The Solution of Traveling Salesman Problem with Time Windows by MILP in Matlab Code

JAROMÍR ZAHRÁDKA
Department of East Bohemia Region,
Žampach 10, 53401 Žamberk,
CZECH REPUBLIC

Abstract: - A delivery driver uses his truck to distribute ordered goods from the warehouse (depot) to n customers. Each customer determines the delivery point (by GPS coordinates) and time window for the distribution of goods. This problem can be called the traveling salesman problem with time windows (TSPTW). The objective of the solution is to select the sequence of delivery points so that the travel distance and the total travel time are minimal. The necessary condition is that the deliverer leaves the warehouse, visits all delivery points (where he arrives at the required time windows), and returns to the warehouse. In this article, one exact robust solution of the TSPTW is presented by using mixed integer linear programming implemented in Matlab code. The created algorithm can be used for any number n of customers.

Key-Words: - Matlab, mixed-integer linear programming, MILP, optimization, traveling salesman problem with time windows, TSPTW.

Received: June 6, 2023. Revised: January 21, 2024. Accepted: February 8, 2024. Published: March 21, 2024.

1 Introduction

This article aims to create a functional Matlab algorithm that can find the optimal solution of the traveling salesman problem with time windows (TSPTW) for any given number n of customers.

The simple traveling salesman problem (TSP) lies in finding the shortest path or shortest time that a salesman should take to visit all n customers and return to the depot. Related works on the topic of TSP are [1], [2], [3], [4] and [5].

The TSPTW is an extension of the TSP which requires, in addition, that the traveling salesman arrive at the customers during pregiven time windows.

Our work is focused on the use of branch and bound, and branch and cut algorithms that have been used, for example, in studies [5], [6] and [7]. The TSPTW was solved in [8] and [9]. Authors used the compressed-annealing heuristic and the QUBO model.

The related problem of finding the real optimal road route from Start-point to End-Point is described by pseudocode for Dijkstra's algorithm in [10]. The minimum distances of the routes are obtained using google navigation which is also used in our work.

The TSP solution using Matlab code is demonstrated in [11]. The Matlab code for the solution of the traveling salesman problem was used in [5] and [7].

2 Mixed-integer Linear Programming

The problem of mixed-integer linear programming (MILP) is generally expressed by:

$$\min_{\boldsymbol{X}}(\boldsymbol{f}^T \cdot \boldsymbol{X}) \text{ subject to} \begin{cases} \boldsymbol{X}_{intcon} \text{ are integers} \\ \boldsymbol{A} \cdot \boldsymbol{X} \leq \boldsymbol{b} \\ \boldsymbol{A}_{eq} \cdot \boldsymbol{X} = \boldsymbol{b}_{eq} \\ \boldsymbol{l}_b \leq \boldsymbol{X} \leq \boldsymbol{u}_b \end{cases} \tag{1}$$

The vector $\boldsymbol{X}$ is the column vector of all flow variables. The symbol $\boldsymbol{f}$ denotes the column vector with coefficients represented in the objective function which is $\boldsymbol{f}^T \cdot \boldsymbol{X}$. The number of components of $\boldsymbol{f}$ is equal to the number of all flow variables. Writing $\boldsymbol{X}_{intcon}$ means the list of variables of the vector $\boldsymbol{X}$ that takes only the integer values. The other flow variables are treated as real variables.

The linear inequality constraint matrix $\boldsymbol{A}$ is specified as a matrix of real numbers, which are the linear coefficients in the system of inequality constraints given by $\boldsymbol{A} \cdot \boldsymbol{X} \leq \boldsymbol{b}$. The size of the matrix $\boldsymbol{A}$ is $m_{ineq} \times n$, where m_{ineq} is the number of inequality constraints and n is the number of flow variables. Linear inequality constraints vector $\boldsymbol{b}$ is the vector of given real numbers (right sides of the inequalities). The length of the vector $\boldsymbol{b}$ is m_{ineq}.

The linear equality constraint matrix $\boldsymbol{A}_{eq}$ is specified as the matrix of real numbers, which are the linear coefficients in the system of equations $\boldsymbol{A}_{eq} \cdot \boldsymbol{X} = \boldsymbol{b}_{eq}$. The size of the matrix $\boldsymbol{A}_{eq}$ is $m_{eq} \times n$,

where m_{eq} is the number of linear equation constraints and n is the number of flow variables. Linear equality constraint vector $\boldsymbol{b}_{eq}$ is the vector of given real numbers (right sides of the equations). The length of the vector $\boldsymbol{b}_{eq}$ is m_{eq}.

The lower and upper bounds of flow variables (items of $\boldsymbol{X}$) are specified as real numbers - elements of vectors $\boldsymbol{l}_b$, and $\boldsymbol{u}_b$. The vector $\boldsymbol{X}$ satisfies the inequality $\boldsymbol{l}_b \leq \boldsymbol{X} \leq \boldsymbol{u}_b$. The core of the solver for the MILP solution in Matlab code is the command:

X=intlinprog(f,intcon,A,b,Aeq,beq,lb,ub), (2)

where previously used variables are expressed by the corresponding variable identifiers of Matlab langu-age. The transformation of all variable labels into Matlab identifiers is contained in Table 1. A more detailed explanation of the intlinprog command is available in [11].

3 Mathematical Formulation of the Traveling Salesman Problem with Time Windows

The traveling salesman problem with time windows (TSPTW) can be defined as follows. Let $G_0 = (V, E)$ be a connected graph consisting of a set V of $n+1$ nodes indexed by $= 0, 1, \cdots, n$, and a set E of non-negatively weighted arcs between pairs of corresponding nodes of the graph G_0. The index $i = 0$ is used for the seller, and indexes $i = 1, \cdots, n$ for customers. For easier reference, let $I = \{1, \cdots, n\}$ be the set of customers, and $I_0 = I \cup \{0\}$. Each of the customers $i \in I$ can be reached only within a specified time interval (window) $[l_{w_i}, u_{w_i}]$. Vectors of lower and upper boundaries of the time window are denoted by $\boldsymbol{l}_w$ and $\boldsymbol{u}_w$. For each customer $i \in I$ let m_i be the service time associated with the handover and unloading of goods. The vector $\boldsymbol{m}$ of customer service times means $\boldsymbol{m} = (m_1, \cdots, m_n)$. The constant t_0 means the moment when the vehicle can first leave the depot.

Let d_{ij} be the length of the one-way path (in meters) and c_{ij} the one-way time-distance (in seconds), i.e. the travel duration from node i to node j for all $i, j \in I_0$. Therefore $\boldsymbol{D} = (d_{ij})_{i,j \in I_0}$ is a distance matrix and $\boldsymbol{C} = (c_{ij})_{i,j \in I_0}$ is a time-distance matrix. Both matrices $\boldsymbol{D}$ and $\boldsymbol{C}$ are non-negative, asymmetric, and related (but essentially independent non-negative), with zeros on the main diagonal, i.e. $c_{ii} = 0$ and $d_{ii} = 0$ for each $i \in I_0$. It is necessary to satisfy the triangular inequalities among all nodes of the graph G_0 in both matrices.

Since we have to respect time windows, it is necessary that the time-distance matrix $\boldsymbol{C}$ be used for optimization. The elements of the time-distances matrix c_{ij} are not strictly proportional, in general, to the corresponding elements of the distance matrix d_{ij}. The driving time between two nodes is significantly influenced by the quality of the roads used.

To start the optimization process, it is necessary to select the time moment t_0 when the vehicle leaves the depot. Regarding the lower valid time windows of all customers, the appropriate time t_0 is:

$$t_0 = \min_{j \in I} \left(l_{w_j} - c_{0j} \right) \quad (3)$$

But when the first visited customer $s \in I$ is selected during the optimization process, for which it is valid that $l_{w_s} - c_{0s} > t_0$, then a waiting time $l_{w_s} - c_{0s} - t_0$ will be required. To eliminate the unnecessary waiting time at the first customer, it is appropriate to shift the departure time of the vehicle from the depot to the real value $t_{Start} = l_{w_s} - c_{0s}$.

The core of the TSPTW solution is to find the cycle in the graph G_0 which contains all nodes of the graph so that the travel time is the shortest, and where all time windows are respected. 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 value $x_{ij} = 1$ means that the arc from the node i to j is included in the cycle. The value $x_{ij} = 0$ means that the corresponding arc is not included. For a systemic reason, variables x_{ii} are included, but all these variables are fixed by the value zero ($x_{ii} = 0$), for each $i \in I_0$. Variables x_{ij} are elements of a matrix $\boldsymbol{X} = (x_{ij})_{i,j \in I_0}$. The number of all variables x_{ij} is $(n+1)^2$.

Other flow variables that need to be used in the TSPTW solution are the vehicle departure times from all customers. Therefore real (non-integer) flow variables t_i are introduced for each $i \in I$. The variable t_i indicates the moment when the driver leaves the customer number i. The variables t_i are arranged as n elements of a vector $\boldsymbol{t} = (t_1, \cdots, t_n)$. So, the number of all flow variables is $(n+1)^2 + n$.

The solution of TSPTW is realized by the optimal solution of the mixed-integer linear programming problem:

$$\min_{(\boldsymbol{X}, \boldsymbol{t})} \left\{ \sum_{i,j=0}^{n} c_{ij} \cdot x_{ij} + \sum_{i=1}^{n} k_f \cdot t_i \right\} \text{ subject to} \quad (4)$$

$$x_{ij}, i, j \in I_0 \text{ are integers, even true that } x_{ij} \in \{0,1\} \quad (5)$$

$$\left(c_{ij} + u_{w_i} + m_i - l_{w_j} \right) x_{ij} + t_i - t_j \leq u_{w_i} - l_{w_j} + m_i - m_j, \quad i, j \in I, i \neq j \quad (6)$$

$$\left(c_{0j} + t_0 + l_{w_j}\right) x_{0j} - t_j \leq -l_{w_j} - m_j, \;\; j \in I \tag{7}$$

$$\sum_{j \in I_0} x_{ij} = 1, \;\; i \in I_0 \tag{8}$$

$$\sum_{i \in I_0} x_{ij} = 1, \;\; j \in I_0 \tag{9}$$

$$x_{ii} = 0, \;\; i \in I_0 \tag{10}$$

$$0 \leq x_{ij} \leq 1, \;\; i,j \in I_0, \; i \neq j \tag{11}$$

$$\max\left(l_{w_j} + m_j;\; t_0 + c_{0j} + m_j\right) \leq t_j \leq u_{w_j} + m_j, j \in I \tag{12}$$

In the above expressed model (4), with flow variables x_{ij} and t_i, the linear function:

$$\sum_{i,j=0}^{n} c_{ij} \cdot x_{ij} + \sum_{i=1}^{n} k_f \cdot t_i \tag{13}$$

is minimized. The first part, $\sum_{i,j=0}^{n} c_{ij} \cdot x_{ij}$ guarantees finding the cycle, which takes the minimum travel time. For this, in the optimization process, it is necessary that the second part of the minimized function (13), i.e. $\sum_{i=1}^{n} k_f \cdot t_i$ will be smaller by at least 10 times. This condition is certainly met if we choose the coefficients:

$$k_f = \frac{\min\limits_{i,j \in I_0,\, i \neq j} (c_{ij})}{86400\, n} \; . \tag{14}$$

It is assumed that the values of the variables t_i do not exceed the order of 86400 sec (1 day).

This objective function guarantees that the members containing the variables t_i will not affect the optimal values of the variables x_{ij}. At the same time, the arrival times t_i will be minimized and downtimes of vehicles will not be necessary before reaching customers.

Using statement (5) it is given that x_{ij}, $i,j \in I_0$ are all integer variables, even binary ones.

If $x_{ij} = 0$, the inequality (6) expresses the relationship $0 \leq \left(u_{w_i} + m_i\right) - t_i + t_j - \left(l_{w_j} + m_j\right)$, for each $i,j \in I$, $i \neq j$. The right side of this inequality can only be nonnegative because the upper bound of time window u_{w_i} plus service time m_i is greater than or equal to the departure time t_i (from the node i), and the departure time t_j (from the node j) is greater than or equal to the sum of lower bound of the time window l_{w_j}, and the service time m_j.

In the case $x_{ij} = 1$ the inequality (6) defines a relationship $c_{ij} + t_i + m_j \leq t_j$, $i,j \in I$, $i \neq j$. This expresses the condition that the departure time t_j (from node j) is greater than or equal to the sum of the departure time t_i (from node i), the traveling time c_{ij} from node i to the j one, and the service time m_j.

The Constraint (7) defines n relations between the flow variables x_{0j} and t_j, for each $j \in I$. In the case $x_{0j} = 0$ the inequality (7) expresses a relationship $l_{w_j} + m_j \leq t_j$, i.e. the departure time t_j from node j is greater than or equal to the sum of the lower bound of the time window l_{w_j} and service time m_j. In the case $x_{0j} = 1$ the inequality expresses the relationship $t_0 + c_{0j} + m_j x_{0j} \leq t_j$. The departure time from node j is greater than or equal to the sum of the departure time t_0 (from the depot), traveling time c_{0j} (from the depot to the node j), and the service time m_j.

Statements (8), (9) and (10) declare $3\,(n+1)$ equation constraints. The inequalities in (11) declare the lower and upper bounds (0 and 1) for variables x_{ij}, $i,j \in I$, $i \neq j$, $x_{ij} \in \{0,1\}$. The inequalities in (12) enforce permitted limits for departure times t_j, $j \in I$.

4 Transfer of TSPTW Model to the Matlab Environment

The procedure for solving TSPTW in Matlab code is contained in the M-function TSPTW_SOLVER.m, which is listed as an Appendix at the end of this article.

For solving the TSPTW in the Matlab environment, it is necessary to reliably modify the used variables following Matlab rules. The main problem with the transformation is that the index 0 cannot be used in Matlab, therefore, some variables need to be reindexed. An overview of the main variables transformed into the Matlab is contained in Table 1., where identifier p substitutes (n+1)^2.

We assume the vehicle goes around to n customers $\{1, 2, \cdots, n\}$. Each customer requests the arrival of a vehicle in a certain time window. The lower and upper boundaries of the time windows and service times for all customers are stored in M-vectors lw, uw and m. For departure time t_0 from the depot, the identifier t0 is used in Matlab. The distance matrix $\boldsymbol{D}$ and the travel time matrix $\boldsymbol{C}$ have to be transferred to the Matlab environment as matrices D and C thus $\mathrm{D(i,j)} \equiv d_{i-1\,j-1}$ and $\mathrm{C(i,j)} \equiv c_{i-1\,j-1}$ for indexes $i,\, j \in \{1, 2, \cdots, n+1\}$.

For the solution of TSPTW via the intlinprog command, all flow variables have to be arranged in a column vector X. First $(n+1)^2$ flow variables of vector X are elements x_{ij} of the matrix $\boldsymbol{X}$. Each variable x_{ij}, $i,j \in I_0$ is represented in Matlab code by

flow variable X(i*(n+1)+j+1,1). Each variable t_i, $i \in I$ is represented in Matlab code by flow variable X((n+1)^2+i,1).

In the intlinprog command of Matlab the objective function (13) is expressed like f'*X, where f is a column vector with $(n+1)^2+n$ components. The first $(n+1)^2$ components of the vector f are elements of travel time matrix C such that f((i-1)*(n+1)+j,1)=C(i,j), i,j $\in \{1,2,\cdots,n+1\}$, and the last n components have the same value k_f according to relation (14). The value of k_f (kf in Matlab code) is calculated by commands on lines No. 2 to 8 of the Appendix. The vector f of all components of the objective function are created in Matlab code on line No. 9.

The value t_0 according to (3) is introduced in Matlab code on row No. 10 of the Appendix.

The constraints (5), (6) are transformed to the Matlab code by inequalities
(C(i+1,j+1)+uw(i)+m(i)-lw(i))*X(i*(n+1)+j+1,1)+...
X(p+i,1) - X(p+j,1)<= uw(i)-lw(j)+m(i)-m(j), for all indexes i, j$\in \{1,2,\ldots,n\}$, i $\neq$ j, and
(C(1,j+1) +t0 -lw(j))*X(j+1,1) -X(p+j,1)<= -lw(j)-m(j), for j $\in \{1,2,\ldots,n\}$.

Concerning the last two inequalities, we can fill in the elements of matrix A and vector b, which are essential input parameters of the intlinprog function. The corresponding Matlab code can be found on rows No. 11 to 24 of the Appendix.

Concerning equalities (7), (8), (9), we can fill the elements into the matrix Aeq and vector beq. The relevant Matlab code can be found on rows No. 25 to 43 of the Appendix.

Two input variables lb, ub of the intlinprog command (2) are the lower and upper bounds of the flow variables. Concerning the relations (9), (10), (11) the relevant components are filled in using the commands on lines No. 44 to 63 of the Appendix.

The vector $\boldsymbol{X}_{intcon}$ of integer variables is, in the Matlab, coded by the vector intcon, and specifies all indices of flow variables, which are taken as integers. It uses the command intcon=1:p on row No. 64, p=(n+1)^2. This means that all flow variables x_{ij} are taken as integers.

By installing the input variables f, intcon, A, b, Aeq, beq, lb, and ub, and running the command intlinprog (on row No. 65), we get the optimal TSPWT solution if it exists. The optimal solution is given by the optimal values of components of flow variables vector X. The optimal TSPWT solution may not exist if the set of time windows cannot be met due to too short time windows or too long-time travel distances between customers.

While the variable X(k,1), k $\in \{1,2,\ldots,(n+1)^2\}$ takes the value 1, the corresponding arc between the nodes (customers) of the graph is part of the traveller's cycle. By processing these variables, we can obtain the order of passing the vertices, i.e. components of the Matlab vector CYCLE, as the code shows on lines No. 66 to 79 of the Appendix.

The last flow variables X(k,1), for k $\in \{p+1,p+2,\ldots,p+n\}$, where $p=(n+1)^2$, are specifically the departure times of the vehicle from the nodes (customers) k at the optimal cycle. The vector of the departure times $\boldsymbol{t}$ is created on line No. 80 of the Appendix.

The real departure time t_{Start} (tStart in Matlab) of the vehicle from the depot is calculated by a command on line No. 81. The time t_{Ret} (tRet in Matlab) of the vehicle's arrival back to the depot after visiting all customers is calculated by a command on line No 82. The total duration of the business trip $t_{TotDur}=t_{Ret}-t_{Start}$ is calculated on line No 83.

Table 1. The transformations of variable labels to Matlab identifiers

Variable label	n	$\boldsymbol{A}$	$\boldsymbol{b}$	$\boldsymbol{A}_{eq}$	$\boldsymbol{b}_{eq}$	$\boldsymbol{C}$	$\boldsymbol{D}$
Matlab	n	A	b	Aeq	beq	C	D
Variable label	$\boldsymbol{l}_b$	$\boldsymbol{u}_b$	$\boldsymbol{l}_w$	$\boldsymbol{u}_w$	$\boldsymbol{f}$	k_f	$\boldsymbol{m}$
Matlab	lb	ub	lw	uw	f	kf	m
Variable label	d_{ij}		x_{ij}				
Matlab	D(i+1,j+1)		X((n+1)*i+j+1,1)				
Variable label	c_{ij}		$\boldsymbol{t}$				
Matlab	C(i+1,j+1)		X(p+1:end,1)				
Variable label	$\boldsymbol{X}$	$\boldsymbol{X}_{intcon}$	t_0	t_i			
Matlab	X	intcon	t0	X(p+i)			
Variable label	t_{Start}	t_{Ret}	t_{TotDur}	$d_{TotDist}$			
Matlab	tStart	tRet	TotDur	TotDist			

For a clear output of the gained solution it is useful to create vectors of departure and arrival times t_{Arr} (tArr in Matlab) and t_{Dep} (tDep in Matlab) from and to customers with items in the order of the shortest cycle nodes. It is useful to extend the vector of departure times $\boldsymbol{t}_{Dep}$ by adding a new first item t_{Start} (the departure time from the depot). For the last item of the arrival times vector $\boldsymbol{t}_{Arr}$, it is useful to add t_{Ret} (the arrival time to the

depot). It is also good to arrange the lower and upper bounds of time windows $\boldsymbol{l_w}$ and $\boldsymbol{u_w}$ in the order of the nodes visited. The vectors $\boldsymbol{t_{Dep}}$, $\boldsymbol{t_{Arr}}$, $\boldsymbol{l_w}$ and $\boldsymbol{u_w}$ are created and modified using commands on lines No. 84 to 91. The output time items of all mentioned vectors are expressed in the clock time format 'hh:mm'. The total distance $t_{TotDist}$ (TotDist in Matlab) travelled during the sales trip is calculated by executing the commands on lines No. 92 to 99.

5 Application

Our created optimization program was applied to the delivery of frozen and refrigerated goods from a central warehouse (vehicle depot) in a company in the Czech Republic. The optimization of one line with ten customers is chosen for the presentation. The GPS coordinates of the vehicle depot are longitude $E_0 = 15.9084^0$ and latitude $N_0 = 50.0266^0$. The customers GPS coordinatesE_i, N_i, time window lower and upper bounds l_{w_i}, u_{w_i} (clock), and service times m_i (minutes) are given in Table 2.

Table 2. The customer's GPS coordinates, time window bounds, and service times

i	E_i (⁰)	N_i (⁰)	l_{w_i}	u_{w_i}	m_i (min)
1	15.7741	49.9841	6:00	8:00	23
2	16.4334	49.9015	10:00	14:00	21
3	14.5875	49.9039	6:00	11:00	22
4	14.8589	49.9939	6:00	8:00	21
5	14.5668	50.0419	6:00	12:00	21
6	16.2156	49.9883	8:00	14:00	22
7	15.3376	49.9463	6:00	8:00	24
8	14.0708	49.9624	6:00	9:00	23
9	14.2490	49.9196	6:00	12:00	21
10	16.2042	49.9052	8:00	13:00	22

The depot and customer locations are plotted in Figure 1. The data of distances and time distances bet-ween each two nodes have been obtained directly from navigation applications. All distances and time distances are contained in the distance matrix $\boldsymbol{D}$ and time distance matrix $\boldsymbol{C}$ in Table 3 and Table 4.

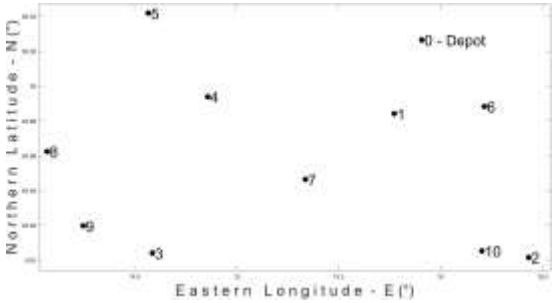

Fig. 1: Positions of the depot and customers according to GPS coordinates

Table 3. The distance matrix $\boldsymbol{D}$

Dist. (m)		j					
		0	1	2	3	4	5
i	0	0	14172	49463	111603	85370	109170
	1	141780	0	59899	102123	75890	99690
	2	49454	59734	0	156662	130429	154229
	3	112058	102104	156666	0	26711	20196
	4	85750	75796	130358	26609	0	24176
	5	109477	99523	154085	20138	24130	0
	6	27459	40398	22489	137326	111093	134893
	7	51454	39296	93858	63847	37614	61414
	8	144563	137484	197204	50579	65232	42444
	9	135290	129140	183702	35563	53747	33171
	10	29914	40194	19780	137122	110889	134689

Dist. (m)		j				
		6	7	8	9	10
i	0	27459	50951	143159	137742	29893
	1	40571	39392	136367	128162	40329
	2	22489	93831	196394	182801	19767
	3	137338	63824	50661	34512	137096
	4	111030	37516	66234	52748	110788
	5	134754	61243	42792	33726	134515
	6	0	74495	177058	163465	13220
	7	74530	0	103161	89986	74288
	8	177876	103944	0	18336	177634
	9	164374	90860	18382	0	164132
	10	13241	74291	176854	163261	0

Table 4. The time distance matrix $\boldsymbol{C}$

Time (sec)		j					
		0	1	2	3	4	5
i	0	0	678	1955	4646	3492	4454
	1	678	0	2346	4314	3160	4122
	2	1956	2329	0	6382	5227	6189
	3	4644	4312	6379	0	1199	945
	4	3486	3154	5221	1194	0	1002
	5	4446	4114	6180	942	1000	0
	6	1238	1643	1050	5695	4541	5503
	7	2137	1637	3704	2735	1581	2543
	8	6088	5773	7938	2202	2849	1913
	9	5642	5325	7391	1470	2211	1467
	10	1202	1574	769	5627	4473	5435

Time (sec)		j				
		6	7	8	9	10
i	0	1238	2142	6133	5684	1201
	1	1662	1637	5810	5352	1593
	2	1050	3705	7982	7419	770
	3	5695	2733	2208	1488	5625
	4	4537	1575	2861	2232	4468
	5	5496	2535	1925	1485	5427
	6	0	3018	7296	6733	548
	7	3020	0	4315	3773	3951
	8	7254	4271	0	903	7185
	9	6707	3746	906	0	6638
	10	548	2950	7227	6664	0

5.1 The Smallest Cycle Duration with Respecting Time Windows

By running the TSPTW_SOLVER.m function with the above given input parameters (Table 2 and Table 4), the optimal solution was found. The smallest duration cycle concerning time windows is given by

the sequence of nodes 0-1-7-4-8-9-3-5-10-2-6-0. The found cycle is drawn in Figure 2. The total time spent on the delivery route is 8 h 49 min 19 sec and the corresponding travel distance is 435.650 km.

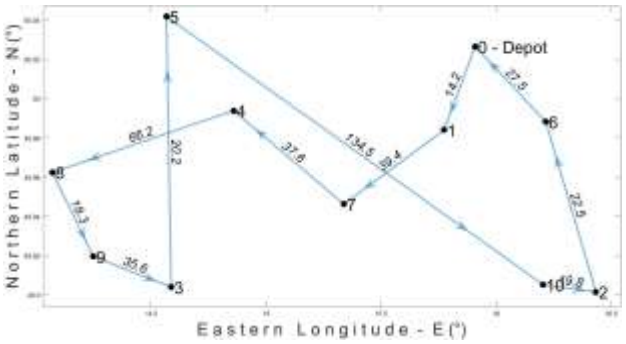

Fig. 2: The smallest duration cycle of the seller's journey with respecting the time windows

Table 5. The arrival and departure times of the smallest duration cycle concerning time windows

Time (clock)	Start	The arrived and dep. times of cycle 0-1-7-4-8-9-3-5-10-2-6-0					Re-turn
i	**0**	**1**	**7**	**4**	**8**	**9**	-
l_{w_i}	-	6:00	6:00	6:00	6:00	6:00	-
t_{Arr_i}	-	6:00	6:50	7:40	8:49	9:27	-
t_{Dep_i}	5:48	6:23	7:14	8:01	9:12	9:48	-
u_{w_i}	-	8:00	8:00	8:00	9:00	12:00	-
i	-	**3**	**5**	**10**	**2**	**6**	**0**
l_{w_i}	-	6:00	6:00	8:00	10:00	8:00	-
t_{Arr_i}	-	10:12	10:50	12:42	13:16	13:55	14:38
t_{Dep_i}	-	10:34	11:11	13:04	13:37	14:17	-
u_{w_i}	-	11:00	12:00	13:00	14:00	15:00	-

The calculated vehicle departure and arrival times t_{Dep_i}, t_{Arr_i}, and the lower and upper bounds l_{w_i}, u_{w_i} of time windows are in Table 5. The time values shown in the table are rounded to the nearest minute.

5.2 The Smallest Duration Cycle Without Time Windows

If all customers do not require any time window, then there exists n! different cycles that the seller can use to visit all customers. The mean value of the travel time after all possible cycles (without service times) is $\sum_{j=0}^{n}\left(\frac{1}{n}\sum_{i=0}^{n} c_{i,j}\right)$. We can find the cycle of actual minimum travel time. For this purpose, we modify the previous mixed-integer linear programming problem by cancelling constraints (5), (6), and changing (11) to a constraint:

$$t_e + m_j \le t_j \le t_e + \sum_{j=0}^{n}\left(\frac{1}{n}\sum_{i=0}^{n} c_{i,j}\right) + \sum_{i=1}^{n} m_i\,, j \in I. \quad (15)$$

We assume that all customers accept the arrival of the vehicle at the earliest $t_e = 6$ a. m. By running this modified M-function, the solution for the cycle of minimal travel time can be found. This is the node sequence 0-6-2-10-1-7-4-5-8-9-3-0, and the corresponding cycle is shown in Figure 3. To solve the described TSP, the Matlab code published in [4] can also be used.

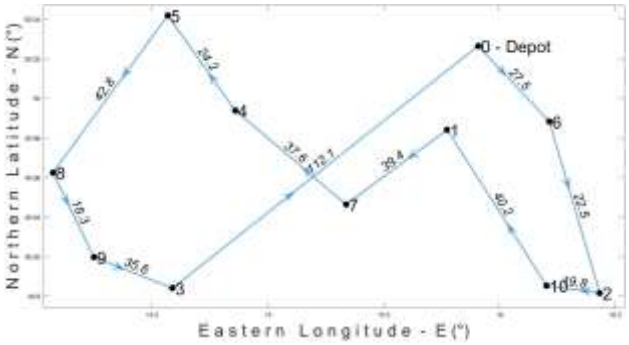

Fig. 3: The smallest duration cycle of the seller's journey without respecting the time windows.

The calculated vehicle departure times $t_{\mathrm{Dep_i}}$ and arrival times $t_{\mathrm{Arr_i}}$ for each node i are included in Table 6. The time values shown in the table are rounded to the nearest minute. This business trip lasts exactly 8 h 36 min 34 sec and the corresponding travel distance is 419.740 km. This means in our case that, when respecting the time windows, the travel time is 12 min and 45 seconds longer and the distance traveled is 25.910 km longer.

Table 6. The arrival and departure times of the smallest duration cycle without time windows

Time (clock)	Start	The arrived and dep. times of cycle 0-6-2-10-1-7-4-5-8-9-3-0					Re-turn
i	**0**	**6**	**2**	**10**	**1**	**7**	-
t_{Arr_i}	-	6:00	6:39	7:13	8:01	8:51	-
t_{Dep_i}	5:39	6:22	7:00	7:35	8:24	9:15	-
i	-	**4**	**5**	**8**	**9**	**3**	**0**
t_{Arr_i}	-	9:42	10:19	11:12	11:51	12:36	14:15
t_{Dep_i}	-	10:19	10:40	11:35	12:12	12:58	-

5.3 The Smallest Length Cycle without Time Windows

As is clear from the elements of the used matrices $\boldsymbol{C}$ and $\boldsymbol{D}$, the time distances between the same pairs of nodes are not exactly proportional to the length distances. The reason may be, for example, the different quality of the roads used. Driving time is shorter on higher quality roads than when using

lower class ones. The consequence may be that the shortest cycle in time may not be the shortest in length and vice versa.

We can present it in our case. If we optimize our TSP problem without time windows based on the use of the distance matrix $\boldsymbol{D}$, we obtain the optimal solution, given the node sequence 0-5-8-9-3-4-7-1-10-2-6-0, which is shown in Figure 4. For optimization we used the code published in [5].

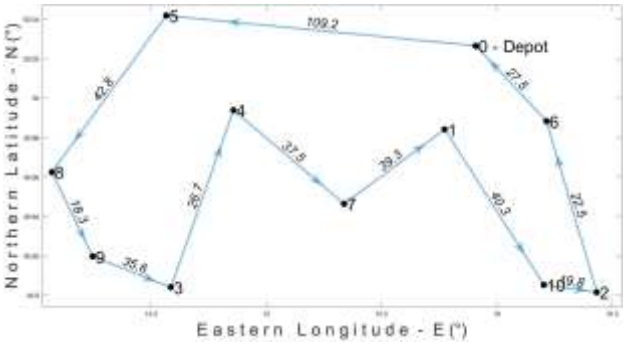

Fig. 4: The minimal length cycle without respecting time windows

The length of this cycle is 419.441 km and the total time spent on the delivery route is 8 h 36 min 53 sec. The cycle length is 299 m less than the cycle length in the previous case, but the travel time is 19 s longer. The last obtained cycle is minimal in terms of travel distance but not minimal in terms of time. The calculated vehicle departure times t_{Dep_i} and arrival times t_{Arr_i} (in hh:mm) for each node i are included in Table 7.

Table 7. The arrival and departure times of the smallest duration cycle without time windows

Time (clock)	Start	The arrived and dep. times of cycle 0-5-8-9-3-4-7-1-10-2-6-0					Re-turn
I	**0**	**5**	**8**	**9**	**3**	**4**	-
t_{Arr_i}	-	6:00	6:53	7:31	8:16	8:58	-
t_{Dep_i}	4:45	6:21	7:16	7:52	8:38	9:19	-
i	-	**7**	**1**	**10**	**2**	**6**	**0**
t_{Arr_i}	-	9:45	10:37	11:26	12:01	12:40	13:20
t_{Dep_i}	-	10:09	11:00	11:48	12:22	13:02	-

6 Conclusion

This paper proposes a new practical algorithm for the TSPTW solution for any number of customers, by using a developed Matlab code. The TSPTW is formulated as a mixed-integer linear programming problem with a new approach, which respects the given matrix of distances, time windows, service duration times of customers, and constant speed of the vehicle. The solution lies in minimizing the vehicle trip duration that services all nodes (customers). In practice, there is not strict proportionality between the time and the length of the path connecting the nodes. Therefore, the shortest path in time may not be the shortest in length. The designed objective function using members with appropriate multiples of departure times allows minimization of customer departure times and avoids unnecessary waiting times.

The main output of this paper is a computational algorithm implemented in an M-function created in Matlab code, which allows a general solution of the newly formulated TSPTW problem for any number n of customers. The created M-function TSPTW_SOLVER.m is given in the Appendix at the end of this paper.

The application in section 5 shows one practical TSPWT solution for 10 customers on an ordinary PC, this calculation takes 7.23 sec. For comparison, the optimal solutions for the shortest time and the shortest distance when no time windows are applied, are shown.

The M-function TSPTW_SOLVER.m is practically usable on a regular personal computer for up to 20 customers. For less than 13 customers, the calculation takes a few seconds. With a larger number of customers, the calculation can take tens of minutes or more. The program was successfully tested for up to 60 customers.

Entering too short and inappropriately combined time windows can easily make the problem unsolvable. It is therefore important to negotiate with customers and use time windows only when necessary.

Testing of the created program showed that it is highly usable in practice. Respecting the results obtained during the delivery of goods ultimately enables fuel savings.

The author intends to extend his future research to solving the vehicle routing problem with time windows (VRPTW) and respecting the predetermined priority of customers. In the next stage of the TSPTW and VRPTW solution, the use of artificial intelligence is expected.

Acknowledgement:
The submitted manuscript contains a specific model created by the author for solving TSPTW as an optimization problem of linear programming (4) with a specifically designed objective function (13) and constraints (5) to (12). Service times and time windows are respected.

The main and unique result of the article is the transformation of the created model into the Matlab environment, and the creation of the corresponding

M-function, which is applicable for solving TSPTW for any number of n customers. The created M-function TSPTW_SOLVER.m is in the Appendix.

References:

[1] Bentley, J. J.: Fast Algorithms for Geometric Traveling Salesman Problems. *ORSA Journal on Computing.* Vol. 4, No. 4, 1992, pp. 387–411.

[2] Davendra, D., *Traveling Salesman Problem, Theory and Applications.* InTech, Jerza Trdina 9, Rijeka, Croatia, 2010.

[3] Gutin, G., and Punnen, A. P., *The Traveling Salesman Problem and Its Variations.* New York: Springer Science+Business Media, LLC, 2007.

[4] Warren, R. H.: Generating Cyclic Permutations: Insights to the Traveling Salesman Problem. *WSEAS Transactions on Information Science and Applications,* Vol. 18, 2021, pp. 68–72, https://doi.org/10.37394/23209.2021.18.9.

[5] Zahradka, J.: The Traveling Salesman Problem Solution by Mixed Integer Linear Programming in Matlab Code. *Journal of Applied and Computational Sciences,* Vol. 1, No. 1, 2022, 45–52. https://doi.org/10.5281/zenodo.6880928.

[6] Bard, J. F., Kontoravdis, G., and Yu. G.: A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows. *Transportation Science,* Vol. 36, No. 2, 2002, pp. 250–269.

[7] Gradle, K. P., and Muley, Y. M.: Travelling Salesman Problem with MATLAB Programming. *International Journal of Advances in Ap-plied Mathematics and Mechanics.* Vol. 2, No. 3, 2015, pp. 258–266.

[8] Ohlmann, J. W., J. W., and Thomas, B. W.: A Compressed-Annealing Heuristic for the Traveling Sales-man Problem with Time Windows. *Informs Journal on Computing,* Vol. 19, No. 1, 2007, pp. 80–90. https://doi.org/10.1287/ijoc.1050.0145.

[9] Papalitsas, C., Andronikos, T., Ciannakis, K., Theocharopoulou, G., and Fanarioti, S.: A QU-BO Model for the Traveling Salesman Problem with Time Windows. *Algorithms,* Vol. 12, No. 11, 2019, pp. 224. https://doi.org/10.3390/a12110224.

[10] Sushma, M., Reddy, V.: Finding an Optimal Path with Hospital Information System Using GIS-based Network Analysis *WSEAS Transactions on Information Science and Applications,* Vol. 18, 2021, pp. 1–6. https://doi.org/10.37394/23209.2021.18.1.

[11] MathWorks. Inc., *Optimization Toolbox, User's Guide*. The MathWorks, Inc. Natic, Massachusetts, United States, 2024, [Online]. https://uk.mathworks.com/help/optim/index.html?s_tid=CRUX_lftnav (Accessed Date: Marcch 11, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The sole author of this scientific article independently conducted and prepared the entire work from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

This work was funded by the private sources of the author.

Conflict of Interest

The author has no conflict of interest to declare.

APPENDIX

TSPTW_SOLVER.m

```
function[X,CYCLE,AllDist,AWT,tStart,tArr]=...
    TSPTW_SOLVER(n,D,C,lw,uw,m)
Ctemp=C'; Ctemp=Ctemp(:); p=(n+1)*(n+1);
for i=1:p
   if Ctemp(i)==0
      Ctemp(i)=Inf;
   end
end
kf=min(Ctemp)/86400/n; Ctemp=[];
CT=C'; f=CT(:); f=[f;kf*ones(n,1)];
t0=min(lw-C(1,2:end));
A=zeros(n^2,(n+1)^2+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)...
         +m(i)+uw(i)-lw(j);
         A(k,p+i)=1; A(k,p+j)=-1;
         b(k,1)=uw(i)-lw(j)+m(i) -m(j);
      end
   end
end
for j=1:n
   k=k+1; A(k,(n+1)*i+1+j)=C(1,j+1)+t0-lw(j);
   A(k,p+j)=-1; b(k,1)=-lw(j)-m(j);
end
Aeq=zeros(3*n+3,p+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+1
   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
    if lw(i)<t0+C(1,1+i)
       lb(p+i,1)=t0+C(1,1+i)+m(i);
    else
       lb(p+i,1)=lw(i)+m(i);
    end
end
ub(p+1:p+n,1)=uw+m;
intcon = 1:p; options = optimoptions'intlin...
    prog','MaxTime',420,'MaxNodes',3000000);
X=intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,[],...
    options); X=round(X);
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
t=X(p+1:end);
tStart=(t(CYCLE(2))-m(CYCLE(2))-C(1,...
    CYCLE(2)+1))
tRet=t(CYCLE(end-1))+C(CYCLE(end-1)+1,1);
TotDur=tRet-tStart;
tDep=[tStart,t(CYCLE(2:end-1))];
tDep=hours(tDep); tDep.Format='hh:mm';
tArr=[t(CYCLE(2:end-1))-(m(CYCLE...
    (2:end-1))),tRet];
tArr=hours(tArr); tArr.Format='hh:mm';
uw=hours(uw(CYCLE(2:end-1)));
uw.Format='hh:mm';
lw=hours(lw(CYCLE(2:end-1)));
lw.Format='hh:mm';
TotDist=0;
for i=1:(n+1)
    for j=1:(n+1)
       if X((n+1)*(i-1)+j)==1
```

```
96:          TotDist=TotDist+D(i,j); break;
97:      end
98:    end
99: end
```