Krassimir Markov, Vitalii Velychko, Oleksy Voloshin
(editors)

Information Models of Knowledge

I T H E A®
KIEV – SOFIA
2010

Krassimir Markov, Vitalii Velychko, Oleksy Voloshin (ed.)
Information Models of Knowledge
ITHEA®
Kiev, Ukraine – Sofia, Bulgaria, 2010
ISBN 978-954-16-0048-1

First edition
Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA
ITHEA IBS ISC: 19.

This book maintains articles on actual problems of research and application of information technologies, especially the new approaches, models, algorithms and methods fot information modeling of knowledge in: Intelligence metasynthesis and knowledge processing in intelligent systems; Formalisms and methods of knowledge representation; Connectionism and neural nets; System analysis and sintesis; Modelling of the complex artificial systems; Image Processing and Computer Vision; Computer virtual reality; Virtual laboratories for computer-aided design; Decision support systems; Information models of knowledge of and for education; Open social info-educational platforms; Web-based educational information systems; Semantic Web Technologies; Mathematical foundations for information modeling of knowledge; Discrete mathematics; Mathematical methods for research of complex systems.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Ukraine

ISBN 978-954-16-0048-1

C\o Jusautor, Sofia, 2010

CYCLE ROUTES OPTIMIZATION FOR NOT FULL GRAPH

Anatoly Panishev, Anton Levchenko

***Abstract**: Searching for cycling routes of minimum cost is an urgent task of transport nets designing, traffic flows routing, etc. Each locality, transport node can be the vertex of the graph, and each cut line, linking localities, − the edge of the graph. The edge's cost represents distance, repair cost, channel bandwidth, circuit resistance, etc.*

If closed route reaches all vertexes of the graph exactly once, it is called Hamiltonian cycle (HC), and the task of searching for minimum cost is the Hamiltonian Traveling Salesman's Problem (HTSP). HTSP is NP-hard and not always has a solution [Майника, 1981]. Algorithms delivering optimum to HTSP are described as an exhaustive search reduction.

If restriction of visiting only once every vertex is taken off, then such task can be called Common Traveling Salesman Problem (CTSP). Besides completeness of the route is an additional restriction for this task. CTSP seems the generalization of HTSP [Бондаренко, 2004]. For every linked graph always the CTSP's set of solutions is not empty, and it allows using approximate algorithms and heuristics. The authors have no data about exact methods, bringing optimum of closed CTSP.

This article offers exact algorithm of CTSP solution, bringing the cycle of minimum cost, which includes all vertexes of the graph at least once and contains the less number of edges.

***Keywords**: TSP, Traveling Salesman Problem, Hamiltonian cycle, closed route, optimal route, exact algorithm.*

***ACM Classification Keywords**: Algorithms, Theory.*

Introduction

Transport nets, electric chains are often presented as graph. Most methods come from the idea that graph is full. The ideas, offered in the article, allow using standard methods for arbitrary connected graphs.

One of the most famous graph problems is Traveling Salesman's Problem (TSP). We call Hamiltonian TSP (HTSP) task of finding minimal cost route visiting all graph's vertexes only once [Бондаренко, 2004]. If route can visit any vertex at least once we call such task Common TSP (CTSP).

Algorithm's foundation

The graph $H=(U,V)$ is given, where V – is the set of vertexes, U – is the set of edges, connecting these vertexes. Every edge $\{i,j\}\in U$ has cost $d_{ij}\in R_0^+$, where R_0^+ – is the set of real non-negative numbers. The task is to find in graph H minimal cost cycle, which reaches every vertex of the graph.

If H is linked, then every random pair of vertexes $i,j\in V$ is connected by the set of simple chains A_{ij}, this set contains the chain α_{ij} with the minimal total weight of included edges $D(\alpha_{ij})$. Matrix $\left[D(\alpha_{ij})\right]_n$ specifies the full graph $H_\alpha=(V,E)$, with every edge $\{i,j\}\in E$ responses to the chain α_{ij} with weight $D(\alpha_{ij})$ in the graph H. H_α is full and meets triangle's condition

$$d_{ij}\le d_{ik}+d_{kj}\ ,\ i\ne k\ne j \tag{1}$$

By analogy with (1) let's write following inequality, characterizing weight relationships between edges of the graph H, even if it is not full:

$$d_{ij}\le D(\alpha_{ij}) \tag{2}$$

There are two possibilities: a) all edges of H meet inequality (2), b) at least one of the edges doesn't meet (2).

The graph $H=(V,U)$ is assumed to be Hamiltonian; τ, T – are optimal solutions of HTSP and CTSP accordingly, which are built in the H, $D(\tau)$ и $C(\mathrm{T})$ – costs of these built solutions. Let's find the minimal cost cycle σ in the full graph H_α.

Predicate 1. If inequality (2) holds for all edges $\{i,j\}\in U$ of the Hamiltonian graph, $H=(V,U)$ then $\mathrm{T}=\tau$, $C(\mathrm{T})=D(\tau)$.

Demonstration. HTSP solution τ contains n edges of the set U. In case a) the edge $\{i,j\}$ of the full graph H_α has cost $D(\alpha_{ij})=d_{ij}$, if $\{i,j\}\in U$, and $D(\alpha_{ij})\geq d_{ij}$ else. Implies, that in the H_α exact solution of TSP σ matches τ, and it's cost doesn't exceed the cost of random route which contains n and more edges, including T. $\square$

Predicate 2. If at least for one edge $\{i,j\}$ of the Hamiltonian graph, $H=(V,U)$ inequality (2) doesn't hold, then $C(\mathrm{T})\leq D(\sigma)$.

Demonstration. In case b) in the full graph H_α there is at least one edge $\{i,j\}$, which in the H has cost $d_{ij}>D(\alpha_{ij})$. If Hamiltonian cycle σ of the graph H_α contains the edge $\{i,j\}$, then it's cost is greater than the cost of the corresponding route in the H, which contains chain α_{ij} instead of the edge $\{i,j\}\in U$. If the cycle σ doesn't include edges, that break inequality (1), then it matches HTSP. $\square$

The graph $H=(V,U)$ is assumed not to be Hamiltonian. Then in any case a) or b) cost of the optimal TSP solution σ for the full graph $H_\alpha=(V,E)$ is equal to cost of the CTSP's solution T for the graph H. So closed route T could be found by building Hamiltonian cycle σ in the graph H_α, and replacing every edge $\{i,j\}\in E$ by the chain α_{ij} obtained from edges of the set U.

Exact solution algorithm for CTSP

S0. $H=(V,U)$ – linked weighted graph with the set of vertexes V, $|V|=n$, and the set of edges U, $[d_{ij}]_n$ – the matrix of weights of the graph H, where if $\{i,j\}\in U$ then $d_{ij}\in R_0^+$, else $d_{ij}=\infty$; $i,j=\overline{1,n}$, R_0^+ – the set of real non-negative numbers.

S1. Build by Floyd's algorithm the matrix $[\alpha_{ij}]_n$ of the shortest chains between all pairs of vertexes of the graph H and the matrix $[D(\alpha_{ij})]_n$, where every element (i,j) is equal to the cost $D(\alpha_{ij})$ of the chain α_{ij}; matrices $[\alpha_{ij}]_n$ and $[D(\alpha_{ij})]_n$ sets the full weighted graph $H_\alpha=(V,E)$, where every edge $\{i,j\}$ replaces the chain α_{ij} in the graph H.

S2. Find minimal cost circuit σ in the graph H_α by any known algorithm of metric TSP solution.

S3. Build the optimal solution of CTSP T replacing every edge $\{i,j\}$ of circuit σ by the graph's H chain α_{ij}.

Modified Little's method

Graph H can contain several tours with the cost equal to optimal, but with different edges number, with the above described algorithm can select any of them. Proposed use at the step S2 modification of the classic Little's algorithm, which reaches optimum with less number of edges.

According to Little's method, building of the optimal solution occurs during branching with the purpose of partition of the set of feasible solutions to disjoint subsets. These subsets are represented as vertexes of the solutions tree

[Харари, 1973]. The root of the tree is the vertex $\varnothing$, denoting the set of all feasible TSP solutions. It is the initial vertex of branching.

Let $D(\alpha_{ij})$ is value of the element (i,j) of the reduced matrix of costs of the initial graph, $i,j=\overline{1,n}$. Every matrix's element is represented as an arc in the oriented multigraph G_α, where every two vertexes i and j are connected by the pairs of arcs (i,j) and (j,i).

Branching starts with selection of the most appropriate for tour arc (k,l) of the multigraph G_α. The set of feasible solutions is divided to two subsets: including the arc (k,l), and not including that arc. These subsets vertexes can be marked as $((k,l)\circ)$ and $((\overline{k,l})\circ)$ accordingly. For them lower bounds of TSP's costs $\varphi((k,l)\circ)$ and $\varphi((\overline{k,l})\circ)$ are calculated. Reduced matrices are formed too. Among the terminal vertexes of the search tree the vertex with the lowest bound is determined – a branching vertex. According to the branching vertex the subset is divided to two subsets by arc, based on reduced matrix. For the corresponding vertexes reduced matrixes are formed and lower bounds calculated.

The way from the search tree's root to terminal vertex contains a part of a feasible TSP's solution consisting of arcs included in the process of branching. The method is considered completed when the next branching vertex includes all tour's arcs in multigraph G_α.

The number of edges in a tour can be called length of tour. If at any step of the branching several terminal vertexes have the same minimum bound, then Little's method offers to choose any of them that do not guarantee a CTSP's solution with the lowest length. We introduce an additional parameter which is a lower bound for the length of the resulting tour for each vertex.

In the building of solutions tree the path from the root to terminal vertex contains the arcs included and not included in the partial solution. Let $P_j=\{(v_{j_p},v_{j_q})\}$ be the set of arcs included into the partial solution as a result of building of the path from root to terminal vertex j. Every arc (v_{j_p},v_{j_q}) of the set P_j in multigraph G_α is an edge (v_p,v_q), in the graph H with corresponding shortest chain α_{pq}, which connects vertexes p and q and contains $l(\alpha_{pq})$ edges. Then the partial solution for vertex j includes $l_j=\sum_{(v_{j_p},v_{j_q})\in P_j} l(\alpha_{pq})$ edges of the graph H.

Reduced matrix $[d']_j$ of the vertex j containts at least one zero element in every column and row. $d'_{ij}=0$ means the arc (i,j) of multighraph G_α is the most likely candidate for inclusion in the desired tour. From $[d']_j$ columns $\{j_p\}$ and rows $\{j_q\}$ are deleted corresponding to arcs $\{(v_{j_p},v_{j_q})\}$, included to partial solution P_j. Zeros in the same row (column) of the matrix $[d']_j$ correspond to arcs of G_α, starting or ending in the same vertex.

Let's build matrix $[d'']_j$ in such a way that if $d'_{ij}\neq 0$, then $d''_{ij}=\infty$, else $d''_{ij}=l(\alpha_{ij})$. Obviously, adding G_α to tour arc $\{i,j\}$ with $d'_{ij}=0$, results to the fact that tour in the graph H increases by d''_{ij} edges. Let τ''_j is a permutation of columns of matrix $[d'']_j$, which is solution of Assignment Problem (AP). The cost of this solution $C(\tau''_j)$ is the sum of diagonal elements of permutated matrix and this sum is minimal. Corresponding to diagonal matrix's elements edges don't start or finish at the same vertex. Consequently $C\left(\tau_j''\right)$ is the lower bound for the number of edges of graph H, which can be included to the tour in the further branching of the search tree.

Let every terminal vertex j, is characterized by value:

$$L_j = l_j - \left|P_j\right| + C\left(\tau_j''\right) \tag{3}$$

in addition to lower bound $\varphi\left(j\right)$. If lower bound of several terminal vertexes is equal then we choose the vertex with minimal value of (3) that facilitates the choice of the optimal solution with less length.

Graph's topology and algorithm's productivity

Graph of the actual transport network carries information to assess whether it is connected. The three components of the graph describing its connectivity are the articulation point, bridge, and a subset of the pendant or terminal nodes.

If deletion of vertex transforms a connected graph into a disconnected, then it is called the point of articulation (cutpoint), and an edge with the same property - a bridge. A connected, non-empty, having no cutpoints subgraph of graph H is called a block. Cutpoint is a common vertex of several blocks [Харари, 1973]. Non-empty subset of terminal nodes of graph H creates subgraph H' in the form of forest [Гаращенко, 2007]. Every tree of the forest corresponds to root node, connecting it with the part of the graph H, which doesn't include edges and other vertexes of the tree. Obviously, the each tree's root vertex is a point of articulation, and therefore, every tree of the forest is a block where all vertexes are the points of articulation, except for the hanging ones.

The above mentioned algorithm of CTSP's exact solution permits improvement, reducing the work time of the branch and bound procedure for a graph containing blocks of trees and bridges. The improvement mechanism is based on demonstrable fact.

Predicate 3. Any CSTP's solution for a tree has a value equal to the double sum of weights of tree's edges.

Let graph H contain blocks built from bridges and trees. First we select all blocks-trees in H and build for each of them closed route starting in route vertex and visiting every vertex, and passing every edge twice.

Each route planning, we note the root vertex. Then we select in graph H all bridges and mark the articulation point of each bridge. Obviously, the cost of a closed path on the bridge is equal to the double weight of the edges, representing the bridge. The sum of the values of the routes built for the selected objects is a constant component of the cost of any CTSP's solution in graph H.

Let's study the subgraph obtained by removing from the graph H all bridges and all trees except root vertexes. The subgraph is not linked if the graph H contains bridges. The connected components of subgraphs together contain all the marked vertexes. The component could be an articulation point, a block or a decomposable subgraph, i.e. subgraph containing the articulation points, other than marked. Then we construct the salesman's route for each component, which begins and ends at the marked nodes incidental to the bridge. CTSP's solution for graph H is the result of union all successive routes in the marked vertexes.

To select the trees of the forest H' apply version of the algorithm proposed in [Гаращенко, 2007]. Mark as V' and U' sets of vertexes and matrices of the forest $H'\left(V',U'\right)$, K – set of root vertexes, $K \subset V'$. The following algorithm constructs all the trees of the forest H' in graph H and defines set of root vertexes K.

S0. $H = \left(V,U\right)$ – linked graph where V – set of vertexes, U – set of edges $u = \left\{i,j\right\}$, $\left|V\right| = n$; vertexes of graph H placed in non-decreasing order of their degrees: $\deg 1 \le \deg 2 \le ... \le \deg n$; $K = \varnothing$, $V' = \varnothing$, $U' = \varnothing$.

S1. If $\deg 1 > 1$, then end: graph H doesn't contents graph H', else $i = 1$.

S2. For edge $u = \left\{i,j\right\}$ let $\deg i = 1$, $\deg j = \deg j - 1$, $V = V - \left\{i\right\}$, $V' = V' \cup \left\{i,j\right\}$, $U' = U' \cup \left\{u\right\}$; $i = i + 1$.

S3. If $i = n-1$, then $V' = V$, $U' = U$, end: graph H is a tree.
S4. If $\deg i = 1$, then go to S2.
S5. $K = V' \cap V$, to build for every root vertex from K tree of forest H'.
Time of the algorithm is obviously commensurable with the time of ordering vertexes' degrees of the graph H, evaluated, as it is well known by value $O(n \log n)$.

Every edge $\{v,w\}$ of the tree H'_k, $k = \overline{1,|K|}$, generally speaking, is a bridge, the removal of which leads to a non-connected spanning subgraph of graph H, not containing $\{v,w\}$.

Let's consider the subgraph $\langle S \rangle$ of the subgraph H, generated by a subset of vertexes $S = (V - V') \cup K$. To find and select all the bridges in the subgraph $\langle S \rangle$ depth-first search algorithm is used that determines the set of vertexes S all articulation points during $O(|S| + |V| - |V'|)$ [Рейнгольд, 1980]. If a pair of cutpoints is connected by an edge deletion, which increases the number of connected components of the subgraph S, then it creates bridge M_m. Set of s bridges generates $s+1$ connected components H''_l of subgraph $\langle S \rangle$. Closed route visiting all vertexes of the component H''_l, $l = \overline{1,s+1}$, and delivering the minimum sum of weights of edges, is a part of traveling salesman's tour τ in graph H.
We show how to unite to CSTP's solution τ routes τ'_k, e_m, τ''_l, which are built for trees H'_k of the forest H', $k = \overline{1,|K|}$, bridges M_m, $m = \overline{1,s}$, and connected components H''_l of subgraph $\langle S \rangle$, $l = \overline{1,s+1}$.
Among all the components H''_l always exists such that contains exactly one vertex V_u of the bridge. It is defined as the first in the set of components H''_l, $l = \overline{1,s+1}$. The vertex V_u is marked as the starting and ending point of a salesman's route τ. If $s > 2$, then the part of components H''_l, $l = \overline{2,s+1}$, is interconnected by several bridges. In such components there is one vertex of each bridge.
Let $\tau''_l = (w,\ldots,a,\ldots,b,\ldots,w)$ be the salesman's tour built for H''_l, $l = \overline{1,s+1}$, where w – bridge's vertex; $w = v$ if $l = 1$. Suppose it contains vertexes from the set of root vertexes K. Then we execute in τ''_l replacement of every vertex $a \in K$ by the route $(a,\ldots,a)$ for the tree H'_k with ending vertex a. We mark all vertexes in every route τ''_l, incidental to bridges.

Let's build graph (L,M), in which to every vertex $l \in L$, $l = \overline{1,s+1}$, is assigned to the route τ''_l, and to every edge $\{i,j\} \in M$ – a bridge M_m, $m = \overline{1,s}$. In graph (L,M) a pair of vertexes i and j creates an edge $\{i,j\}$, if vertexes $p \in \tau''_i$ and $q \in \tau''_j$ are connected by bridge $\{p,q\}$, $i \neq j$. Graph (L,M) is a tree, since by construction it is linked, and $|L| = |M| - 1$. We assume vertex 1 of the tree (L,M) as a root vertex. To salesman's tour τ every tour is assigned including all tree's vertexes, which starts and ends in the vertex 1 and и runs twice along every edge from M.

All details of uniting to tour τ of sets of routes $\left\{\tau''_l \middle| l = \overline{1,s+1}\right\}$, linked by bridges M_m, $m = \overline{1,s}$, can be presented on the basis of Fig. 1. First τ includes tour $\tau''_1 = (v,\ldots,v)$ and the bridge $\{v,w\}$, linking τ''_1 with route $\tau''_2 = (w,\ldots,x,\ldots,y,\ldots,w)$. As x and y – marked vertexes, i.e. vertexes of bridges $\{x,a\}$ и $\{y,b\}$, then τ runs along the part $(w,\ldots,x)$ of tour τ''_2, edge $\{x,a\}$, route $\tau''_3 = \{a,\ldots,a\}$ and edge $\{a,x\}$. Then it includes the part $(x,\ldots,y)$ of the route τ''_2, edge $\{y,b\}$, route $\tau''_4 = (b,\ldots,b)$, edge $\{b,y\}$, one more part $(y,\ldots,w)$ of the route τ''_2 and the edge $\{w,v\}$. Thus the tour $\tau = (v,\ldots,v,w,\ldots,x,a,\ldots,a,x,\ldots,$ $y,b,\ldots,b,y,\ldots,w,v)$ is built. Obviously, the time to unite all the selected subgraphs of the graph H is limited by value $O(|V|)$.

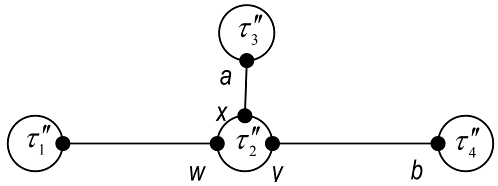
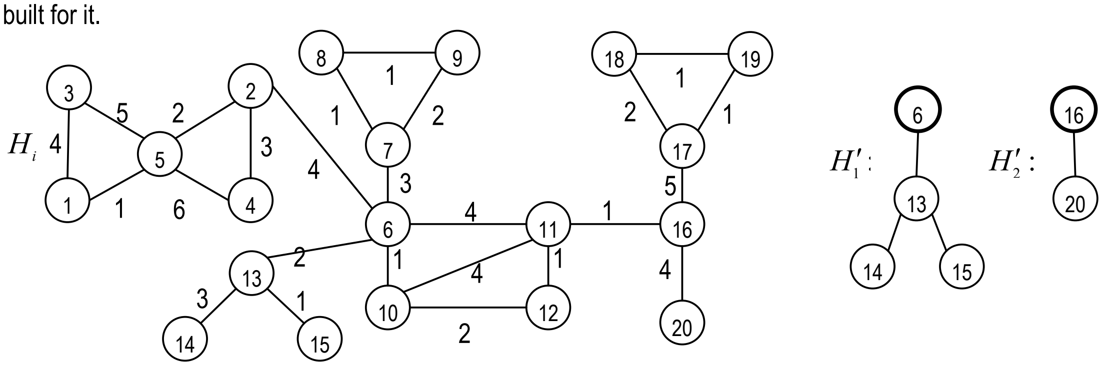


Fig. 1. The tree (L,M) with distinguished vertexes

Salesman's tour's τ cost equals to $C(\pi)=\sum_{k=1}^{|K|} C(\tau'_k)+2\sum_{m=1}^{s} C(M_m)+\sum_{l=1}^{s+1} C(\tau''_l)$, where the first two terms are constants for a given graph H. The cost of subgraph's $\langle S\rangle$ components, which is represented by one vertex, is equal to 0.

Example 1. At the Fig 2. a connected weighted graph H is shown. Closed salesman's tour τ is required to be built for it.



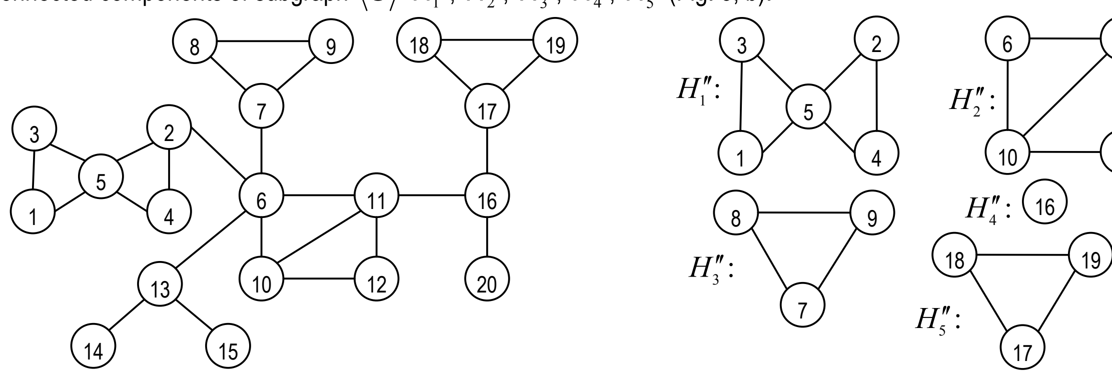
a) b)

Fig. 2 a) original graph; б) trees of the forest H'.

First the forest's H' building algorithm is performed, which consists of two stages. In the first stage it sorts all graph's H vertexes by their degrees in non-decreasing: deg 14=1, deg 15=1, deg 20=1, deg 1=2, deg 3=2, deg 4=2, deg 8=2, deg 9=2, deg 12=2, deg 18=2, deg 19=2, deg 2=3, deg 13=3, deg 7=3, deg 10=3, deg 16=3, deg 16=3, deg 17=3, deg 5=4, deg 11=4, deg 6=5. In the second stage algorithm finds the set of forest's vertexes $V=\{14,15,13,6,20,16\}$, subset of root vertexes $K=\{6,16\}$ and builds и trees H'_1, H'_2 (Fig. 2, b). Trees' root vertexes at Fig. 2 b) are marked by thicker lines of circles.

Generated by subset $S=\{1,2,3,4,5,6,7,8,9,10,11,12,16,17,\ 18,19\}$ subgraph $\langle S\rangle$ is shown at fig 3 a).

Finding cutpoints algorithm finds there are 7 such cutpoints in $\langle S\rangle$: 5, 2, 6, 7, 11, 16, 17. Vertexes 2, 6, 7, 11, 16, 17 from them create bridges $M_1=\{2,6\}$, $M_2=\{6,7\}$, $M_3=\{11,16\}$. Deletion of bridges makes 5 connected components of subgraph $\langle S\rangle$ H''_1, H''_2, H''_3, H''_4, H''_5 (Fig. 3, b).



a) b)

Fig. 3 a) the induced subgraph $\langle S\rangle$; b) subgraph's $\langle S\rangle$ connected components.

We build closed salesman's routes, starting and ending at the marked vertexes, for each tree and each connected component of subgraph $\langle S \rangle$: $\tau_1' = (6,13,14,13,15,13,6)$, $\tau_2' = (16,20,16)$, $\tau_1'' = (2,4,2,5,3,1,5,2)$, $\tau_2'' = (6,10,12,11,6)$, $\tau_3'' = (7,8,9,7)$, $\tau_4'' = (16)$, $\tau_5'' = (17,18,19,17)$. As a result of route τ_1' unites with route τ_2'' we get the tour $\tau_2'' = (6,13,14,13,15,13,6,10,12,11,6)$. Union τ_2' with τ_4'' gives the rout $\tau_4'' = (16,20,16)$.

The tree (L, M), built for subgraph $\langle S \rangle$ of graph H, is presented at Fig 4.

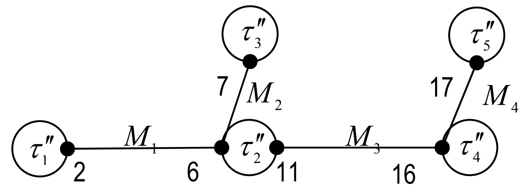


Fig. 4. The corresponding to subgraph $\langle S \rangle$ of graph H tree (L, M).

Select any tour of the tree's (L, M) vertexes, which starts and ends in vertex 1. Assume that it is (1, 2, 3, 2, 4, 5, 4, 2, 1). Build salesman's tour τ for graph H $\tau =$(2, 4, 2, 5, 3, 1, 5, **2**, **6**, 13, 14, 13, 15, 13, **6**, **7**, 8, 9, .**7**, **6**, **11**, **16**, 20, **16**, **17**, 18, 19, **17**, **16**, **11**, 12, 10, **6**, **2**). In τ bridge's vertexes are marked bold, and rout τ_2'' is presented by two parts (**6**, **11**) and (**11**, 12, 10, **6**), which are doesn't following one after the other. The cost of such CTSP's solution for the graph H is $C(\tau) = \sum_{k=1}^{2} C(\tau_k') + 2\sum_{m=1}^{4} C(M_m) + \sum_{l=1}^{5} C(\tau_l) =$ (2+3+3+1+1+2)+(4+4)+2(4+3+1+ +5)+(3+3+2+1+4+5+2)+ (1+2+ +1)+ (2+2+1)=75. □

Conclusion

The exact algorithm of CTSP solution is offered. The modification of Little's method is also offered which chooses among several optimal solutions the one with the least number of graphs. The procedure of branching for graphs with the determined topology can be more effective. Some features of his topology are described in the paper.

Bibliography

[Майника, 1981] Майника Э. Алгоритмы оптимизации на сетях и графах. – М.: Мир, 1981 .- 323 с.

[Бондаренко, 2004] Бондаренко М.Ф., Белоус Н.В., Руткас А.Г. Компьютерная дискретная математика. – Харьков: «Компания СМИТ», 2004 .- 476 с.

[Харари, 1973] Харари Ф. Теория графов. – М.: Мир, 1973. – 300 с.

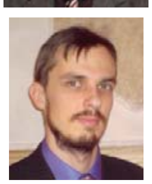
[Гаращенко, 2007] Гаращенко И.В., Панишев А.В. Об одном классе задач построения остовного дерева в неориентированном взвешенном графе // Искусственный интеллект. – Донецк, 2007. – Вып. 3. – С. 486-493

[Рейнгольд, 1980] Рейнгольд Э., Нивергельт Ю, Део Н. Комбинаторные алгоритмы. Теория и практика. – М.: Мир, 1980. –476 с.

Authors' Information

Anatoliy Panishev *– Professor, Doctor of Technology, Chief of the Chair of Informatics and Computer Modeling ZSTU; Chernyakhovsky str. 103, 10005 Zhitomir, Ukraine; e-mail: irinagar@rambler.ru*

Major Fields of Scientific Research: Combinatorial optimization, Graph theory



Anton Levchenko *– Assistant, Chair of Informatics and Computer Modeling ZSTU; Chernyakhovsky str. 103, 10005 Zhitomir, Ukraine ; e-mail: dear_anton@yahoo.com*

Major Fields of Scientific Research: Combinatorial optimization, Software technologies