		Revenues	1530	30							1	1	6				
40		14	20	30	0	-3		Eq 19	812														
41		20	26	30	0	-3																	
42		26	32	30	1	34																	
43		3	9	20	0	33																	
44		9	15	20	0	-3																	
45		15	21	20	1	34																	
46		21	27	20	0	4																	
47		27	33	20	0	-10																	
48		4	10	15	0	1																	
49		10	16	15	0	7																	
50		16	22	15	0	1																	
51		22	28	15	0	-11																	
52		28	34	15	0	23																	
53		5	11	12	1	34																	
54		11	17	12	1	34																	
55		17	23	12	1	34																	
56		23	29	12	0	-12																	
57		29	35	12	0	22																	
58		6	12	10	0	1																	
59		12	18	10	0	1																	
60		18	24	10	0	1																	
61		24	30	10	0	1																	
62		30	36	10	0	1																	
63		2	1	60	1	0																	
64		3	2	60	1	34																	
65		4	3	60	1	34																	
120		24	18	10	1	34																	
121		30	24	10	1	34																	
122		36	30	10	1	34																	
123		Costs		718																			

Table 5: Formulas for spreadsheet in Figure 15

Cell	Formula	Copied to	Name / Task
B63	=C3	B64:B122	Reverse arcs: stop→start
C63	=B3	C64:C122	Reverse arcs: start→stop
D63	=D3	D64:D122	Reverse arcs: copy costs
D123	=SUMPRODUCT(D3:D122;E3:E122)	-	*Computing costs*
F3	=IF(OR(B3=1;C3=1);0;INDEX($K$3:$K$38;B3)-INDEX($K$3:$K$38;C3) +$H$37*E3)	F4:F122	*Eq 16* (LHS)
I39	=SUMPRODUCT(I3:I38;J3:J38)	-	*Computing revenues*
I40	=I39-D123	-	*Eq 19*
L3	=SUMIF($C$3:$C$122;$H$3:$H$38;$E$3:$E$122)	L4:L38	*Eq 20* (LHS)
M3	=SUMIF($B$3:$B$122;$H$3:$H$38;$E$3:$E$122)	M4:M38	*Eq 9* (RHS)
E3:E122			*Var_x*
H36			*Param_n_2*
H38			*Param_n*
K4:K38			*Var_u*
J4:J38			*Var_y*
O3	=IF(J3=0;"";RANK(K3;$K$3:$K$38;1))	O4:O38	A preliminary rank of nodes
P3	=IF(J3=0;"";RANK(O3;$O$3:$O$38;1))	P4:P38	Rank of a node in the tour
Q3	=MATCH(U3;$P$3:$P$38;0)	Q4:Q38	Visiting sequence
Q39	=Q3	-	Last leg, return to depot
R3	=INDEX($U$3:$W$38;$Q3;2)	R4:R39	A visited node's x-coordinate
S3	=INDEX($U$3:$W$38;$Q3;3)	S4:S39	A visited node's y-coordinate

Figure 16 Solver settings for spreadsheet in Figure 15



Solver Parameters

Set Target Cell: Eq_19

Equal To: ● Max ○ Min ○ Value of: 0

By Changing Cells:

Var_x;Var_y;Var_u

Subject to the Constraints:

Eq_16 <= Param_n_2
Eq_20 = Eq_9
Eq_20 = Set_y
Var_u <= Param_n
Var_u >= 2
Var_x = binary

From Figure 15 we see that the salesman visits only 30 of the 36 nodes, with a net profit of $812. Notice that the tour actually includes node 10 with $0 benefits, but skip node 13 with $30 in benefits. Using the Standard Solver it takes more than a full weekend to find the optimal solution. An early test version of Excel 2010 (Technical Preview) with the new Standard Solver finished during an overnight run. Again, an assignment formulation applying a full *n*×*n* matrix of the *x* variables would include too many variables for the Standard Solver. Using the commercial PSP V9.04, the solution time in Excel is 9.06 seconds for the flow formulation. This also reveals an alternative optimal solution, visiting 32 nodes. The solution time when the assignment formulation is implemented is 11.45 seconds, on the same computer. In the assignment formulation a complete graph is assumed, and the extra variables are eliminated by extra constraints, thereby increasing the solution time.

Unfortunately the MTZ SEQs used (16) – (17) assumes node 1 to be the depot. Therefore we cannot apply this type of SEC if our goal is to find the optimal location for the depot.

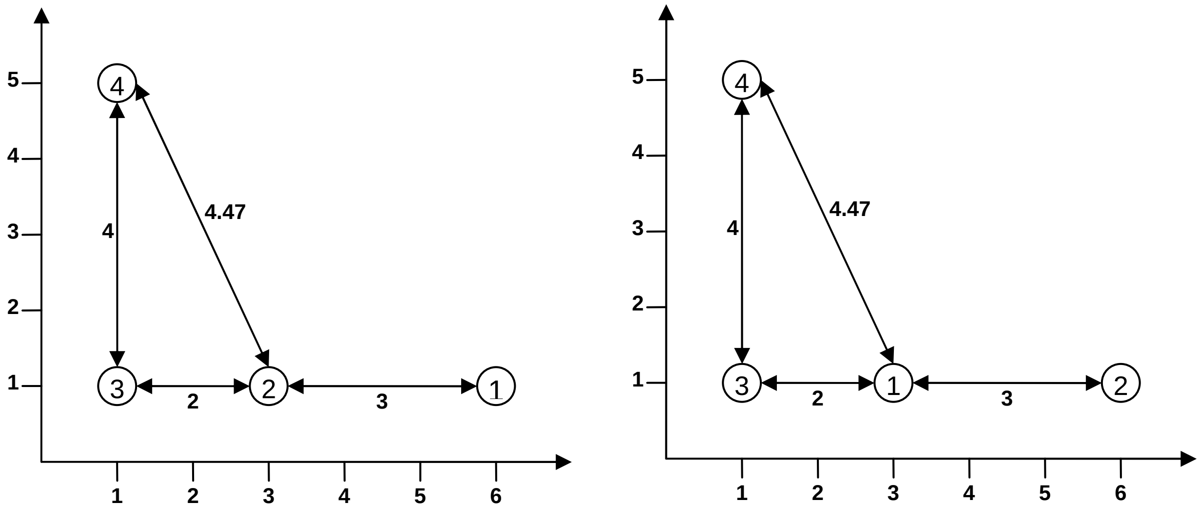
14. Pitfalls using MTZ SECs

For a closed tour the starting node can always be chosen arbitrarily, if all nodes have to be visited. Unfortunately, when applying the MTZ SECs, the solution may depend upon which node has been selected as the start node. When do we need to stay alert?

15. Non-complete graphs requiring multiple visits

Take a look at Figure 17. Two identical graphs are displayed, except that node 1 and 2 have been renumbered. You can easily find the optimal solution by visual inspection. Trying to solve one of them fails, whereas the other succeeds, if the MTZ SECs are applied in a closed TSP. There are two other nodes that could be renumbered as number 1. None of them will succeed if we try to solve using the MTZ SECs.

Figure 17 TSP in a non-complete graph, requiring multiple visits in a closed tour

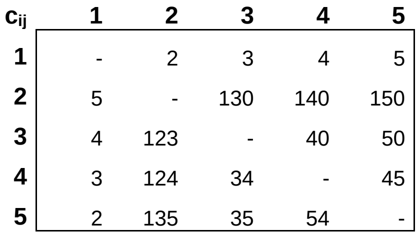


Since this is a non-complete graph, a spreadsheet layout similar to Figure 7 is probably most efficient, skipping equation (18), as it is hardly needed.

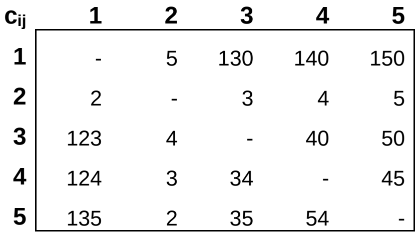
16. The triangle inequality is not always satisfied

Have a look at the cost matrixes in Table 6. The only difference is that once more the nodes 1 and 2 have been renumbered. This is a complete graph, as there is a direct link between any node to every other node.

Table 6: Identical cost matrixes of a complete graph, the triangle inequality does not apply



c_{ij}	1	2	3	4	5
1	-	2	3	4	5
2	5	-	130	140	150
3	4	123	-	40	50
4	3	124	34	-	45
5	2	135	35	54	-



c_{ij}	1	2	3	4	5
1	-	5	130	140	150
2	2	-	3	4	5
3	123	4	-	40	50
4	124	3	34	-	45
5	135	2	35	54	-

Solving one of them using the flow formulation (7) – (10) and the MTZ SECs (16) – (17) returns a solution with a minimal cost of 209, visiting every node once. Solving the other returns a minimum cost of 28, visiting four of the nodes once and visiting one node four times. The numbering of the nodes has thus a great impact on which solution is obtained. The assignment formulation would return a minimum cost of 209 instead of 28. See also Lee and Raffensperger (2006) for using AMPL teaching TSP, where the DFJ SECs are being implemented.

For a complete graph a spreadsheet layout similar to Figures 9 to 11 is probably most efficient, skipping columns AO – AR and the plot, since no coordinates are given in this example.

17. Conclusion

Solving TSP using general purpose optimisation tools like MIP solvers in spreadsheets has been regarded practical only for problems of a ‘small’ size. Recent advancements in these types of software have increased this limit; problems of size 358 of non-complete graphs have been solved in less than 10 minutes.

Such general purpose optimisation tools also allow for a greater variety of types of TSP, whereas procedures designed specifically for TSP often restrict the problem to a limited number of variants. In fact, we may overlook the best solution by applying the ‘standard’ approach using the assignment formulation or these specific tools for solving TSP. We must be absolutely sure our problem formulation is valid – all relevant costs and revenues have to be considered, and the constraints must not be too limiting. Otherwise we may end up solving the wrong problem. Anticipating a specific type of solution when formulating the problem is like starting at the wrong end, and may lead to a poor result.

However, the SECs needed in a TSP formulation may have unfortunate consequences and limitations. When using the MTZ SECs, the selection of the base node can be critical, even for a closed tour in a complete graph. It may also prohibit a solution, even in a connected graph.

We have further seen that formulation matters – including non-existing variables and eliminating them by adding non-existing constraints – can both increase solution time and cause problems in finding the optimal solution (the number of variables or constraints may even become too big for the solver). A ‘wide’ formulation will always include the optimal solution. A ‘tight’ formulation may help finding the optimal solution, but may also exclude the optimal solution. A ‘tight’ formulation may reduce or increase the solution time, this depends on the type of solver used and the problem at hand.

To play safe a ‘wide’ formulation seems like a good strategy. If the graph is not complete or the triangle inequality does not apply, both imply situations where multiple visits may be required, then a replacement of the cost matrix may be advisable. Then the costs should be replaced by a minimum cost matrix (you can use the shortest path formulation *$(n\text{-}1)^2$* times), which for many links may involve a lengthy tour visiting many nodes from node *i* to node *j*. The TSP model can then be ‘tight’, which may or may not be helpful for some solvers. Unfortunately the TSP model will then also be blind-eyed; it has no real track of the tour or how many visits are actually being made at each node. The TSP solution then only indicates the sequence of the visits, and ignores any revisits.

References

Chlond, M. J. (2008). 'Shasha's gridspeed puzzle', *INFORMS Transactions of Education*, Vol. 9(1), pp. 46–52. Available online at http://ite.pubs.informs.org/.

Dantzig, G., Fulkerson, D. and Johnson, S. (1954). 'Solution of a large scale traveling salesman problem', *Operations Research*, Vol. 2, pp.393–410.

Gutin, G. and Punnen, A. P. (eds) (2007). *The traveling salesman problem and its variations*, New York: Springer.

Lee J., and J.F. Raffensperger: (2006). 'Using AMPL for teaching the TSP', *INFORMS Transactions on Education*, Vol. 7(1), pp. 37-69. http://archive.ite.journal.informs.org/Vol7No1/LeeRaffensperger/

Malandraki, C. and Daskin, M. S. (1993). 'The maximum benefit Chinese postman problem and the maximum benefit traveling salesman problem', *European Journal of Operational Research*, Vol. 65, pp. 218–34.

Miller, C. E., Tucker, A.. W. and Zemlin, R. A. (1960). 'Integer programming formulation of traveling salesman problems', *Journal of ACM*, Vol. 7, pp.326–9.

Pataki, G (2003). 'Teaching integer programming formulations using the traveling salesman problem', *SIAM Review*, Vol. 45(1), pp. 116–23.

Ragsdale, C. T. (2001). *Spreadsheet Modeling and Decision Analysis*, Cincinatti: South-Western College Publishing.

Author Biography

Rasmus Rasmussen's teaching experience is mostly in the fields of business economics, finance and management science. His research interests are in the fields of applied management science related to problems in business economics, quite often using spreadsheets, the tool frequently used also in teaching.

Contact details

Rasmus Rasmussen
Molde University College
P.O.Box 2110
6402 Molde, Norway
Tel: +47 71 21 42 42
Fax: +47 71 21 41 00
rasmus.rasmussen@himolde.no