AN EFFECTIVE EXACT ALGORITHM FOR ONE PARTICULAR CASE OF THE TRAVELING-SALESMAN PROBLEM
Panishev A.V., Plechysty D.D.

Abstract: The article presents the exact algorithm for solving one case of the job-scheduling problem for the case when the source matrix is ordered by rows.

Keywords: traveling-salesman problem, optimization, optimal path, permutations.

Introduction
In this paper the task is reviewed, for which the space of admissible solutions is the subset of set $P$ of all permutations $\pi = (\pi[1], ..., \pi[n])$ of $n$ objects, numbered $1, 2, ..., n$. It is known, that functional $C(\pi)$, defined on $P$, reaches it's optimum in class $P_1 \subset P_1$, where $P_1$ is set of all pyramidal permutations $p = (p[1], ..., p[n])$.

It should be noted that pyramidal permutation of $n$ elements of set $\{1, 2, ..., n\}$ produces two pyramidal permutations of $n+1$ elements of set $\{1, 2, ..., n, n+1\}$ as a result of inserting the element $n+1$ directly before and after the element $n$. In such way all permutations of set $P_1$ can be produced for any $n$.

Obviously, the pyramidal permutation $p = (p[1], ..., p[n])$ is cyclic, and $P_1 \subset P_1$, where $P_1$ is the set of all cyclic permutations of $n$ elements of set $\{1, 2, ..., n\}$, $|P_1| = (n-1)!$. In a pyramidal permutation $p = (p[1], ..., p[l-1], p[l], p[l+1], ..., p[n])$, where $p[l] = n$, one of elements $p[l-1], p[l+1]$ is the element $(n-1)$ of set $\{1, 2, ..., n\}$.

Let's look at one of the tasks of minimization of functional, defined on the set $P_1$ of all cyclic permutations $\pi = (\pi[1], ..., \pi[n])$, and in particular, at the case, where the optimal solution belongs to class $P_1$. The reviewed task is one of the fundamental tasks of the scheduling theory and is defined as follows.

Let's consider the conveyor system consisting of $m$ machines job $i, i = 1, m$, is processed without interruptions. Let $t_{ij}$ be the processing time of stage $j$ of job $i, j = 1, m$. Let's assume that if $t_{ij} = 0$, then job $i$ is processed by machine $j$, but the duration of stay at this machine is negligibly small. In the conveyor system processing $n$ jobs schedule $\pi$ is calculated from matrix $[t_{ij}]_{n \times n}$ and is defined by matrix of jobs' starting moments $[\tau_{ij}]_{n \times n}$ or by matrix of jobs' finishing moments $[\tau_{ij}]_{n \times n}$. Considering these formulas, the moment of finishing all jobs of conveyor schedule $\pi$ equals

$$\tilde{t}_{\text{max}}(\pi) = \max \{\tilde{t}_{ij} : i = 1, m, j = 1, m\}.$$ (1)

The goal is to create an optimal schedule $\pi^*$, which delivers the minimum of $\tilde{t}_{\text{max}}(\pi)$.

It is known that for the described above problem the schedule of minimum duration $\pi^*$ is located in the class of permutational schedules $\pi = (i_1, i_2, ..., i_n)$, where $i_k$ is the index of $k$-th from the left job in sequence $\pi$ [1].

Conversion between job-scheduling and traveling-salesman problems
In article [1] the task of finding permutation $\pi^* = (i_1, i_2, ..., i_n)$ of $n$ uninterruptedly-processed jobs, delivering the minimum of $\tilde{t}_{\text{max}}(\pi)$, is formulated as the traveling-salesman problem (TSP) with $n+1$ cities, numbered $0, 1, ..., n$, and the matrix of costs $[c_{pq}]_{n+1}$, where $c_{pq} = \infty, c_{pq} \geq 0$, $p = 0, 1, ..., n$, $q = 0, 1, ..., n$, $p \neq q$. For such variant of TSP the task is to find permutation $\pi^* = (0, i_1, i_2, ..., i_n)$, delivering the minimum of the following functional on the set of all $(n-1)!$ permutations $\pi_0 = (0, i_1, i_2, ..., i_n)$:
\[
C(\pi_0) = c_{00} + \sum_{i=1}^{n-1} C_{i_i} + C_{0n} ,
\]
where \( c_{pq} = 0 \), \( c_{p0} = \sum_{j=1}^{m} t_{pj} \), \( c_{pq} = \max \left( \sum_{j=1}^{i} t_{pj} - \sum_{j=1}^{i-1} t_{pj} \right) \), \( p=1, n \), \( q=1, n \), \( p \neq q \), \( c_{00} = c_{pp} = \infty \).

The solution of TSP in its given formulation gives optimal schedule \( \pi = (i_1, \ldots, i_n) \) for \( n \) uninterruptedly-processed jobs in the conveyor system with \( m \) machines. In this article we present an algorithm which finds exact solution for the particular case of minimization problem (1) in polynomial time.

If in the source matrix \( \left[ t_{ij} \right]_{m \times n} \) of problem (1), where \( m > 2 \), it is true for any pair of jobs \( i \) and \( k \) that either \( t_{ij} > t_{kj} \), or \( t_{ij} \leq t_{kj} \), for all \( j = 1, m \), then matrix \( \left[ t_{ij} \right]_{m \times n} \) is said to be ordered by rows.

Let's assume that in the matrix ordered by rows \( t_{ij} \leq t_{kj} \) and \( i < k \), \( j = 1, m \). If for any pair of jobs \( i \) and \( k \) in the source matrix ordered by rows it is true that \( t_{ij} \geq t_{kj} \), \( i < k \), \( j = 1, m \), then we should permute rows in the non-decreasing order of values \( t_{ij} \), \( i = 1, n \), \( j = 1, m \). It is known from article [1] that the optimal solution of problem (1) for the case, when matrix \( \left[ t_{ij} \right]_{m \times n} \) is ordered by rows, is located in the class of pyramidal permutations \( p = (p[1], \ldots, p[n]) \), and such solution can be achieved in polynomial time. We shall describe the procedure, which is faster than the algorithm presented in [1].

Using correlations (2a) we'll transform the matrix \( \left[ t_{ij} \right]_{m \times n} \) ordered by rows into matrix of costs \( \left[ c_{pj} \right]_{m \times n} \), \( p, q = 0, n \). The sought for solution of the particular case of problem (1) is a sequence \( p^* = (0, p^*[1], \ldots, p^*[n]) = (0, p^*) \), where \( p^* = (p^*[1], \ldots, p^*[n]) \) is a permutation of columns of matrix \( \left[ c_{pj} \right]_{m \times n} \) with indexes \( 0, \ldots, n \), \( p^* \in P_r \).

Let's use the known properties of matrix \( \left[ c_{pj} \right]_{m \times n} \), achieved from transformation of matrix \( \left[ t_{ij} \right]_{m \times n} \) ordered by rows:

a) \( c_{ik} \geq c_{ik}, n \geq r \geq i \geq 1, r \neq k, i \neq k, k = 1, n \);

b) \( c_{ik} \geq c_{ij}, n \geq j \geq k \geq 1, i \neq k, i \neq j, j = 1, n \);

c) \( c_{ik} - c_{ij} \geq c_{ik} - c_{ij}, n \geq j > k \geq 1, n \geq r > i \geq 1, r \neq k, i \neq k, r \neq j, i \neq j \).

Similarly to the way in which inequality c) was proven [1], we can show that elements of the matrix \( \left[ c_{pj} \right]_{m \times n} \) are connected by inequality

d) \( c_{ik} - c_{ij} \geq c_{ik} - c_{ij}, n \geq j > k \geq 0, n \geq r > i \geq 0, n \geq r > i \geq 0, r \neq k, i \neq k, r \neq j, i \neq j \).

Sequence \( p = (p[1], \ldots, p[n]) \) defines closed route or path \( \tau(p) = (0, p[1, \ldots, p[i, \ldots, p[n, \ldots, 0) \), in which all \( p[i] \in N \) are different. Cost of path \( \tau(p) \) is calculated as follows

\[
C(\tau(p)) = c_{01} + \sum_{i=1}^{n-1} c_{0} + c_{p[i]},
\]

where \( c_{00} = 0 \), \( c_{01} = 0 \).

Thus the minimization task (1) for the case, when the source matrix of processing times \( \left[ t_{ij} \right]_{m \times n} \) is ordered by rows, is a traveling-salesman problem with target functional (3), defined on the set of all pyramidal permutations \( P_r \), and with the matrix of costs \( \left[ c_{pj} \right]_{m \times n} \), whose elements are bound by inequalities a)-d).

Let \( p_l = (p[1, \ldots, p[k-1], p[k], p[k+1], \ldots, p[l]) \) be one of pyramidal permutations of subset \( \{1, 2, \ldots, l\} \) of set \( N \), \( l > 2 \), \( p[k] = l \). Cost of path \( \tau(p_l) \), corresponding to \( p_l \), can be calculated from table \( \left[ c_{pj} \right]_{m \times n} \), which is achieved by deleting rows and columns numbered \( l+1, \ldots, n \) from matrix \( \left[ c_{pj} \right]_{m \times n} \):
Sequence $p_i$ produces two partial pyramidal permutations with size $l + 1$:

- $p_{i^1}^1 = (\ldots, p[k-1], l + 1, p[k], p[k+1], \ldots)$,
- $p_{i^2}^2 = (\ldots, p[k-1], p[k], l + 1, p[k+1], \ldots)$

The costs of corresponding paths are:

- $C(\tau(p_{i^1}^1)) = C(\tau(p_i)) + c_{p[i-1][i+1]} + c_{\tau(p_{i^1}^1)} - c_{\tau(p_{i^1}^1)},$
- $C(\tau(p_{i^2}^2)) = C(\tau(p_i)) + c_{p[i][i+1]} + c_{\tau(p_{i^2}^2)} - c_{\tau(p_{i^2}^2)}.$

The general idea of the offered algorithm of minimization of (3) is step-by-step construction of partial pyramidal permutations with size $l$, $l = 3, n$, and exclusion of those permutations which lead to loss of optimal solution.

First of all it should be noted that when $l = 3$ there are two pairs of pyramidal permutations and corresponding paths:

- $\tau(p_i^1) = (0, 1, 3, 2, 0)$, $\tau(p_i^1) = (0, 3, 2, 1, 0)$ и $\tau(p_i^1) = (0, 1, 2, 3, 0)$.

As a result of path comparison in received pairs using properties a)-d) we can choose those pairs, which do not lead to loss of optimal solution of task of minimization of functional (3).

**The algorithm**

Let’s consider two pyramidal permutations $p_i$, $\sigma_i$ and corresponding paths:

- $\tau(p_i) = (0, \ldots, p[k-1], p[k], p[k+1], \ldots, 0)$, $p[k] = l$, $k = 1, l$;
- $\tau(\sigma_i) = (0, \ldots, \sigma[i-1], \sigma[i], \sigma[i+1], \ldots, 0)$, $\sigma[i] = l$, $i = 1, l$.

If $k = l$ then $p[k + 1] = 0$. The same way, if $i = l$ then $\sigma[i + 1] = 0$.

If $p[1] = l$ then $p[0] = 0$, and if $\sigma[1] = l$, then $\sigma[0] = 0$.

There’re following possible cases of comparison of paths $\tau(p_i)$ and $\tau(\sigma_i)$:

1. $p[k-1] = \sigma[i-1]$, $p[k+1] = \sigma[i+1]$. Obviously, the permutation with greater cost should be excluded. If $C(\tau(p_i)) = C(\tau(\sigma_i))$ then either one of two permutations $p_i$ and $\sigma_i$ can be excluded.

2. $p[k+1] = \sigma[i+1] = l-1$. It can be shown that if $p[k-1] > \sigma[i-1]$ and $C(\tau(p_i)) \leq C(\tau(\sigma_i))$ then only permutation $p_i$ should be considered later on.

Permuation $p_i = (\ldots, p[k-1], l, l-1, \ldots)$, $l < n$, produces two partial permutations $p_{i^1}^1 = (\ldots, p[k-1], l+1, l-1, \ldots)$ and $p_{i^2}^2 = (\ldots, p[k-1], l, l+1, l-1, \ldots)$.

3. $p[k-1] = \sigma[i-1] = l-1$. It can be shown that if $p[k+1] > \sigma[i+1]$ and $C(\tau(p_i)) \leq C(\tau(\sigma_i))$ then permutation $\sigma_i$ should be excluded from consideration.

Permuation $p_i = (\ldots, l-1, l, p[k+1], \ldots)$, $l < n$, produces two partial permutations with size $l + 1$:

- $p_{i^1}^1 = (\ldots, l-1, l+1, l, p[k+1], \ldots)$ and $p_{i^2}^2 = (\ldots, l-1, l, l+1, p[k+1], \ldots)$.

4. $p[k+1] = \sigma[i+1] = l-1$, $p[k-1] > \sigma[i-1]$, $C(\tau(\sigma_i)) < C(\tau(p_i))$, $l < n$.

Properties c) and d) do not allow to exclude from consideration either one of permutations $p_i$ and $\sigma_i$. If all possible permutations with size $l + 1$ are produced from $p_i$ and $\sigma_i$ then it is clear that in case 4 if $C(\tau(p_{i^1}^1)) \leq C(\tau(\sigma_{i^1}^1))$ only two of partial solutions $\sigma_{i^1}^1$ and $p_{i^1}^1$ need to be considered further, or if $C(\tau(\sigma_{i^1}^1)) < C(\tau(p_{i^1}^1))$ then only three permutations $\sigma_{i^1}^1$, $p_{i^1}^1$, $\sigma_{i^1}^1$ need to be considered.

5. $p[k-1] = \sigma[i-1] = l-1$, $p[k+1] > \sigma[i+1]$, $C(\tau(\sigma_i)) < C(\tau(p_i))$, $l < n$. 

The same way as in case 4, both permutations \( p_i \) and \( \sigma_i \) need to be expanded. If \( C(\tau(p_{i+1})) \leq C(\tau(\sigma_{i+1})) \) then only two permutations \( \sigma_{i+1} = (..., l-1, l+1, l, \sigma[i+1], ...) \) and \( p_{i+1} = (..., l-1, l, l+1, \sigma[k+1], ...) \) remain for consideration, or if \( C(\tau(\sigma_{i+1})) < C(\tau(p_{i+1})) \) then three partial solutions \( \sigma_{i+1} \), \( p_{i+1} \), \( \sigma_{i+1} = (..., l-1, l+1, \sigma[i+1], ...) \) need to be considered further to avoid loss of optimal solution of task (3).

Let us describe the exact algorithm for solution of limited version of TSP, formulated by (3).

S0. \([c_{pq}]_{m+1} \) – matrix of costs for TSP, in which values \( c_{pq} \), \( p = 0, n \), \( q = 0, n \) are bound by properties a)-d);

Build paths \( \tau^1 = (0.1, 3, 2, 0), \tau^2 = (0.1, 2, 3, 0), \tau^3 = (0.3, 2, 1, 0), \tau^4 = (0.2, 3, 1, 0) \), calculate their costs \( C(\tau^1), C(\tau^2), C(\tau^3), C(\tau^4) \); set \( l = 3 \), \( m = 4 \).

S1. If \( l = n \) then finish: the cost of the optimal path \( \tau^* \) equals \( C(\tau^*) = \min\{C(\tau_j) \mid j = 1, m\} \).

S2. Separate set \( \{\tau_j \mid j = 1, m\} \) into two subsets \( T_a \) and \( T_b \) in such manner that in each path belonging to set \( T_a \) index \( l-1 \) comes after \( l \), and in each path belonging to set \( T_b \) it comes before \( l \). Exclude from considerations “hopeless” paths in \( T_a \) and \( T_b \), using adequately cases 1,2 and 1,3.

S3. \( l = l + 1 \); expand every of remaining paths into two paths \( \tau^1 = (..., l, l-1, ..., \) ) and \( \tau^2 = (..., l-1, l, ..., \) ), calculate their costs \( C(\tau^1), C(\tau^2) \); go to step S1.

In the following passage we prove that the algorithm correctly solves the particular case of TSP, minimizing functional (3).

**Conclusion**

Let us describe the process of construction of minimal-cost path \( \tau^* \) using the tree of \( n \) levels. The only vertex of the first level, which is the root of the tree, is matched by path \((0,1,0)\) and its cost \( c_{00} \). The second level consists of 2 vertices \( \tau_1^2 = (0,1,2,0), \tau_2^2 = (0,2,1,0) \) and their costs \( C(\tau_1^2) = c_{12} + c_{20} \) and \( C(\tau_2^2) = c_{13} + c_{30} \). Every second-level vertex produces two third-level vertices \( \tau_1^3 = (0,1,3,2,0), \tau_2^3 = (0,1,2,3,0) \), \( \tau_3^3 = (0,3,2,1,0), \tau_4^3 = (0,2,3,1,0) \), which define the set of all pyramidal permutations of size 3. Vertex \( \tau_1^3 \) produces a pair of descendants \( \tau_3^3 \) and \( \tau_4^3 \), and vertex \( \tau_2^3 \) produces \( \tau_1^3 \) and \( \tau_2^3 \). Every third-level vertex \( \tau_j^4 \), \( j = 1,4 \), is assigned a weight equal to the cost of corresponding path: \( C(\tau_j^4) = C(\tau_j^1) + c_{j+1} + c_{j-1} - c_{j} \), \( C(\tau_j^2) \), \( C(\tau_j^3) \). Any vertex of level \( l \), \( l = 4, n \), produces not more than two descendants. The set of all paths \( \tau_j^*, j = 1, m \), containing the optimal task solution, can be presented as a union of non-overlapping subsets \( T_{22} \) and \( T_{23} \). Subset \( T_{22} \) consists of all paths in which index 2 follows index 3, while subset \( T_{23} \) contains all paths where index 2 comes before index 3. Obviously, values \( C(\tau_j^1) \) and \( C(\tau_j^2) \) limit from below the minimum of functional (3) defined on subset \( T_{22} \). In the same way costs \( C(\tau_j^3) \) and \( C(\tau_j^4) \) are lower bounds for the minimum of functional (3) on subset \( T_{23} \).

During comparison of paths \( \tau^1 \) and \( \tau^3 \) which compose subset \( T_a \), if \( C(\tau_j^1) \leq C(\tau_j^3) \) then \( \tau^1 \) is excluded, which means that all admissible solutions \( \tau^1 \in T_{22} \) with lower bound greater or equal to \( C(\tau_j^3) \) are also excluded. Thus at level 4 vertex \( \tau_1^4 \) produces two vertices \( \tau_1^4 = (0,1,4,3,2,0) \) and \( \tau_2^4 = (0,1,3,4,2,0) \), \( \tau_4^4 \in T_a \), \( \tau_5^4 \in T_b \).

Comparison of paths \( \tau^2 \) and \( \tau^4 \) which compose subset \( T_b \), in the case when \( C(\tau_j^2) \leq C(\tau_j^4) \) allows to exclude \( \tau^2 \) and all admissible solutions in \( T_{23} \) for which the lower bound of (3) is no less than \( C(\tau_j^4) \). At level \( l = 4 \) vertex \( \tau_1^5 \) produces two vertices \( \tau_1^5 = (0,2,4,3,1,0), \tau_2^5 = (0,2,3,4,1,0) \), \( \tau_4^5 \in T_a \), \( \tau_5^5 \in T_b \).
If $C(τ_i^3) < C(τ_i^4)$ then as a result of comparison of costs of paths $(0,1,3,4,2,0)$ and $(0,3,4,2,1,0)$, correspondingly equal to $C(τ_i^3) + c_{34} + c_{42} - c_{32}$ and $C(τ_i^4) + c_{34} + c_{42} - c_{32}$, path $(0,1,3,4,2,0)$ is excluded from subset $T_{3i}$, and, obviously, all admissible solutions like $(0,1,3,...,n,...,2,0)$ are also excluded. In this case at level $l = 4$ vertices $τ_i^3$ and $τ_i^4$ produce three vertices $τ_i^3 = (0,1,4,3,2,0)$, $τ_i^4 = (0,4,3,2,1,0)$, $τ_i^5 = (0,3,4,2,1,0)$, $τ_i^3 \in T_K$, $τ_i^4 \in T_L$.

If $C(τ_i^3) < C(τ_i^4)$ then as a result of comparison of costs of paths $(0,1,2,4,3,0)$ and $(0,2,4,3,1,0)$, correspondingly equal to $C(τ_i^3) + c_{24} + c_{43} - c_{23}$ and $C(τ_i^4) + c_{24} + c_{43} - c_{23}$, partial solution $(0,2,4,3,1,0)$ and all admissible solutions like $(0,2,...,n,...,3,1,0)$ from subset $T_{3i}$ are excluded. Then at level 4 vertices $τ_i^3$ and $τ_i^4$ produce three vertices $τ_i^3 = (0,1,2,4,3,0)$, $τ_i^4 = (0,1,2,3,4,0)$, $τ_i^5 = (0,2,3,4,1,0)$, $τ_i^3 \in T_K$, $τ_i^4 \in T_L$, $τ_i^5 \in T_L$.

In general case as a result of comparing paths received at level $l - 1$, $l = 3, n - 1$ there are formed subsets of partial solutions $T_K$ and $T_L$, where the former one contains solutions like $(...,l-1,l,...)$ and the latter one contains solutions like $(...,l-1,l,...)$. At the $n$-th level of solution tree set $T_K \cup T_L$ is matched by a subset of set $T_{32} \cup T_{33}$ of admissible tasks solution, containing the optimal solution.

Using the obvious logic it can be shown that at level $l$, $l = 2, n$, the number of partial solutions excluding the loss of optimal one is limited by value $2l - 2$.

Let us estimate the upper bound for the number of all solution-tree vertices:

$$\sum_{l=1}^{n} m_l \leq \sum_{l=2}^{n} (2l - 2) = n(n - 1).$$

The above-given arguments allow us to make the following statement.

Theorem. The algorithm presented by a sequence of steps S0-S3 constructs solution which delivers the minimum of functional (3) in time $O(n^2)$.

Bibliography


Author information

Anatoliy Vasilyovich Panishev – head of the chair of informatics and computer modeling, professor, doctor of technical sciences, Zhitomir State Technological University, Chernyahovskogo 103, Zhitomir, Ukraine, 10003; e-mail: dep@ziet.zhitomir.ua

Dmytro Dmytrovych Plechystyy – teacher-assistant of the chair of informatics and computer modeling, aspirant, Zhitomir State Technological University, Chernyahovskogo 103, Zhitomir, Ukraine, 10003; e-mail: dep@ziet.zhitomir.ua