*МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2018*
*MATHEMATICS AND EDUCATION IN MATHEMATICS, 2018*
*Proceedings of the Forty-seventh Spring Conference*
*of the Union of Bulgarian Mathematicians*
*Borovets, April 2–6, 2018*

# THE TRAVELLING SALESMAN PROBLEM WITH THE SYMBOLIC COMPUTATION SYSTEM MATHEMATICA[*]

**Marin Marinov, Lasko Laskov**

This paper examines the classical problem of the travelling salesman. Using the symbolic computational system Mathematica we presenta heuristic method, an exhaustive search algorithm, and a branch and bound method. Using numerical experiments we illustrate different capabilities of the program implementations of the distinct methods.

**1. Introduction.** In this paper the following classical problem is examined: A travelling salesman must visit $n$ cities. The travel from an arbitrary selected city $i$ to a city $j$ has a price $f(i, j)$, where also $f(i, j) \neq f(j, i)$ is possible. The route of the travelling salesman starts from city 1 and ends in city 1, where each city must be visited exactly once. The total price of the route is the sum of all prices $f(i, j)$ of the individual roads that comprise it. The objective is to find the minimal price of all routes, and all routes with the minimal price.

Since the number of cities is finite, a minimum price route with the above requirements exists, and the problem is to find it among the number of possible routes.

The travelling salesman problem has a number of applications (see for example [6]). On the oter hand, from a computational point of view, it is an NP-hard problem [6], and the known algorithms that solve it have an exponential complexity. Something more, even the problem of finding an $\varepsilon$ sub-solution remains NP-hard.

The travelling salesman problem is a good test for many newly discovered methods. Furthermore, the development of many methods have been inspired by the attempts to find an efficient solution to this problem. The survey paper [6], and the monograph [2] give an extensive view of the scientific research on the topic up to 1989. The problem of the travelling salesman continues to be a subject of many scientific researches and university courses (see [1], [3], [7], [5]).

The goal of our paper is to show how symbolic computation system provide a way for clear and comprehensive implementation of the methods that solve the travelling salesman problem. The following type of methods are presented: heuristic methods, exhaustive methods, and a branch and bound method.

[*]**2010 Mathematics Subject Classification:** ???.
**Key words:** travelling salesman problem, branch and bound method, symbolic computation systems.

**2. Notations.** Let $A$ be an arbitrary matrix. We will use the following notations:

- $A(i,\, j)$ is the element of the matrix $A$ located on the $i$-th row and $j$th column;
- $A(i,\, \cdot)$ is the $i$th row of $A$;
- $A(\cdot,\, j)$ is the $j$th column of $A$.

We denote by $v_0$ the set $\{2,\, 3,\, \ldots,\, n\}$. We record all permutations of $v_0$ in the matrix $P$, where its $i$-th row:

$$P(i,\, \cdot) = \left\{i_1,\, i_2,\, ,\, i_{(n-1)}\right\} \tag{1}$$

is the $i$-th permutation of $v_0$. Besides that, for each $i \in \{1,\, 2,\, \ldots,\, (n-1)!\}$ we define

$$\overline{P(i,\, \cdot)} = \left\{1,\, i_1,\, i_2,\, \ldots,\, i_{(n-1)},\, 1\right\}. \tag{2}$$

It is clear that each permutation $P(i,\, \cdot)$ defines a single possible route $\overline{P(i,\, \cdot)}$,:

$$1 \to i_1 \to i_2 \to i_3 \to \cdots \to i_{(n-1)} \to 1 \tag{3}$$

and vice versa. This allows us to regard

$$W = \{\overline{P(i,\, \cdot)}\ :\ i \in \{1,\, 2,\, \ldots,\, (n-1)!\} \tag{4}$$

as the set of all possible routes.

We will call the matrix $M$ the *cost matrix* in the travelling salesman problem defined above, if

$$M(i,\, j) = \begin{cases} f(i,\, j), & \text{if} \quad i \neq j \\ B, & \text{if} \quad i = j, \end{cases} \tag{5}$$

where $B$ is a *big* number.

The cost matrix $M$ defines the function $Ff(w)$, which relates to each route $w$ from $W$ its price

$$Ff(w) = \sum_{i=1}^{n} M\left(w_i,\, w_{(i+1)}\right), \tag{6}$$

where $w = \{w_1,\, w_2,\, \ldots,\, w_{(n+1)}\}$.

Now the travelling salesman problem can be defined in the following way:

**Problem 1.** *The travelling salesman problem.* Find:

$$r_0 = \min_{w \in W} Ff(w) \tag{7}$$

and give a list $v$ of all routes with price $r_0$.

For clarity we will use the following example in which the cities are 6 and the cost matrix is

$$M = \begin{pmatrix} 3000 & 26 & 42 & 15 & 29 & 25 \\ 7 & 3000 & 16 & 1 & 30 & 25 \\ 20 & 13 & 3000 & 35 & 5 & 1 \\ 21 & 16 & 25 & 3000 & 18 & 18 \\ 12 & 46 & 27 & 48 & 3000 & 5 \\ 23 & 5 & 5 & 9 & 5 & 3000 \end{pmatrix} \tag{8}$$

The big number $B = 3000$ is placed on the main diagonal of $M$.

**3. Heuristic methods.** These are rules without a rigorous proof and are based on likely comprehensions that partially solve the examined task. Usually they do not lead to an exact solution, but they are fast and in many cases are a good starting point of the research.

A typical example are the so-called *Greedy algorithms.* In the case of the examined problem, the idea can be formulated in the following way: *Being in the city $i$, we choose among the cities not yet visited the city $j$, for which the price $f(i, j)$ is minimal.*

We implement this idea, starting from the city 1:

```
B0 = 3000; M[[All, 1]] = M[[All, 1]] + B0;
For[i=1, i < d0[[1]], i++, j = 1;
     While[M[[z[i], j]] > Min[M[[z[i]]]], j++];
     z[i + 1] = j;
     M[[All, z[i + 1]]] = M[[All, z[i + 1]]] + B0;
     v = Join[v, {z[i + 1]}]];
v = Join[v, {1}]
```

Listing 1: On each step, at city $i$, the greedy algorithm selects the city $j$ with minimal price $f(i, j)$

For the examined case of the matrix $M$ we get the following route:

$$1 \to 4 \to 2 \to 3 \to 6 \to 5 \to 1, \tag{9}$$

that has price 65. After finding the exact solution we will see that this result is not *bad.* Something more, we can try to improve the described algorithm by initiating the tour from a special starting point. For example:

1. Find the minimum element $M(i_0, j_0)$ of the cost matrix $M$.
2. Select the transition $i_0 \to j_0$.
3. Complement the built section of the route with transition of type $k \to i_0$, or transition of type $j_0 \to p$ depending on where the minimum is reached.

Let us examine Problem 1 in the case of 4 cities, and a cost matrix

$$M = \begin{pmatrix} 100 & 1 & 4 & 4 \\ 4 & 100 & 2 & 4 \\ 4 & 4 & 100 & 3 \\ x & 5 & 5 & 100 \end{pmatrix}, \tag{10}$$

where $x \gg 5$. It turns out that the solution of the Greedy algorithm has a price $x + 6$, while the minimal price is 14. This example shows that the difference between the result of the Greedy algorithm and the exact solution can be an arbitrarily big number.

Heuristic methods are built, for which the deviation from the optimal solution is not bigger than the optimal solution itself, provided that for the elements of the matrix cost matrix $M$ the following conditions are fulfilled:

(i) Symmetry

$$f(i, j) = f(j, i), \ \forall \{i, j\} \subseteq \{1, 2, \dots, n\}. \tag{11}$$

(ii) Triangle inequality

$$f(i, j) \leq f(i, k) + f(k, j), \ \forall \{i, j, k\} \subseteq \{1, 2, \dots, n\}. \tag{12}$$

We will mention two such methods:

**Remark 1.** Let the conditions (11) and (12) are fulfilled. By modification of Listing 1, we implement the following algorithm:

1. $w = \{1\}$ and $V = \{1, 2, \ldots, n\}$.
2. If $V \backslash w = \emptyset$, on the last position of $w$ put 1; end.
3. if $V \backslash w \neq \emptyset$, find $i_0 \in w$ and $j_0 \in V \backslash w$, for which $f(i_0, j_0) \leq f(i, j), \forall i \in w$ and $\forall j \in V \backslash w$.
4. Include $j_0$ in $w$, and place it after $i_0$.
5. Go to 2.

**Remark 2.** Another algorithm based on the above conditions can be described by definition of the Problem 1 in the terms of the graph theory. Each city $j$ is a vertex $j$ in the graph, while each path from city $i$ to city $j$ is an edge $\{i, j\}$ with cost $f(i, j)$. By condition for each $i \in \{1, 2, \ldots, n\}$ and each $j \in \{1, 2, \ldots, n\}$ an edge $\{i, j\}$ with cost $f(i, j)$ is defined. We assume that the conditions (11) and (12) are fulfilled. Hence, a complete, undirected graph $G$ is defined. The permissible routes of the travelling salesman are the Hamiltonian cycles of the so defined graph $G$. Then the Problem 1 is brought to finding the Hamiltonian cycle with minimal length $r_0$. The algorithm can be described in the following way:

1. Given a graph $G$, build the function $F(G)$ that finds its minimum spanning tree $T$.
2. By doubling of each edge of $T$ get a graph $TT$, whose vertices have even powers.
3. Given a graph with even powers of its vertices $G$, build a function $E(G)$, that finds an Oiler cycle $E(G)$.
4. Separate a Hamiltonian cycle, that is contained in $E(G)$.

The algorithms in Remark 1 and Remark 2 have complexity $O(n^2)$. Besides that, if we denote $w_0$ a route built by means of these two algorithms, the following inequality holds: $r_0 \leq Ff(w_0) \leq 2r_0$.

**4. Exhaustive search methods.** The first obvious approach to solve Problem 1 is by an exhaustive search of all possible routes. This method has a simple structure:

1. Form a list of all possible routes.
2. Calculate the price of each route.
3. Select a list of all optimal routes (routes that have minimal price).

We will illustrate this method by implementing the algorithm in the Listing 2.

In the implementation of this algorithm we will use the build-in function `Permutations[v`$_0$`]` that calculates the matrix $P$ of the permutations (see (1)). The generation of all permutations is a classical example given in the courses of computer programming, and can be easily implemented if needed.

```
v0 = {2, 3, 4, 5, 6};
P = Permutations[v0];
d = Length[v0]!;
v = {Join[{1}, P[[1]], {1}]};
r0 = Ff[v[[1]]];
For[i = 2,i < d + 1, i++, w = Join[{1}, P[[i]], {1}];
     If[Ff[w] < r0,r0 = Ff[w];
     v = {w}, If[Ff[w] == r0, v = Join[v, {w}]]]]
```

Listing 2: Exhaustive search solution of Problem 1

For the examined case of $n = 6$ and a cost matrix $M$ we obtain a minimal cost $r_0 = 62$, and a single optimal route that attains it:

(13) $$1 \to 4 \to 3 \to 5 \to 6 \to 2 \to 1.$$

In this solution we have a matrix with $(n-1)!$ rows and $2(n-1)!$ number of tests, where $n$ is the number of cities. This makes the solution quite lavish. Its good feature is that it is clear and effective for a *small* number of cities. On the other hand, for *big* values of $n$ this method is useless.

Let us clarify the above with a numerical experiment. For this purpose we define two matrices. The matrix

$$M_1 = (M_1(1,\,\cdot),\, M_1(2,\,\cdot),\, M_1(3,\,\cdot),\, \ldots,\, M_1(15,\,\cdot))\,, \tag{14}$$

where the rows $M_1(j,\,\cdot)$ are:

$M_1(1,\,\cdot) = (3000, 26, 42, 15, 29, 25, 35, 23, 19, 25, 25, 25, 15, 18, 25)$,
$M_1(2,\,\cdot) = (7, 3000, 16, 1, 30, 25, 7, 11, 21, 20, 12, 11, 10, 11, 20)$,
$M_1(3,\,\cdot) = (20, 13, 3000, 35, 5, 1, 26, 6, 16, 15, 15, 8, 13, 22, 15)$,
$M_1(4,\,\cdot) = (21, 16, 25, 3000, 18, 18, 6, 46, 25, 23, 26, 5, 31, 9, 23)$,
$M_1(5,\,\cdot) = (12, 46, 27, 48, 3000, 5, 67, 13, 23, 27, 14, 35, 21, 32, 27)$,
$M_1(6,\,\cdot) = (23, 5, 5, 9, 5, 3000, 32, 42, 32, 19, 22, 23, 19, 18, 19)$,
$M_1(7,\,\cdot) = (35, 7, 26, 6, 67, 32, 3000, 11, 15, 21, 21, 12, 16, 17, 21)$,
$M_1(8,\,\cdot) = (5, 23, 65, 10, 16, 67, 57, 3000, 32, 18, 18, 21, 27, 23, 18)$,
$M_1(9,\,\cdot) = (8, 11, 25, 35, 11, 21, 17, 21, 3000, 24, 22, 20, 23, 35, 24)$,
$M_1(10,\,\cdot) = (11, 25, 11, 21, 17, 21, 24, 22, 24, 3000, 21, 17, 19, 30, 19)$,
$M_1(11,\,\cdot) = (25, 9, 12, 15, 26, 14, 22, 21, 18, 21, 3000, 26, 21, 16, 21)$,
$M_1(12,\,\cdot) = (26, 42, 15, 29, 25, 35, 23, 19, 25, 25, 25, 3000, 31, 13, 16)$,
$M_1(13,\,\cdot) = (26, 6, 16, 15, 15, 8, 13, 22, 15, 25, 25, 22, 3000, 26, 25)$,
$M_1(14,\,\cdot) = (6, 46, 25, 23, 26, 5, 31, 9, 23, 20, 12, 32, 32, 3000, 23)$,
$M_1(15,\,\cdot) = (67, 13, 23, 27, 14, 35, 21, 32, 27, 15, 15, 25, 25, 16, 3000)$.

The second matrix is $M_2$, which is obtained from the matrix $M_1$ after removing the first two rows and first two columns.

We solve the problem with matrix $M_2$ using the algorithm in Listing 2 and we receive that the minimal price is 146 and there is one optimal route:

$$1 \to 6 \to 13 \to 8 \to 11 \to 5 \to 2 \to 10 \to 12 \to 9 \to 7 \to 3 \to 4 \to 1 \tag{15}$$

The solution was received after more than 4 hours execution on a standard PC configuration.

The attempt to solve the problem with the algorithm in Listing 2 with the matrix $M_1$ on the same PC configuration, shows that this is not possible.

The attempts to improve the complexity of this algorithm have a partial success, and more or less place it among the hybrid approaches.

## 5. Branch and bound method.

When solving the travelling salesman problem with the *branch and bound method* it is not needed to generate the set of all possible routes $W$ (see (4)) and to traverse all the routes. This method connects two processes: branching (separation of non-intersecting subsets of $W$), and calculation of an upper bound $R$ and a lower bound $u$, for which $u \le r_0 \le R$.

### 5.1. General description.

**Bounds calculation.** It is accepted to call the upper bound *current record* and to improve it during the process of problem solving. The calculation of bounds is formed by the following stages:

1. Set $R = Ff(w)$, where $w$ is an arbitrary selected element in $W$.
2. If during the process of solving a route $w_0$ is found, for which $Ff(w_0) < R$, then the current record is changed $R = Ff(w_0)$.

We will call the route that has a price equal to the current record an *approximate solution.*

To calculate the lower bound $u$ we apply the following procedure. The price $Ff(w)$ of each route $w = \{w_1, w_2, \dots, w_{(n+1)}\}$ is the sum of the following elements of the matrix $M$ (see equation (6):

$$\left\{M\left(w_1, w_2\right), \; M\left(w_2, w_3\right), \; \dots, \; M\left(w_n, w_{(n+1)}\right)\right\} \tag{16}$$

It follows that each row and each column of $M$ contains a single element of the set (16) from the definition of the route $w$. This allows easily to to prove the following rule for calculation of $u$:

1. Find the number $m_i^{(r)}$, that is the minimum of the elements in the $i$-th row of $M$, and the number $m_j^{(c)}$, that is the minimum of the $j$-th column of $M$.
2. Define: (a) the *row reduced* matrix $M_r$, whose $i$-th row is obtained from the $i$-th of $M$ by subtracting from each element the number $m_i^{(r)}$, and (b) the *column reduced* matrix matrix $M_c$, whose $j$-th column is obtained from the $j$-th column of $M$ by subtracting from each element the number $m_j^{(c)}$.
3. Calculate
$$u_1 = \sum_{i=1}^{n} \left(m_i^{(r)} + m_i'\right), \tag{17}$$
where $m_i'$ is the minimum element from the $i$-th column of$M_r$.
4. Calculate
$$u_2 = \sum_{j=1}^{n} \left(m_j^{(c)} + m_j''\right), \tag{18}$$
where $m_j''$ is the minimum element from the $j$-th row of$M_c$.
5. $u = \max\{u_1, \; u_2\}$.

It is clear that the *accuracy* with which the approximated solution approximates $r_0$ is qqual to $R - u$.

We will note that we can obtain from each matrix $M$ the following two matrices $M_1$ and $M_2$:

- $M_1$ is obtained when we reduce $M_r$ by columns;
- $M_2$ is obtained when we reduce $M_c$ by rows.

In future we will call the matrix $M_1$ *reduced matrix of* $M$, when $u_1 \geq u_2$. In the case when $u_1 < u_2$, we will call the matrix $M_2$ *reduced matrix of* $M$.

For convenience, we will define the function $F(M)$, such as for each matrix $M$ it will calculate the lower bound $u$ and the reduced matrix $M^{(r)}$.

**Branch (separation of non-intersecting subsets of $W$).** We will implement the branch together with the search for a better approximation of the solution. More precisely, using an heuristic algorithm we define a candidate for the approximate solution $w_0$ and we define a system of non-intersecting subsets $A = \{W_j\}$ of $W$, for which the following statement is true: *If there exists $x \in W$, for which $Ff(x) \leq R$, then*

$$x \in \{\cup W_j\} \qquad \text{or} \qquad x = w_0. \tag{19}$$

**Algorithm for $W$ branch.** We set $R = Ff(\overline{w})$, where $\overline{w}$ is an arbitrarily chosen element of $W$. We define the list of chosen solutions $v = \{\overline{w}\}$.

**Stage 1.**

1.1. Find the reduced matrix $M^{(r)}$ and the lower bound $ux$.

1.2. Cut test $ux > R$. The inequality is not fulfilled, so continue with 1.3.

1.3. If $M^{(r)}(s,\, p) = 0$ is a zero element of $M^{(r)}$, define its weight $t_{sp}$ as follows:

$$t_{sp} = \min\left\{M^{(r)}(s,\, j) :\ j \neq p\right\} + \min\left\{M^{(r)}(i,\, p) :\ i \neq s\right\} \tag{20}$$

1.4. Choose transition $i_1 \to j_1$ for which $t_{i_1 j_1} = \max\left\{t_{ij} :\ M^{(r)}(i,\, j) = 0\right\}$.

1.5. Divide the set $W$ into two non-intersecting subsets $W_1$ and $X_1$, where $W_1$ contains all routes of $W$, that do not contain the tour $i_1 \to j_1$. Such a branch is called *dichotomy.*

1.6. We store in $wtx$ the information for the built connected parts of $w_0$. Initially $wtx = \{\, \{i_1,\, j_1\}\, \}$.

1.7. Create the matrix $M_1$ that defines the routes complementing the chosen transition $i_1 \to j_1$ to a route from $X_1$:
   (a) Put $M^{(r)}(j_1,\, i_1)$ to be equal to a boundary big number $B$.
   (b) Remove the $i_1$-th row and $j_1$-th column of $M^{(r)}$.

**Stage 2.** Repeat the considerations from the *Stage 1*, after replacing the matrix $M$ by the matrix $M_1$.

2.1. Find the reduced matrix $M_1^{(r)}$ and the correction of the lower bound $\alpha$. Define $ux := ux + \alpha$.

2.2. *Cut down.* If $ux > R$, then each route in $X_1$ will have a price bigger than the current record, so do not examine it. Continue by examination of $W_1$. If $ux \leq R$, then continue with the next step 2.3.

2.3. Define the weights $t_{sp}^{(1)}$ using (20), in other words
$t_{sp}^{(1)} = \min\left\{M_1^{(r)}(s,\, j) :\ j \neq p\right\} + \min\left\{M_1^{(r)}(i,\, p) :\ i \neq s\right\}$.

2.4. Choose the tour $i_2 \to j_2$, defined by $\max\left\{t_{ij}^{(1)} :\ M_1^{(r)}(i,\, j) = 0\right\}$.

2.5. Divide the set $X_1$ into two non-intersecting subsets $W_2$ and $X_2$, where $W_2$ contains all the routes of $X_1$ that include the transition $i_2 \to j_2$.

2.6. Store in $wtx$ the information for the built connected parts of $w_0$:
   (i) If $j_2 = i_1$, then $wtx = \{\, \{i_2,\, j_1\}\, \}$ and $M^{(r)}(j_1,\, i_2) = B$.
   (ii) If $i_2 = j_1$, then $wtx = \{\, \{i_1,\, j_2\}\, \}$ and $M^{(r)}(j_2,\, i_1) = B$.
   (iii) If $j_2 \neq i_1$ and $i_2 \neq j_1$, then $wtx = \{\, \{i_1,\, j_1\}\, \{i_2,\, j_2\}\, \}$ and $M^{(r)}(j_2,\, i_2) = B$.

2.7. Define the matrix $M_2$, by removing the row $i_2$ and the column $j_2$ of the resulting matrix $M_1^{(r)}$ from step 2.4. The matrix $M_2$ defines the routes that complement the chosen transitions $i_1 \to j_1$ and $i_2 \to j_2$ to a route to $X_2$.

**Stage 3.** We repeat the considerations from the *Stage 2*, after replacing the matrix $M_1$ by the matrix $M_2$, and after replacing the set $X_1$ by the set $X_2$. As a result we choose the transition $i_3 \to j_3$; we define the matrix $M_3$; we define the sets $W_3$ and $X_3$.

The set $W_3$ contains the routes of $W$ that contains transitions $i_1 \to j_1$ and $i_2 \to j_2$, but does not contain the transition $i_3 \to j_3$. The set $X_3$ contains these routes of $W$, that include the transitions $i_1 \to j_1$, $i_2 \to j_2$ and $i_3 \to j_3$.

If the procedure is not interrupted because of the cut down, then after the $(n-2)$-th stage the system of subsets $A = \{W_j\}$, we were looking for, is received, as well as $(n-2)$ of the transitions of $w_0$.

**Stage $(n-1)$.** Using a direct verification, we finalize the construction of $w_0$.

**Stage $n$.** If $Ff(w_0) < R$, then we put $R = Ff(w_0)$ and $w_0$ to be the new approximated solution, in other words $v = \{w_0\}$.

If $Ff(w_0) = R$, then we fill in the list of approximate solutions $v := v \bigcup \{w_0\}$.

***5.2. General structure of the branch and bound algorithm.***

1. Form a list $A = \{\, W \,\}$.
2. If $A = \emptyset$, then the problem is solved. If $A \neq \emptyset$, go to step 3.
3. Choose an element $X_1$ from $A$ and remove it from $A$.
4. Complete a branch of $X_1$:
   (i) Define the non-intersecting subsets $W_j$, and store them in $A$.
   (ii) Update the current record $R$ and the list of the approximated solutions $v$.
5. Go back to 2.

**Remark 3.** The subsets of $W$, that are included in the list $A$, are stored with their characteristics:

- matrix $M_x$;
- initial value for the lower bound $ux$;
- vector $rx$ with initial numbering of the rows of the matrix $M_x$;
- vector $rx$ with initial numbering of the columns of the matrix $M_x$;
- vector $wc$ that contains the transitions already chosen from $w_0$;
- vector $wtx$ that contains the information for the parts of $w_0$ already built.

**Remark 4.** For a clearer structure of the program, initially we define four functions:

(1) Function $F(X)$, that calculates for each matrix $X$ its reduced matrix $X^{(r)}$ and its lower bound $ux$.

(2) Function $G(X)$, that calculates the weights of the zeroes of $X$. For each matrix $X$ it calculates: the matrix $v$ of the coordinates of the zeroes of $X$, a vector $t$ of the weights of the zeroes, and the number of zeroes.

(3) Function $H(X)$, that determines together with $G(X)$ the zeroes of $X$ with maximum weight.

(4) Function $S(\{i, j\})$ that is determined using $rx, cx, wtx$. For a chosen transition $i \to j$, the function $S(\{i, j\})$ calculates the new coordinates of the barrier (it prohibits the internal contours) and renews $wtx$.

**Remark 5.** The implementation of the *Branch and Bound* method for solving Problem 1 is capable of finding not more than initially selected number of optimal routes.

**Remark 6.** The described program will find a single route, if the condition for cutting down is $ux \geq R$. In Stage 2, Step 2.2 the strict inequality $ux > R$ must be substituted by the non-strict equality $ux \geq R$.

The structure of the program is given in the block diagram in Fig. 1. It is obvious that adopting this approach allows the reduction of the tests performed by the program.

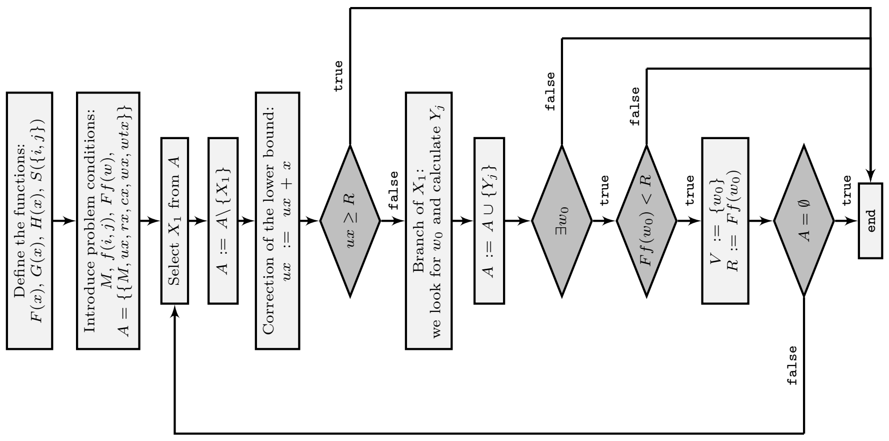
Define the functions: $F(x)$, $G(x)$, $H(x)$, $S(\{i,j\})$
Introduce problem conditions: $M$, $f(i,j)$, $Ff(w)$, $A = \{\{M, ux, rx, cx, wx, wtx\}\}$
Select $X_1$ from $A$
$A := A \backslash \{X_1\}$
Correction of the lower bound: $ux := ux + x$
$ux \geq R$
true
false
Branch of $X_1$: we look for $w_0$ and calculate $Y_j$
$A := A \cup \{Y_j\}$
$\exists w_0$
false
true
$Ff(w_0) < R$
false
true
$V := \{w_0\}$ $R := Ff(w_0)$
$A = \emptyset$
true
false
end

Fig. 1. Block diagram of the Branch and Bound method implemented for the Problem 1

***5.3. Validation and example.*** The validation of the program is performed by solving the following problems.

Problem 1 is solved for the matrix $M$ defined by equation (8). The result of the exhaustive search method is confirmed.

Problem 1 is solved for the matrix $M_2$, that has been defined as a sub-matrix of the matrix $M_1$ (see equation (14)). The result of the exhaustive search method is confirmed. The huge difference is in the execution time of the program – the *Branch and bound* method completes for less than a second.

Problem 1 is solved for the matrix $C_A$ from [5] p. 289. The result from p. 294 is obtained.

Let us examine the following example. We solve Problem 1 for the matrix $M_1$, that is defined with equation (14)). The program implementing the *Branch and bound* method solved the problem for less than a second. We obtain minimal price 151, and two optimal routes:

$$1 \to 13 \to 9 \to 2 \to 7 \to 4 \to 12 \to 15 \to 10 \to 3 \to 6 \to 5 \to 11 \to 14 \to 8 \to 1$$

$$1 \to 13 \to 2 \to 7 \to 4 \to 12 \to 14 \to 8 \to 15 \to 10 \to 3 \to 6 \to 5 \to 11 \to 9 \to 1$$

## 6. Conclusions.

***6.1. Connection between the travelling salesman problem and the assignment problem.*** Together with Problem 1 we will examine the *assignment problem*:

**Problem 2.** The *assignment problem* has the same cost matrix $M$ (see equation (5)). Each assignment can be represented with a permutation $v = \{i_1, i_2, \ldots, i_n\}$ of the elements $\{1, 2, \ldots, n\}$. The permutation $v$ shows that the $k$-th candidate is assigned on the $i_k$-th position. The cost of this assignment is

$$F_1(v) = \sum_{k=1}^{n} M(k, i_k)\,. \tag{21}$$

In the assignment problem the minimal price is searched:

$$p_0 = \min_{v \in V} F_1(v), \tag{22}$$

where $V$ is the set of all assignments.

We note that any possible route $1 \to w_2 \to w_3 \to \cdots \to w_n \to 1$ in Problem 1 defines a single assignment in $v$ in which candidate 1 is assigned on position $w_2$, candidate $w_2$ is assigned on position $w_3$, candidate $w_3$ is assigned on position $w_4$, etc. Besides that, $Ff(w) = F_1(v)$. Therefore,

$$p_0 \leq r_0, \tag{23}$$

because $\{Ff(w) \,:\, w \in W\} \subset \{F_1(v) \,:\, v \in V\}$.

The above shows that the *Branch and bound* method can be used to solve Problem 2, and gives a better lower bound of branching. In this case it is better to modify the implementation of the method, because:

- if the optimal assignment has a single cycle, then $p_0 = r_0$, and this gives directly the solution of the Problem 1;
- if the optimal assignment has $k > 1$ cycles, then $r_0 - p_0$ can be an arbitrarily big number which can be shown easily with an example. In this case we select the cycle, that has the fewest number of edges and we perform branching on each separate edge of the selected cycle.

***6.2. Integer programming.*** The symbolic computation system Mathematica has a built-in function `LinearProgramming[c, m, b]` that solves the *linear programming problem.* We will bring the Problem 1 to a problem that can be solved with the function `LinearProgramming[c, m, b]`.

We represent each possible route with an $n \times n$ matrix $X$ with elements:

$$x_{ij} = \begin{cases} 1, & \text{if the route contains the transition } i \to j \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

Hence, the elements of $X$ fulfill the condition:

$$x_{ij} \in \{0,\ 1\}, \ \ \forall i \in \{1,\, 2, \ldots, n\} \ \text{ and } \ \forall j \in \{1,\, 2, \ldots, n\}. \tag{25}$$

The price of the admissible route is $F_2(X) = \displaystyle\sum_{i=1}^{n} \sum_{j=1}^{n} M\,(i,\, j)\, x_{ij}$.

The condition that the travelling salesman must visit each city only once imposes the following restrictions on the matrix $X$:

$$\sum_{i=1}^{n} x_{ij} = 1 \tag{26}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \tag{27}$$

The requirement that $X$ must define a single cycle imposes the following restrictions on the elements of the matrix $X$:

$$u_i - u_j + n x_{ij} \leq n - 1; \ \ u_i \geq 0, \ \ i, j \subset \{1,\, 2, \ldots, n\} \ \text{ and } \ i \neq j. \tag{28}$$

Using the above, Problem 1 is brought to the problem of finding the minimum of the function $F_2(X)$ with conditions (25), (26), (27), and (28), which can be solved using the function `LinearProgramming[c, m, b]`. Certain inconveniences are caused by the conditions (28). Even for small $n$ the number of conditions is *big* enough, because it is equal to $n(n-1)$. In particular, in the case of matrix $M_1$ we will have 210 conditions.

We will point out that the problem for finding the minimum of the function $F_2(X)$ brings the solution of the Problem 2. This allows us to use the above algorithm to solve it.

The integrated environment of the symbolic computation system gives us the opportunity for comfortable implementation of the algorithms that solve Problem 1. This allows us to solve many generalizations and applications of the travelling salesman problem.

## REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Introduction to Algorithms, Third Edition. The MIT Press, 3rd edition, 2009.

[2] E. L. Lawler. The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley-Interscience series in discrete mathematics and optimization, John Wiley & sons, 1985.

[3] A. Schrijver. A Course in Combinatorial Optimization, Course notes. University of Amsterdam and CWI Amsterdam, 2017.

[4] Х. Кристофидес. Теория графов. Алгоритмический подход. Мир, 1978.

[5] Кр. Манев. Алгоритми в графи. Основни алгоритми. КЛМН, София, 2013.

[6] И. И. Меламед, С. И. Сергеев, И. Х. Сигал. Задача коммивояжера. Вопросы теории. Автоматика и телемеханика, **9** (1989), 3–33.

[7] И. Х. Сигал, А. П. Иванова. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. ФИЗМАТЛИТ, 2002.

Marin Laskov Marinov
e-mail: mlmarinov@nbu.bg
Lasko Marinov Laskov
e-mail: llaskov@nbu.bg
New Bulgarian University
Informatics Department
21 Montevideo Str.
1618 Sofia, Bulgaria

## РЕШАВАНЕ НА ЗАДАЧАТА НА ТЪРГОВСКИЯ ПЪТНИК СЪС СРЕДСТВАТА НА СИСТЕМАТА MATHEMATICA

**Марин Маринов, Ласко Ласков**

В статията се разглежда класическата задача за търговския пътник. Със средствата на системата Mathematica са представени евристичен метод, методът на пълното изчерпване и методът *разклоняване и граници.* Чрез числени експерименти се илюстрират различните възможности на програмните реализации на отделните методи.