# The Exact Solution of Travelling Salesman by Mixed Integer Programming in Matlab

Jaromir Zahradka

July 10, 2022

# The Exact Solution of Travelling Salesman by Mixed Integer Programming in Matlab

Jaromír Zahrádka[1]

**Abstract.** This contribution comes up with a specific solution of the travelling salesman problem. The driver of hauler has to deliver, using his truck, goods from the depot to n customers. Each customer point of delivery is given by GPS coordinates. The objective of the solution is to select the sequence of delivery points so that firstly the travel distance and subsequently the total travel time are minimal. The driver visits all delivery points and returns to the depot. In this contribution, one general solution is presented using the bound-and-branche method and by using mixed integer linear programming implemented in M-function. The created algorithm can be used in general for any number n of customers.

**Keywords:** branch-and-bound, linear programming, Matlab, travelling salesman

**JEL Classification:** C64
**AMS Classification:** 68W04; 90C11, 05C20

## 1 The Travelling Salesman Problem

The travelling salesman problem (TSP) and its classical solutions are described e.g. in [1, 2, 4, 5]. Our solution came from the use of integer programming which was published in [7]. In [3] is presented one implementation of the TSP solution with Matlab programming.

### 1.1 Mathematical Formulation

The TSP can be defined as follows. Let $G_0 = (V, E)$ be a connected directed graph consisting of a set of $n+1$ nodes, seller depot ( $i = 0$ ) and customer locations ( $i = 1, \dots, n$ ), and a set $E$ of non-negatively weighted arcs between each pairs of corresponding nodes of the graph $G_0$. For easier reference, let $I = \{1, \dots, n\}$ be the set of $n$ customers, and $I_0 = \{0\} \cup I$. The constant $t_0$ means the time-moment when the dealer vehicle leaves the depot. Each customer can be visited only once at any time greater than $t_0$. The order of the customers visited is not limited, other than by the requirement that the duration of the seller's journey through all customers (terminated by return to the depot) be as short as possible. For each customer $i \in I$ let $m_i$ be the assumed service time associated with the unloading of goods and dealing with the customer.
Let $d_{ij}$ be the length of the path from $i$ - node to $j$ - node for all $i, j \in I_0$. Therefore $\mathbf{D} = \left(d_{ij}\right)_{i,j \in I_0}$ is the non-negative distance matrix. The matrix $\mathbf{D}$ can be, in general, an asymmetric one with zeros on the places of the main diagonal, i.e. $d_{ii} = 0$ for each $i \in I_0$. It is necessary that the triangular inequalities be satisfied for distances among nodes of graph $G_0$. Instead of the distance matrix $\mathbf{D}$ we will use, for our solution TSP, the time matrix $\mathbf{C} = \left(c_{ij}\right)_{i,j \in I_0}$. Each element $c_{ij}$ represents the pure travelling time of the seller from $i$ node to $j$ one. It is assumed that if the average speed $v$ of the vehicle among each two nodes is used, then the driving time $c_{ij}$ can be expressed $c_{ij} = \dfrac{d_{ij}}{v}$. In this case the travel time $c_{ij}$ is proportional to the distance $d_{ij}$. We assume that it is given the moment $t_0$ when the seller's vehicle leaves the depot.

[1] University of Pardubice, Department Mathematics and Physics, Studentská 95, 53210 Pardubice, jaromir.zahradka@upce.cz.

## 1.2 Mathematical Solution

The core of the practical TSP solution is to find the one cycle in the graph $G_0$ which includes all nodes of the graph and which gives the shortest total driving time. For this purpose, integer variables $x_{ij}$ for $i, j \in I_0$ are introduced, which can only take the values 0 or 1. The variables $x_{ij}$ are called binary variables. Value $x_{ij} = 1$ means that the arc from node $i$ to $j$ is included in the cycle and value $x_{ij} = 0$ means that the corresponding arc is not included. For systemic reason variables, $x_{ii}$ are used but all are fixed by the value zero, i.e. $x_{ii} = 0$, for each $i \in I_0$. Variables $x_{ij}$ are elements of a matrix $\mathbf{X} = \left(x_{ij}\right)_{i,j \in I_0}$. The number of flow variables $x_{ij}$ is $(n+1)^2$.

In our work we use other specific non-integer variables $t_i$, for each $i \in I$. Each $t_i$ indicates the moment when the seller leaves the $i$ 's customer location. By using variables $t_i$, it is guaranteed that the solution will be correct with all nodes during only one cycle in the graph $G_0$. The variables $t_i$ are included as $n$ elements of the vector $\mathbf{t} = (t_1, t_2, \ldots, t_n)$. The number of all flow variables is $(n+1)^2 + n$.

The solution of TSP is realized like the optimal solution of a mixed-integer linear programming problem:

$$\min_{(\mathbf{X},\mathbf{t})} \left\{ \sum_{i,j=0}^{n} c_{ij} \cdot x_{ij} + \sum_{i=1}^{n} \frac{1}{n \cdot u} \cdot t_i \right\} \text{ subject to} \tag{1}$$

$$x_{ij},\ i, j \in I_0 \text{ are binary, } x_{ij} \in \{0,1\} \tag{2}$$

$$(c_{ij} + u - t_0 - c_{0j})\, x_{ij} + t_i - t_j \le u - t_0 - c_{0j} - m_j, \quad i, j \in I,\ i \ne j \tag{3}$$

$$c_{0j}\, x_{0j} - t_j \le -t_0 - m_j, \qquad j \in I \tag{4}$$

$$\sum_{j \in I_0} x_{ij} = 1, \qquad i \in I_0 \tag{5}$$

$$\sum_{i \in I_0} x_{ij} = 1, \qquad j \in I_0 \tag{6}$$

$$x_{ii} = 0, \qquad i \in I_0 \tag{7}$$

$$0 \le x_{ij} \le 1, \qquad i, j \in I_0 \tag{8}$$

$$t_0 + c_{0j} + m_j \le t_j \le u, \qquad j \in I \tag{9}$$

In the expressed model (1) is minimized the linear optimization function

$$\sum_{i,j=0}^{n} c_{ij} \cdot x_{ij} + \sum_{i=1}^{n} \frac{1}{n \cdot u} \cdot t_i \tag{10}$$

The main part $\sum_{i,j=0}^{n} c_{ij} \cdot x_{ij}$ of the optimized function guarantees finding the cycle which takes the minimum amount of time. Due to the assumed constant average speed $v$, the total travel length is also minimal. In the second part $\sum_{i=1}^{n} \frac{1}{n \cdot u} \cdot t_i$ of the optimized function (10) is used the value of the constant $u$, which is defined as follows:

$$u = t_0 + \sum_{i=1}^{n} m_i + \sum_{j=0}^{n} \max_{i \in I_0} c_{ij} \tag{11}$$

The value of $u$ guarantees, with respect to the expected values of $t_i$, that the coefficients of flow variables $t_i$ in the optimized function (10) are so small that they do not change the optimal solution for flow variables $x_{ij}$, while the time variables $t_i$ are minimized. The use of terms with flow variables $t_i$ in the optimization function (10) is necessary. If these are not included, a solution could be generated with some values of $t_i$ greather than necessary. Our model prefers, from two shortest cycles (with opposite directions), the one that gives a smaller sum of $t_i$.

Constraint (3) defines $n(n-1)$ conditions between flow variables $x_{ij}$ and departure times $t_i$, $t_j$, for $i,j \in I$. In the case $x_{ij} = 1$, the inequality (3) expresses the relationship $t_j \geq t_0 + c_{0j} + m_j + t_i - u$. Due to the large enough value of $u$, the right side of inequality (3) can be only non-positive and the relationship is satisfied.
In the case $x_{ij} = 1$, the inequality (3) is reduced $t_i + c_{ij} + m_j \leq t_j, \quad i,j \in I$. This expresses that the departure time from the node $j$ has to be greater than or equal to the sum of the departure time $t_i$ (from node $i$), the travelling time $c_{ij}$ (from node $i$ to node $j$) and the service time $m_j$ in the node $j$. The created optimization process ensures that, in the case of $x_{ij} = 1$, the condition (3) is satisfied only by the equation $t_i + c_{ij} + m_j = t_j$.

The constraint (4) defines relations between flow variables $x_{0j}$ and $t_j$, $j \in I$. In the case $x_{0j} = 0$ the inequality expresses the relationship $t_0 + m_j \leq t_j$, $j \in I$. Departure time from the node $j$ is greater than or equal to the sum of departure time $t_0$ and service time $m_j$. In the case $x_{0j} = 1$ the inequality (4) expresses the relationship $t_0 + c_{0j} + m_j \leq t_j$. Departure time from the node $j$ is greater than or equal to sum of departure time $t_0$ from the depot, travelling time $c_{0j}$ from depot to node $j$ and service time $m_j$.

Statements (5) and (6) declare $2(n+1)$ equation constrains, which express that only one arc leads from each node and only one arc leads to each node. Statement (7) declares that each $x_{ii} = 0$.
The inequalities in (10) declare that the lower and upper bounds of variables $x_{ij}$ are 0 and 1. The inequalities in (11) express the bounds of flow variables (departure times) $t_j$, $j \in I$.

## 1.3 Transformation to Matlab

In the Matlab system the index 0 can not to be used, therefore all vector and matrix variables use the smallest index number 1. The distance matrix is transferred to the Matlab environment as matrix `D`, with the row and column indices `i,j = 1,2, … ,n+1`, where each component `D(i,j)` corresponds to the distance $d_{i-1\,j-1}$ of the nodes $i-1$ and $j-1$. Similarly each component `C(i,j)` of the time matrix corresponds to the driving time $c_{i-1\,j-1}$ from the node $i-1$ to the $j-1$ one.

Our created procedure for TSP solving in the Matlab code is included in the M-function *SOLVER_TSP.m* and it is fully listed as an Appendix at the end of the article. The input variables are `n` - number of customers, `D` – distance matrix, `v` – velocity of the vehicle, `t0` – the moment when the seller leaves the depot, and `m` – row vector with customer service duration times. The main output variable is the column vector `X` of flow variables, which is obtained as an output of the optimization via the command `intlinprog`.

The mixed-integer linear programming problem is generally expressed by

$$\min_X f^T \cdot X \text{ subject to} \begin{cases} X\,(intcon) \text{ are integers} \\ A \cdot X \leq b \\ A_{eq} \cdot X = b_{eq} \\ l_b \leq X \leq u_b \,. \end{cases} \tag{12}$$

The solver for this problem is the command `X=intlinprog(f,intcon,A,b,Aeq,beq,lb,ub)` in Matlab code (you can see it on the Appendix row No. 55). A more detailed explanation is in the User's Guide [6].

For the solution of TSP via the `intlinprog` command, all flow variables are arranged in a column vector `X` with $(n+1)^2 + n$ components. First $(n+1)^2$ flow variables are integer variables $x_{ij}$, and each variable $x_{ij}$, $i,j \in I_0$ is represented by Matlab flow variable `X(i*(n+1)+j+1,1)`. The last $n$ flow variables of `X` are the seller's departure times $t_1, t_2, \ldots, t_n$, and each variable $t_i$, $i \in I$ is represented by `X((n+1)^2+i,1)`.

The objective function of the mixed-integer linear programming problem (12) is, in the Matlab code, expressed like `f'*X`, where `f` is a column vector of coefficients with $(n+1)^2 + n$ components. The first $(n+1)^2$

components are elements of the time matrix `C` so that `f((i-1)*(n+1)+j,1)=C(i,j), i,j` $\in\{1,2,...,n+1\}$.

For the last $n$ components of `f` we use the value $\frac{1}{n \cdot u}$ according to relation (10) (the Appendix, row No. 2).

The vector *intcon* in the command *intlinprog* specifies of flow variables, which are taken integers, *intcon = 1:(n+1)^2*. They are first $(n+1)^2$ flow variables, i.e. variables $x_{ij}$.

The constraints (3) and (4) give the system of $n^2$ linear inequalities with $(n+1)^2+n$ variables. The matrix *A* of system inequalities and the column vector *b* of right sides are created for any $n$ in Matlab code statements on lines No. 3 to 7 in the Appendix. The constraints (5), (6) and (7) give the system of $n^2$ linear equalities with $(n+1)^2+n$ variables. The matrix *Aeq* of system equalities and the column vector *beq* of right sides are created for any $n$ in the Matlab code statements on lines No. 8 to 13 in the Appendix.

The last two input variables of the *intlinprog* command (2) are the column vectors *lb* and *ub* of lower and upper bounds of the flow variables. With respect to the relations (7), (8), (9) the components of vectors *lb* and *ub* are filled by commands on lines No. 14 to 17 in the Appendix

By installation of input variables *f, intcon, A, b, Aeq, beq, lb, ub* in the command *intlinprog*, and running it (the line No. 18), we get the optimal TSP solution, this is the vector of flow variables *X*. The values of the first $(n+1)^2$ variables (component of *X*), which have a value of 1, indicate the arcs that are part of the travel cycle. The last $n$ values of flow variables indicate times when the seller leaves individual customers.

The variables *X(k,1)*, *k* $\in\{1,2,...,(n+1)^2\}$, which take the value 1, determine the arcs of the shortest cycle. The commands from lines No. 20 to 24 allow the creation of a sequence of cycle nodes, i.e. the *CYCLE* vector. The first item of the *CYCLE* vector is the number 0 – depot, and the other n items are the sequence of customer numbers, and the last item is supplemented by the number 0 with regard to the fact that the seller returns to the depot.

The values of components of *X(k,1)*, *k* $\in\{(n+1)^2+1,(n+1)^2+2,...,(n+1)^2+n\}$ are the seller's departure times from the customer *k* at the optimal cycle. The vector of the departure times *t*, the time *tRet* of the seller's arrival back to the depot, and the total duration of the seller's business trip *AllWorkTime* are calculated on lines No. 25, 26. On lines No. 27 to 31 is created the vector *tArr* of arrival times to the nodes and the total distance *TotDist* traveled by the seller. The last item of the vector *tArr* means the time *tRet* when the seller returns to the depot. The input variables for the M-function *SOLVER_TSP* and its execution have to be done using a startup M-script that contains commands for drawing the output circle (Figure 1). The startup script is not listed in this article.

## 2 Illustrative Example

To illustrate the program we have created, we assume a seller and twelve customers. The GPS coordinates of the seller's depot are $E_0 = 14.068^\circ$ (the eastern longitude) and $N_0 = 49.427^\circ$ (the northern latitude). The GPS coordinates $E_i$, $N_i$ and the service times $m_i$ of customers you can find in Table 1.

| | Customer | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $E_i$ (°) | 14.436 | 14.174 | 14.026 | 14.955 | 14.431 | 14.962 | 14.762 | 14.007 | 14.680 | 14.706 | 14.645 | 14.552 |
| $N_i$ (°) | 49.221 | 49.452 | 49.017 | 49.266 | 49.358 | 49.090 | 49.168 | 49.094 | 49.161 | 49.202 | 49.274 | 49.024 |
| $m_i$ (min) | 16 | 13 | 13 | 13 | 12 | 20 | 20 | 19 | 18 | 12 | 14 | 12 |

**Table 1** The GPS coordinates and the service times of the customers

The distance between two customer locations (nodes) is taken as their orthonormal distance on the Earth sphere multiplied by a factor of 1.25. The orthonormal distance is calculated with a sphere radius R = 6371 km (mean radius of the Earth). All distances are included in the symmetric distance matrix **D** in Table 2.

By running the function *SOLVER_TSP.m* with the above chosen parameters, the optimal solution was found. The shortest cycle is given with a node sequence 0-2-5-1-11-4-6-7-10-9-12-3-8-0 and is drawn in Figure 1. Due to the symmetry of the matrix **D**, there is another solution that gives the same minimal travel distance and min. driving time. This is the opposite directed cycle 0-8-3-12-9-10-7-6-4-11-1-5-2-0, but its sum of departure times is greather.

| Distance (km) | | j | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| i | 0 | 0 | 58.19 | 15.11 | 55.59 | 125.18 | 51.30 | 132.28 | 102.57 | 45.70 | 92.30 | 93.71 | 82.80 | 86.44 |
| | 1 | 58.19 | 0 | 47.90 | 63.27 | 72.39 | 18.45 | 75.20 | 45.87 | 62.03 | 34.86 | 37.62 | 29.91 | 31.03 |
| | 2 | 15.11 | 47.90 | 0 | 62.14 | 111.40 | 37.90 | 119.87 | 90.22 | 53.55 | 80.50 | 81.24 | 69.71 | 77.98 |
| | 3 | 55.59 | 63.27 | 62.14 | 0 | 133.40 | 72.66 | 130.47 | 104.30 | 10.71 | 92.95 | 97.74 | 92.74 | 73.12 |
| | 4 | 125.18 | 72.39 | 111.40 | 133.40 | 0 | 73.88 | 23.65 | 29.88 | 133.78 | 40.74 | 35.66 | 43.10 | 64.77 |
| | 5 | 51.30 | 18.45 | 37.90 | 72.66 | 73.88 | 0 | 82.13 | 52.63 | 68.84 | 43.59 | 43.61 | 31.82 | 47.99 |
| | 6 | 132.28 | 75.20 | 119.86 | 130.46 | 23.65 | 82.13 | 0 | 29.71 | 132.74 | 40.34 | 38.63 | 50.52 | 57.67 |
| | 7 | 102.57 | 45.87 | 90.22 | 104.30 | 29.88 | 52.63 | 29.71 | 0 | 105.41 | 11.43 | 9.02 | 21.62 | 35.03 |
| | 8 | 45.70 | 62.03 | 53.55 | 10.71 | 133.78 | 68.83 | 132.74 | 105.41 | 0 | 93.98 | 98.24 | 91.93 | 76.34 |
| | 9 | 92.30 | 34.86 | 80.50 | 92.95 | 40.74 | 43.59 | 40.34 | 11.44 | 93.98 | 0 | 6.59 | 15.96 | 25.61 |
| | 10 | 93.71 | 37.62 | 81.24 | 97.74 | 35.66 | 43.60 | 38.63 | 9.03 | 98.24 | 6.59 | 0 | 12.87 | 32.11 |
| | 11 | 82.80 | 29.91 | 69.71 | 92.73 | 43.10 | 31.81 | 50.52 | 21.62 | 91.93 | 15.95 | 12.87 | 0 | 36.03 |
| | 12 | 86.44 | 31.03 | 77.98 | 73.12 | 64.77 | 47.99 | 57.67 | 35.03 | 76.34 | 25.61 | 32.11 | 36.03 | 0 |

**Table 2** The distance matrix **D**

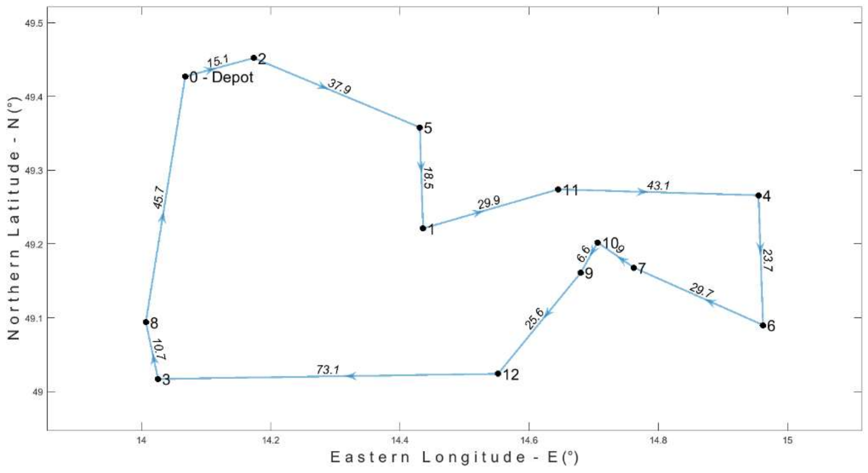

**Figure 1** The minimal length cycle of the seller around all customers

The calculated seller's departure times $t_{dep_i}$ from customers, and arrival times $t_{arr_i}$ to customers are in Table 3. The total travelled distance by the seller vehicle is 368.60 km and the total time of a seller's trip is 9 h 10 min.

| Times | Depot | Customers ranking in minimal cycle | | | | | | | | | | | Depot |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $i$ | 0 | 2 | 5 | 1 | 11 | 4 | 6 | 7 | 10 | 9 | 12 | 3 | 8 | 0 |
| $t_{arr_i}$ | - | 4:15 | 5:06 | 5:36 | 6:22 | 7:19 | 7:56 | 8:45 | 9:14 | 9:33 | 10:17 | 11:42 | 12:05 | 13:10 |
| $t_{dep_i}$ | 4:00 | 4:28 | 5:18 | 5:52 | 6:36 | 7:32 | 8:16 | 9:05 | 9:26 | 9:51 | 10:29 | 11:55 | 12:24 | - |

**Table 3** The arrive and depart times of the seller

## 3 Conclusion

This paper proposes a practical solution of the travelling salesman problem for any number $n$-customers in Matlab code. The TSP is formulated as a mixed-integer linear programming problem with a new approach, which respects the given matrix of distances and service duration times of customers, and the constant speed of the seller's movement. The solution lies in minimizing of the seller's trip duration that leads across all customers. The constant speed of seller's movement is assumed, therefore the total distance travelled is also the minimum. The created objective function guarantees that the total travelled distance and the total travelled time of the seller are minimal.

The main result of this article is the creation of the M-function (Appendix) which allows to solve the TSP generally for any number of $n$ customers. The created M-script is practically usable on a common personal computer for up to 30 customers. For 30 customers, the calculation takes less than 60 minutes, and for up to 20 customers, the calculation takes less then 40 seconds. M-script was successfully tested for a maximum of 40 customers.

The optimal solution of travelling salesman problem ensures the shortest travel distance and shortest duration of the business trip, and thus the best solution in terms of economic costs for the implementation of the business trip.

## Acknowledgements

The paper was supported by institutional support of University of Pardubice.

## References

[1] Bentley, J. J. (1992). *Fast Algorithms for Geometric Travelling Salesman Problems. ORSA Journal on Computing,* (4) 4.

[2] Gavendra, G. at al. (2010). Travelling Salesman Problem. In Tech, P*roceedings of the Theory and Applications*. Rieka.

[3] Gradle, K. P. & Mulley, Y. U. (2015). Travelling Salesman with MATLAB programming. *International Journal of Advances in Applied Mathematics and Mechanics,* (3) 2, 258-266.

[4] Gutin, G. & Punnen, A. P. (2007). The Travelling Salesman Problem and Its Variations. New York: Springer Science+Business Media, LLC.

[5] Jonak, R., Smutný, Z., Simunek, M. & Dolezel, M. (2020). Rout and Travel Time Optimalization for Delivery and Utility Services. *Acta Informatica Pragensia,* (2) 9, 200-209.

[6] Math Works. Inc. (2020). Optimalization Toolbox$^{TM}$. *User´s* Guide. Natick.

[7] Winston, W. L. (1994). Operations Research. Applications and Algorithms. Duxbury: Duxbury Press.

## Appendix

```
function [X, CYCLE, TotDist, AllWorkTime, tArr] = SOLVER_TSP(n, D, v, t0, m)
C=D/60; CT=C'; u=t0+sum(max(CT))+sum(m(1:n)); f=[CT(:);ones(n,1)/u/n];
p=(n+1)*(n+1); A=zeros(n^2,p+n); k=0;
for i=1:n; for j=1:n; if i~=j; k=k+1;
A(k,(n+1)*i+1+j)=C(i+1,j+1)+u-t0-C(1,j+1); A(k,p+i)=1; A(k,p+j)=-1;
b(k,1)=u-t0-C(1,j+1)-m(j); end; end; end
for i=1:n; k=k+1; A(k,1+i)=C(1,1+i); A(k,p+i)=-1; b(k,1)=t0-m(i);end
Aeq=zeros(3*n+3,(n+1)^2+n);
for i=1:n+1; for j=1:n+1; Aeq(i,(i-1)*(n+1)+j)=1; end
Aeq(i,(i-1)*(n+1)+i)=0; beq(i,1)=1; end
for i=1:n+; for j=1:n+1; Aeq(n+1+i,(j-1)*(n+1)+i)=1; end
Aeq(n+1+i,(i-1)*(n+1)+i)=0; beq(n+1+i,1)=1; end
for i=1:n+1; Aeq(2*n+2+i,(i-1)*(n+1)+i)=1; beq(2*n+2+i,1)=0; end
lb = zeros(p,1); for i=1:n; lb(p+i,1)=t0+C(1,1+i)+m(i); end
k=0; for i=1:n+1; for j=1:n+1; k=k+1;
if i==j; ub(k,1)=0; else ub(k,1)=1; end; end; end
for i=1:n; ub(p+i,1)=u; end; intcon = 1:p;
X = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub);
X(1:p)=round(X(1:p));
for i=2:n+1
if X(i)==1; CYCLE=0; Nok=2; CYCLE(Nok)=i-1; TEST=i; break; end; end
while TEST~=1; for j=1:n+1
if X((CYCLE(Nok))*(n+1)+j)==1; Nok=Nok+1; CYCLE(Nok)=j-1; TEST=j; break; end
end;end
for i=1:n; t(i)=X(p+i); end
tRet=t(CYCLE(end-1))+C(CYCLE(end-1)+1,1); AllWorkTime=tRet-t0;
tArr=[t0, t(CYCLE(2:end-1))-(m(CYCLE(2:end-1))), tRet],
tArr=hours(tArr), tArr.Format='hh:mm'; TotDist=0;
for i=1:(n+1); for j=1:(n+1)
if X((n+1)*(i-1)+j)==1; TotDist=TotDist+D(i,j); break; end
end; end
```