UDC 519.161
DOI: 10.15587/1729-4061.2019.188104

CONSTRUCTING AN ALGORITHM OF QUADRATIC TIME COMPLEXITY FOR FINDING THE MAXIMAL MATCHING

A. Morozov
PhD, Associate Professor,
Vice Rector in Scientific and Pedagogical Work
Department of Computer Science*
E-mail: morozov@ztu.edu.ua

T. Loktikova
Senior Lecturer
Department of Software Engineering*
E-mail: dfikt_ltn@ztu.edu.ua

I. Iefremov
PhD, Associate Professor
Department of Software Engineering*
E-mail: org_eyum@ztu.edu.ua

A. Dykyi
PhD, Associate Professor
Department of Economic Security,
Public Administration and Management*
E-mail: anatoliy@ztu.edu.ua

P. Zabrodskyy
PhD, Associate Professor
Department of Mechanics and Agroecosystems Engineering
Zhytomyr National Agroecological University
Staryi blvd., 7, Zhytomyr, Ukraine, 10008
E-mail: zabrm@gmail.com

*Zhytomyr Polytechnic State University
Chudnivska str., 103, Zhytomyr, Ukraine, 10005

Базуючись на розвитку ідеї пошуку в ширину у дводольних графах та основних визначеннях теорії паросполучень, показано, що задача побудови максимального паросполучення в довільному графі може бути зведена до його дводольного випадку. Доведено, що кожному поточному паросполученню в довільному графі взаємно однозначно відповідає паросполучення в дводольному графі. Проілюстровано, що жодний з поточних розв'язків задачі побудови максимального паросполучення в довільному графі не втрачається при переході до ітераційної схеми побудови максимального паросполучення у дводольному графі.

Для знаходження збільшуючого шляху відносно фіксованого паросполучення потужності k запропоновано модифікацію відомого алгоритму пошуку шляхів з даної вершини у всі досяжні вершини довільного графу. Роботу запропонованої модифікації проілюстровано на прикладі.

На основі викладених ідей, доведених тверджень та запропонованих алгоритмів та їх модифікації побудовано алгоритм знаходження максимального паросполучення з покращеною часовою оцінкою, порівняно з відомим алгоритмом Едмонса, що має часову оцінку складності $O(n^4)$. Основним недоліком алгоритму Едмонса є використання трудомісткої техніки стиснення циклів непарної довжини, які називають «квітками», що робить алгоритм непридатним для застосування в системах реального масштабу часу. Інші відомі алгоритми відрізняються від алгоритму Едмонса тільки більш досконалою організацією зберігання даних та обчислень, разом з тим зберігаючи складні дії по виявленню і упаковці циклів непарної довжини.

Запропонований підхід переходу від довільного графу до дводольного графу дозволив уникнути виникнення циклів непарної довжини, що дозволило значно підвищити ефективність алгоритму. Подальше підвищення продуктивності можливо за рахунок побудови паралельних версій алгоритму і оптимальної організації зберігання даних

Ключові слова: паросполучення, максимальне паросполучення, дводольний граф, збільшуючий шлях, задача про призначення

Received date 20.09.2019
Accepted date 18.11.2019
Published date 28.12.2019

1. Introduction

Numerous problems, known as routing tasks, are characterized by an ever-expanding list of practical applications, occupying a traditionally important place in the study of combinatorial optimization problems. The routing task in a broad sense is the problem on current planning, the process of which involves the selection of movable objects and the determination of trajectories and schedules of their movement.

The tasks on routing in road transportation, as well as the methods to solve them, are studied within the framework of a scientific field – transport logistics, whose mathematical apparatus is represented by the theory of graphs and the study of operations. Most routing tasks are NP-complete [1] and can be solved only by combinatorial sorting methods [1, 2]. These methods often require the use of significant computing resources and, as a result, a long time to solve the problem.

The most commonly used methods for solving NP-complete routing problems are the branch and boundary methods [2], which employ relaxation to calculate the lower and upper bounds. Relaxation is generally understood as a combinatorial optimization problem, the set of whose valid solutions are injected with a set of valid solutions to the original problem [3]. Typically, the original NP-problem is complete, and its relaxation is solved over a polynomial time [3].

One of the problems that can be used as a relaxation to an NP-complete salesman problem [3, 4] is the problem about matching (MP).

In [4], a solution to MP is used as an element of the algorithm to obtain an approximate solution to the salesman's problem. The downside of the algorithm from [4] is its computational complexity, specifically $O(n^4)$, which makes the algorithm almost unusable for real-time use. The relevant area of research is to derive a faster algorithm for solving MP, which would speed up the implementation of algorithm [4].

A solution to MP could also be used as a lower bound in the branch and boundary methods used to solve closed-route problems. In most implementations of the branch and boundary method, the vertices of the solution tree are matched with a distance matrix that excludes some of the elements, or some rows and columns are removed. Thus, another important area of research is the construction of an algorithm that, based on an existing solution to MP, finds a new solution to MP for the matrix, which differs from the original one by the absence of some elements.

2. Literature review and problem statement

Paper [1] summarizes the problems on combinatorial optimization related to the construction of vehicle routes. It shows that most routing problems are NP-complete problems of discrete optimization, deriving solutions to which employs sorting methods. This is the method that is considered in work [2], which proposes the use of the branch and boundary method to solve various combinatorial problems on processing big data. Study [3] proposes the use of relaxations as the elements of algorithms and methods for solving the problems on building closed routes, including within the methods of branches and boundaries. Work [4] proposes to use the 2-factor of the minimum weight as a relaxation to the salesman problem within the branch and boundary method. Article [5] reports analysis of the approximate algorithm for solving the metric problem of a salesman with an accuracy estimate of 1.5, using the algorithm for obtaining the perfect matching combination.

The above works [1–5] suggest solving auxiliary subproblems in order to speed up calculations in accurate, and improve accuracy in approximate, methods for solving the problems on building closed routes, at the steps of the relevant algorithms. At the same time, the cited studies do not address issues related to better implementations or lower computational complexity of existing algorithms, which could speed up the construction of closed routes.

A subset of the graph's edges, which do not have common vertices, is termed the M matching [6]. An MP implies finding the $M_{\max}$ matching of maximal power (maximal matching) in a given graph $H=(V, E)$ with the set of vertices V and the set of edges U).

Paper [7] established that MPs belong to the class of effectively solvable tasks by proposing the algorithm to solving MP with a temporal assessment of complexity $O(n^4)$, $|V|=n$, when using a labor-intensive procedure for compressing certain odd cycles – flowers. Other known $M_{\max}$ derivation algorithms, whose authors are listed in [8, 9], differ from the algorithm described in [7] only by a better organization of memory and computation, while maintaining the difficult actions to detect and “cut” flowers. Work [10] gives the statement of an assignment problem with additional conflict constraints, which comes down to the task of finding the maximum perfect matching of minimum cost. The problem under consideration is solved by a labor-intensive algorithm, as well as a salesman problem's variant, which is considered in [11], that uses the search for matching in the graph. The complexity of solving the specified problems is related to the presence of flowers in the graph [8].

A flower is a simple cycle of odd length with $2k+1$ vertices, containing k matching edges [7]. Flowers are not included in a bipartite graph, therefore, for a bipartite graph the task on finding the maximum matching is significantly simplified. In addition, a flower in an arbitrary graph H is determined with respect to a certain fixed M matching as a subgraph with the maximum density of the edges forming a subset $M' \subseteq M$ [7]. Obviously, the less matching power at which activities start aimed at increasing it, the fewer flowers are found in H. If $|M|=1$, there are no flowers in H, or all the flowers are the cycles with three vertices and a common edge of M matching (buds).

The methods discussed in the above studies that are related to the construction of prolonged paths include additional steps to pack the flowers. This is because the algorithms listed in papers [3, 5, 7, 8] are performed on arbitrary graphs that allow the existence of odd-length cycles. Therefore, a transition to a bipartite graph would make it possible to avoid the evolution of flowers, and, as a result, could give an opportunity to improve the speed of the algorithm.

Our considerations suggest it must be a relevant idea worth considering to find the maximum matching in arbitrary graph $H=(V, U)$ by using the simpler structure of a bipartite graph $D=(X, Y, E)$. In a bipartite graph, D under X and Y denotes the sets of vertices, $|X|=|Y|=|V|=n$, E – the set of edges (i,j), $i \in X$, $j \in Y$. In D $(i,j) \in E$, if $\{v_i, v_j\} \in U$, $i \neq j$, $i,j \in \{1,2,\ldots,n\}$, $|E|=2|U|$, in H.

3. The aim and objectives of the study

The aim of this study is to develop an algorithm for solving an MP, which derives a solution to the MP over a time outperforming the existing algorithms for solving an MP.

To accomplish the aim, the following tasks have been set:

– to reduce the problem on matching, solved on an arbitrary graph, to a bipartite case and to justify the correctness of such a reduction;

– to devise and substantiate the linear procedure of building a prolonged path in a bipartite graph relative to the fixed matching;

– to suggest an algorithm for deriving the maximum matching in an arbitrary graph over a quadratic time.

4. Reducing the problem to a bipartite case

Designate $[h_{ij}]_n$ and $[d_{ij}]_n$ – the adjacency matrices of, respectively, graphs H and D: $h_{ij}=1$, if in H the vertex v_i is adjacent to the vertex v_j and $h_{ij}=0$ otherwise; $d_{ij}=1$, if the vertex $i \in X$ in D is adjacent to the vertex $j \in Y$ and $d_{ij}=0$ otherwise. It follows from the match between matrices $[h_{ij}]_n$ and $[d_{ij}]_n$ that if a solution to MP was built for D, the solution is built for H, too.

In graph H, the edge $\{v, u\}$ of the M matching is denoted $[v, u]$. In it, the vertex u is the partner of the vertex v. Edges that are not included in a matching are called free. The vertex, which belongs to the edge of the matching, is defined as saturated. The rest of the graph's vertices are termed unsaturated or free. The power of the maximum matching of graph H with n vertices is limited by the magnitude $\lfloor n/2 \rfloor$. A simple path in graph H is called alternating relative to M if the edges of the path through one are present in M [6–8]. An alternating path, which begins and ends with edges that do not belong to the M matching, is termed increasing relative to the M matching.

We shall define the arbitrary edge $\{v_i, v_j\}$ taken in H as the original matching $M=\{[v_i, v_j]\}$. Then, if $|V|=|U|=3$, then M is the solution to the MP. Assume $|V|=4$, and the graph H includes a single cycle of odd length. Obviously, in this case, the number of edges $|U|$ is minimal and equals 4. Graph $H_{\min}$ of four vertices and four edges, forming the odd cycle Z, is shown in Fig. 1, *a*). If $[v_i, v_j]\in Z$, $i, j\in\{1, 2, 3, 4\}$, then Z is a bud in graph $H_{\min}$ (Fig. 1, *b*). Otherwise, $M=\{[v_i, v_j]\}$ does not form a bud (Fig. 1, *c*). In Fig. 1, and all the following figures, the edges of matching are represented by thickened lines. It is clear from Fig. 1 that $M_{\max}$ is derived either by adding, to $M=\{[v_i, v_j]\}$, an edge that has no common vertices with edge $[v_i, v_j]$ (Fig. 1, *a*) or by building, from any free vertex (v_1 or v_4) (Fig. 1, *b*), the prolonged path relative to M.

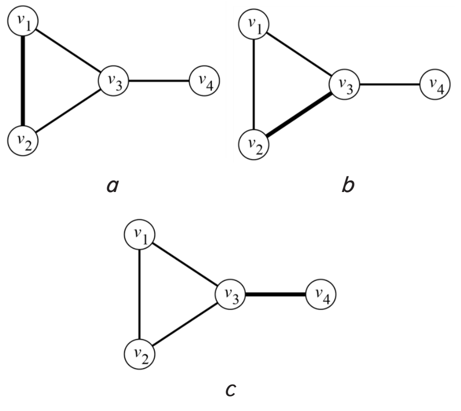

Fig. 1. Graph $H_{\min}$ of four vertices and four edges forming the odd cycle of Z: *a* — $M_{\max}$ is derived by adding, to $M=\{[v_i, v_j]\}$, an edge that has no common vertices with edge $[v_i, v_j]$; *b* — $[v_i, v_j]\in Z$, $i, j\in\{1, 2, 3, 4\}$, then Z is the bud in graph $H_{\min}$; *c* — $M=\{[v_i, v_j]\}$ does not form a bud

Each vertex $v_k\in V$ of graph H will be represented by a pair of vertices (i_k, j_k) of the bipartite graph $D=(X, Y, E)$, where $i_k\in X$ is the beginning of edge $(i_k, j_l)\in E$, and $j_k\in Y$ is the end of edge $(i_m, j_k)\in E$. Then, in D, any prolonged path relative to a fixed matching begins in some free vertex i_r and ends in some loose vertex $j_s, r\neq s$. In graph H, it is matched with a path from v_r to v_s. Obviously, any technique for building a maximal matching in the bipartite graph D, which is simultaneously a solution to the MP for arbitrary graph H, does not imply flower detection activities. Fig. 2, *a* shows graph $D_{\min}$, built based on graph $H_{\min}$, in which the matching M= $=\{[v_2, v_3]\}$ forms bud (v_1, v_3, v_2, v_1) (Fig. 1*b*). In $D_{\min}$, the bud is represented by a path from i_1 to j_1, composed of edges (i_1, j_3), $[i_2, j_3]$, (i_2, j_1). The beginning and end of the path, which increases the power $\{[i_2, j_3]\}$ by unity, is, respectively, the vertex i_4 and j_1. This path includes edges (i_4, j_3), $[i_2, j_3]$, (i_2, j_1), which determine the matching $\{[i_2, j_1], [i_4, j_3]\}$ (Fig. 2, *b*), which in graph $H_{\min}$ is matched with the maximal matching $\{[v_2, v_1], [v_4, v_3]\}$.

Thus, it has been shown that the problem on finding the maximal matching in arbitrary graph $H=(V, U)$ could be reduced to the problem on finding the maximal matching on the bipartite graph $D=(X, Y, E)$. In the bipartite graph D, X and Y denote the sets of vertices of graph D, $|X|=|Y|=|V|=n$, E is the set of edges (i, j), $i\in X$, $j\in Y$. In D $(i, j)\in E$, if $\{v_i, v_j\}\in U$, $i\neq j$, $i, j\in\{1, 2, \ldots, n\}$, $|E|=2|U|$ in H.

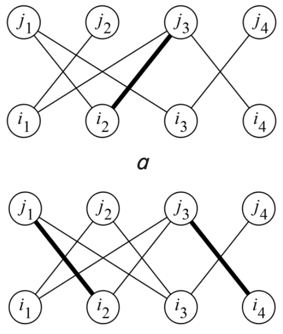

Fig. 2. Graph $D_{\min}$, built on graph: *a* — matching $M=\{[v_2, v_3]\}$ forms a bud (v_1, v_3, v_2, v_1); *b* — the beginning and end of the path that increases the power $\{[i_2, j_3]\}$ by unity, are respectively, vertices i_4 and j_1. This path includes edges (i_4, j_3), $[i_2, j_3]$, (i_2, j_1), which determine the matching $\{[i_2, j_1], [i_4, j_3]\}$

5. Justification of mutual match between matchings in arbitrary and bipartite graphs

Let in the bipartite graph $D=(X, Y, E)$, $|X|=|Y|=n\geq 6$ be the fixed matching $M(D)$, $|M(D)|\geq 2$, corresponding to the M matching in graph H. $M(D)$ transforms into a matching $M'(D)=M(D)\oplus P$ of power $|M'(D)|==|M(D)|+1$ as soon as there is a prolonged path p relative to $M(D)$, P is the set of edges along the path.

Statement 1. Each current matching □ in the arbitrary graph H is mutually unambiguously matched with a matching $M(D)$ in the bipartite graph D.

Proof. Suppose H contains a matching

$$M=\{[v_2, v_3], [v_4, v_5], \ldots, [v_{2k}, v_{2k+1}]\},$$

and the path $(v_1, v_2, v_3, \ldots, v_{2k}, v_{2k+1}, v_{2k+2})$ prolonged relative to M. Assume that D, built from H, includes no any alternating path $(j_1, j_2, j_3, \ldots, j_{2k}, j_{2k+1}, j_{2k+2})$. However, building graph D implies the inclusion of edges (i_1, j_2), $[i_2, j_3]$, ..., $[i_{2k}, j_{2k+1}]$, (i_{2k+1}, j_{2k+2}) and edges (i_2, j_1), (i_3, j_2), ..., (i_{2k+2}, j_{2k+1}), containing a combined prolonged path $(j_{2k+2}, j_{2k+1}, j_{2k}, \ldots, j_3, j_2, j_1)$ from the vertex i_{2k+2} to the vertex j_1 relative to matching

$$M(D)=\{[i_2, j_3], [i_4, j_5], \ldots, [i_{2k}, j_{2k+1}]\},$$

which corresponds to M in graph H. Proof is completed.

The proof of statement 1 indicates that none of the current solutions M in the arbitrary graph H is lost when moving to the iterative scheme for building $M_{\max}(D)$ in the bipartite graph D.

6. Development of a linear procedure for building a prolonged path in a bipartite graph relative to the fixed matching

Let the matching $M_k(D)=\{[i_l, j_l] \mid l=\overline{1,k}\}$, be built in graph D, where i_l is the beginning j_l is the end of the l-th edge, $k<\lfloor n/2\rfloor-1$, $i_l \neq j_l$, $I_k=\{i_l \mid l=\overline{1,k}\}$, $J_k=\{j_l \mid l=\overline{1,k}\}$ are the subsets of the saturated vertices of sets X and Y, respectively. In $M(D)$, denote via $\overline{j}_l \in Y$ the mapping of beginning i_l of edge $[i_l, j_l]$, $\overline{i}_l \in X$ is the mapping of end j_l of this edge, $(\overline{i}_l, \overline{j}_l)$, is the subset of mappings of vertices J_k,

$$\overline{J}_k=\left\{\overline{j}_l \middle| l=\overline{1,\ k}\right\}$$

is the subset of mappings of vertices I_k.

To determine whether the matching $M_k(D)$ is maximal or is converted into the matching $M_{k+1}(D)$, one finds the first free vertex $i_{k+1}\in X-\left(I_k\cup\overline{I}_k\right)$, which is considered to be the beginning of a prolonged path $p_{i_{k+1}}$ relative to $M_k(D)$ to any free vertex $j_s\in Y-\left(J_k\cup\overline{J}_k\cup\left\{\overline{j}_{k+1}\right\}\right)$. Here $\overline{j}_{k+1}$ is the mapping of i_{k+1}. If there is a path $p_{i_{k+1}}$,then

$$M_{k+1}(D)=M_k(D)\oplus P_{i_{k+1}}.$$

It is known that the matching M of graph H is maximum then and only when there is no prolonged path in H relative to M [6, 7]. Let the set $X-\left(I_k\cup\overline{I}_k\right)$ includes not a single vertex i_r that connects it to at least one of the vertices from the set $Y-\left(J_k\cup\overline{J}_k\cup\left\{\overline{j}_{k+1}\right\}\right)$ through the prolonged path in relation to matching $M_k(D)$. $M_k(D)$ is then maximal.

Designate via $D_{i_{k+1}}$ the subgraph of graph D, in which the matching $M_k(D)$ is fixed, all the vertices from the set $\overline{I}_k\cup\overline{J}_k$ are removed, and each edge connecting the vertex $\overline{j}_{k+1}$ with the vertex from I_k is removed (Fig. 3). It is required in subgraph $D_{i_{k+1}}$ to either to build a path $p_{i_{k+1}}$ or to establish that is absent in $D_{i_{k+1}}$.

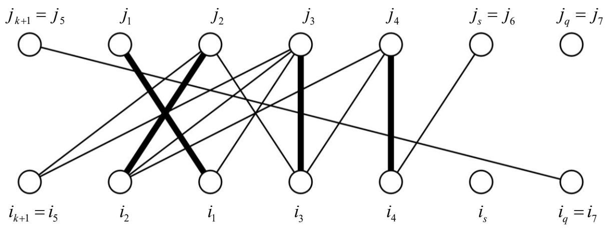

Fig. 3. Subgraph of graph D, in which matching $M_k(D)$ is fixed, all the vertices from set $\overline{I}_k\cup\overline{J}_k$ are removed, and each edge connecting the vertex $\overline{j}_{k+1}$ to the vertex from I_k is removed

We propose a PATH procedure for building $p_{i_{k+1}}$ in the subgraph $D_{i_{k+1}}$. The procedure is the modification of a known algorithm for finding paths from a given vertex to all achievable vertices of arbitrary graph H [6]. The PATH procedure consists of the following steps:

*S*0. Q is the set of marked vertices, R is the set of unmarked vertices; $Q=\{i_{k+1}\}$, $R=I_k$, $l=1$.

*S*1. If $i_l\in R$ and vertex j_l is incident to a vertex from Q, then put

$$Q=Q\cup\{i_l\},\quad R=R-\{i_l\};$$

if i_l is incident to vertex $j_s\in Y-\left(J_k\cup\left\{\overline{j}_{k+1}\right\}\right)$, then proceed to step *S*4.

*S*2. $l=l+1$.

*S*3. If $l>k$, it is over: the subgraph $D_{i_{k+1}}$ does not contain path $p_{i_{k+1}}$ otherwise proceed to step *S*1.

*S*4. If the vertex i_{k+1}, is adjacent to vertex j_s then it is over: $p_{i_{k+1}}=(i_{k+1}, j_s)$, otherwise the saturated vertex $i_m\in Q$; is adjacent to vertex j_s put $p_{i_{k+1}}=(i_m, j_s)$, $r=1$.

*S*5. While $i_l\in Q$ is adjacent to j_m and $i_l\neq i_{k+1}$ put

$$p_{i_{k+1}}=\left(i_l, j_m, p_{i_{k+1}}\right),\quad i_m=j_l,\quad j_m=j_l,\quad r=r+1.$$

*S*6. Put

$$p_{i_{k+1}}=\left(i_{k+1}, j_m, p_{i_{k+1}}\right).$$

Statement 2. The PATH procedure correctly builds in the subgraph $D_{i_{k+1}}$ the prolonged path p_{k+1} relative to the fixed matching of power k over time $O(k)$.

Proof. Step *S*3 determines that subgraph $D_{i_{k+1}}$ has no path $p_{i_{k+1}}$ from the vertex i_{k+1} to any free vertex $j_s\in Y-\left(J\cup\left\{\overline{j}_{k+1}\right\}\right)$, and step *S*4 finds edge (i_m, j_s) and the set of marked vertices

$$Q=(i_{k+1}, i_1,\ldots,i_m).$$

Assume $D_{i_{k+1}}$ does not contain $p_{i_{k+1}}$. Then the induction based on the number of cycle executions that forms steps *S*1–*S*3 establishes that $D_{i_{k+1}}$ has no vertex $j_s\in Y-\left(J\cup\left\{\overline{j}_{k+1}\right\}\right)$, incident to any vertex from Q.

On the other hand, let there be a found edge (i_m, j_s). in $D_{i_{k+1}}$ Then the induction for l yields a simple path $(i_{k+1}, j_1, i_1,\ldots, j_l, i_l,\ldots, i_m, j_s)$, which is built at step *S*5, from the vertex j_s to the vertex i_{k+1}.

The time of procedure execution is estimated based on that in the worst case the maximum number of vertices along path $p_{i_{k+1}}$ is k+1. To establish the existence of path $p_{i_{k+1}}$, one needs $O(k)$ elementary activities. The same amount of action will be needed to build it. Therefore, the time complexity of the PATH procedure is estimated by magnitude $O(k)$. The proof is complete.

Example 1. Let us illustrate work of the PATH procedure for subgraph D_{i_5}, depicted in Fig. 3. In D_{i_5} we defined the current matching

$$M_4(D)=\left\{[i_l, j_l] \mid l=\overline{1,4}\right\},\quad Q=\{i_5\},$$

$$R=\left\{i_l \mid l=\overline{1,4}\right\},\quad i_l\neq j_l.$$

Since the marked vertex i_5 is adjacent to vertices j_2 and j_3, then we mark any of the vertices via i_2 or i_3. Let it be the vertex i_2,

$$Q=\{i_5, i_2\},\quad R=\{i_1, i_3, i_4\}.$$

Vertex j_3 is incident to vertices i_2 and i_3, between which we shall choose, for example, i_3. Now,

$$Q=\{i_5, i_2, i_3\},\quad R=\{i_1, i_4\}.$$

The vertex j_4, adjacent to $i_4\in R$, is connected to vertices $i_2\in Q$ and $i_3\in Q$. After adding i_4 to the current set of marked vertices, we derive $Q=\{i_5, i_2, i_3, i_4\}$. But, (i_4, j_6) is such an edge that $j_6\in Y-\left(J_k\cup\left\{\overline{j}_{k+1}\right\}\right)$.

The path p_{i_5} from vertex i_5 to vertex j_6 is built as a result of execution of steps *S*4–*S*6. Since j_6 is adjacent to i_4, we put $p_{i_5}=(i_4, j_6)$, $r=1$. The vertex i_2 is adjacent to j_4, therefore $p_{i_5} = (i_2, j_4, i_4, j_6)$. The vertex i_3 is adjacent to j_2, therefore, $p_{i_5} = (i_3, j_2, i_2, j_4, i_4, j_6)$, $r = 3$. Since i_5 is incident to i_3, then the desired path $p_{i_5} = (i_5, j_3, i_3, j_2, i_2, j_4, i_4, j_6)$.

7. Algorithm for finding the maximal matching

Our considerations are detailed in the description of the following algorithm.

*S*0. The algorithm for finding in the arbitrary graph $H=(V, U)$ the maximal matching $M_{\max}$, $[h_{ij}]_n$ is the symmetrical adjacency matrices of graph *H*. The solution $M_{\max}$ is mutually unambiguously consistent with the maximal matching

$$M_{\max}(D) = \{[i_l, j_l] \mid i_l \neq j_l, l = \overline{1,k}\}$$

in a bipartite graph $D=(X, Y, E)$, in which *X*, *Y* are the sets of vertices, $|X| = |Y| = n \geq 4$, *E* is the set of edges, $\{i, j\} \in E$, if

$$\{v_i, v_j\} \in U,\ \ i \neq j,\ \ i, j \in \{1, 2, ..., n\},\ \ |E| = 2|U|.$$

*S*1. As a result of the application of heuristics with a temporal difficulty estimate not higher than $O(n^2)$, find in $[h_{ij}]_n$ the original matching

$$M_k = \{[v_{i_l}, v_{j_l}] \mid l = \overline{1,k}\},$$

where i_l is the number of the initial vertex, and j_l is the number of the final vertex of edge $[v_{i_l}, v_{j_l}]$, $i_l<j_l$.

$$M_k(D) = \{[i_l, j_l] \mid l = \overline{1,k}\}$$

is the matching in a bipartite graph *D* corresponding to M_k.

$$I_k = \{i_l \mid l = \overline{1,k}\},\ \ J_k = \{j_l \mid l = \overline{1,k}\},\ \ \overline{I_k} = \{j_l \mid l = \overline{1,k}\} \subset X,$$

$$\overline{J_k} = \{j_l \mid l = \overline{1,k}\} \subset Y,\ \ R = X - (I_k \cup \overline{I_k}),\ \ S = Y - (J_k \cup \overline{J_k}).$$

If $k = \lfloor n/2 \rfloor$, then $M_k(D)$ is the matching corresponding to $M_{\max}$ in *H*.

*S*2. Form a subgraph D_k of graph *D* that includes $M_k(D)$ and all free edges with initial vertices in I_k and final vertices in J_k; $r = 2k+1$.

*S*3. $k=k+1$.

*S*4. Identify the subgraph D_{x_r} of graph *D* by adding to D_{k-1} all the free edges linking the vertex $x_r \in R$ to vertices in J_k and all the free edges with the initial vertex in $\overline{I_k}$ and a final vertex in $S - \{y_r\}$, $y_r = \overline{x}_r$.

*S*5. Perform the PATH procedure to find in the subgraph D_{x_r} the prolonged path p_{x_r} relative to matching $M_{k-1}(D)$ to the vertex $j_s \in S - \{y_r\}$, achievable from the vertex x_r.

*S*6. If the path p_{x_r}, is built, then

$$M_k(D) = M_{k-1}(D) \oplus P_{x_r},$$

otherwise proceed to step *S*8.

*S*7. If $k < \lfloor n/2 \rfloor$, then

$$I_k = I_{k-1} \cup \{x_r\},\ \ J_k = J_{k-1} \cup \{j_s\},\ \ \overline{I}_k = \overline{I}_{k-1} \cup \{j_s\} \subset X,$$

$$j_s = \overline{i}_s,\ \ \overline{J}_k = \overline{J}_{k-1} \cup \{y_r\} \subset Y,\ \ y_r = \overline{x}_r,$$

$$R = R - \{x_r, j_s\} \subset X,\ \ S = S - \{j_s, y_r\} \subset Y,$$

where $x_r \in X$ is the beginning, and $j_s \in Y$ is the end of path p_{x_r}; proceed to step *S*2, otherwise $M_k(D)$ is the matching that corresponds to $M_{\max}$ in graph *H*.

*S*8. $R = R - \{x_r\}, r = r+1$; if $r \leq n$, proceed to step *S*4, otherwise $M_{k-1}(D)$ is the matching corresponding to $M_{\max}$ in *H*.

Theorem. The maximal matching in an arbitrary *n*-vertex graph *H* is correctly derived over time $O(n^2)$.

Proof. Because the bipartite graph *D* does not contain odd-length cycles, the reduction of MP in the arbitrary graph *H* to a bipartite case excludes flower detection and removal operations. By building, the subgraph D_{x_r} contains each edge corresponding in graph *H* to that that may belong to the prolonged path relative to matching $M_{k-1}(D)$, hence, if graph *H* contains M_{k-1}, then, after building at step *S*5 a path p_{x_r}, at steps *S*6–*S*7 we determine the matching $M_k(D)$.

Excluding a vertex x_r from the list of free vertices *R* when there is no p_{x_r}, in the subgraph D_{x_r} is based on the consequence from the theorem proven in [7]. If at some stage of the algorithm for solving MP there is no prolonged path from vertex *v*, then *v* cannot be considered as the starting vertex of the prolonged path at all other stages.

The algorithm terminates at $k \leq \lfloor n/2 \rfloor$ either after constructing $M_k(D)$, when $|R| \in \{0,1\}$, or after constructing $M_{k-1}(D)$, when there is no path p_{x_r} from each vertex $x_r \in R$. In the first case,

$$|M_k(D)| = \lfloor n/2 \rfloor,$$

in the second case, the necessary and sufficient condition is met under which the matching $M_{k-1}(D)$ is maximal.

Let us assess the time complexity of the algorithm presented.

Finding at step *S*1 in graph *H* the original matching M_k could be performed over time $O(n^2)$, for example, by choosing each time in the matrix $[h_{ij}]_n$ the element h_{ij} and by excluding the rows and columns with numbers *i* and *j*. The resulting matching leaves in graph *H* not a single edge incident to two free vertices. The bipartite graph *D*, which includes the matching $M_k(D)$, is formed over n^2 references to the elements of matrix $[h_{ij}]_n$. Therefore, step *S*1 is executed over time $O(n^2)$.

The starting phase for solving MP is step S2 that builds the subgraph D_k, in which the matching $M_k(D)$ corresponds to the original matching *M* in graph *H*. The first subgraph $D_{x_r}, x_r \in R$, derived from D_k, transforms to D_{k+1}, if D_{x_r} includes the path p_{x_r}. Building D_k requires no more than $k(k-1)$ comparison operations with elements from the matrix $[h_{ij}]_n$. To assess the laboriousness in solving MP, it would suffice to consider two extreme cases of the algorithm's termination.

In a first case, the algorithm terminates when the matching $M_k(D)$ is returned, corresponding to the original matching M_k in graph *H*. Then each subgraph $D_{x_r}, x_r \in R$, $|R| = n - 2k$, does not include the prolonged path p_{x_r} relative to $M_k(D)$. The subgraph D_{x_r} is determined by attaching to D_k not more than *k* edges (x_r, j_l), $j_l \in J_k$, and not more than $k(n-2k-1)$ edges (i_l, j_s), $i_l \in \overline{I_k}$, $j_s \in S - \{y_r\}$, $y_r = \overline{x}_r$. Consequently, it takes a time of $k(n-2k)^2+c(k+2)(n-2k)+k(k-1)$, $c<k$, to check the optimality of the initial solution $M_k(D)$. Since the labor-intensity of building $M_k(D)$ is limited by magnitude $O(n^2)$, then the time complexity of the algorithm in the considered case is estimated by the same magnitude.

In a second case, the algorithm converts the original matching $M_k(D)$ into a matching of power $\lfloor n/2 \rfloor > k$. Labor-intensity of building $M_{k+1}(D)$ is estimated by magnitude t_{k+1}, which includes $k(k-1)$ and $k+k(n-2k-1)$ elementary actions to form graphs D_k and D_{x_r}, respectively, as well as $c(k+2)$ operations to build path p_{x_r} and $c'(k+1)$ operations to execute step $S7$:

$$t_{k+1} = k(k-1) + k(n-2k) +$$
$$+c(k+2) + c'(k+1),$$

$$c' < k.$$

It is obvious that the construction of $M_{k+2}(D)$ could be done at labor-intensity

$$t_{k+2} = (k+1)(n-2(k+1)) +$$
$$+c(k+3) + c'(k+2),$$

and construction of $M_m(D)$, $k+3 \le m \le \lfloor n/2 \rfloor$, at labor-intensity

$$t_m = (m-1)(n-2(m-1)) + c(m+1) + c'm.$$

The total cost of converting the matching $M_k(D)$ into a matching of power $\lfloor n/2 \rfloor$ is estimated by magnitude

$$T = k(k-1) + \sum_{m=k+1}^{\lfloor n/2 \rfloor} t_m + c \sum_{m=k+1}^{\lfloor n/2 \rfloor} (m+1) + c' \sum_{m=k+1}^{\lfloor n/2 \rfloor} m,$$

where

$$k(k-1) < n^2,$$

$$\sum_{m=k+1}^{\lfloor n/2 \rfloor} t_m = \sum_{m=k}^{\lfloor n/2 \rfloor - 1} m(n-2m) \le (\lfloor n/2 \rfloor - k) \times$$
$$\times k(n-2k) = O(n^2),$$

$$c \sum_{m=k+1}^{\lfloor n/2 \rfloor} (m+1) < cn^2, \quad c' \sum_{m=k+1}^{\lfloor n/2 \rfloor} m < c'n^2.$$

Since the time to build the original matching $M_k(D)$ is limited by the magnitude $O(n^2)$ and $T=O(n^2)$, then in this case, too, the algorithm solves the MP at labor-intensity $O(n^2)$. The proof of the theorem is complete.

Example 2. Fig. 4, *a* shows the graph $H=(V, U)$, for which, by applying the algorithm from [7] set out in [6], the maximal matching was derived

$$M_{\max} = \left\{ \begin{matrix} [v_1, v_2], [v_3, v_4], [v_5, v_6], [v_7, v_{16}], \\ [v_8, v_9], [v_{10}, v_{17}], [v_{11}, v_{12}], [v_{13}, v_{14}] \end{matrix} \right\}, \ |M_{\max}| = 8.$$

The presented algorithm finds a matching of the same power (Fig. 4, *b*).

At step S1, a rapid heuristic builds the original matching in matrix $[h_{ij}]_{17}$

$$M_6 = \left\{ \begin{matrix} [v_2, v_3], [v_4, v_6], [v_8, v_9], \\ [v_{10}, v_{11}], [v_{12}, v_{13}], [v_{14}, v_{16}] \end{matrix} \right\};$$

$$M_6(D) = \left\{ \begin{matrix} [i_2, j_3], [i_4, j_6], [i_8, j_9], \\ [i_{10}, j_{11}], [i_{12}, j_{13}], [i_{14}, j_{16}] \end{matrix} \right\}$$

– corresponding matching in the bipartite graph D;

$$I_6 = \{i_2, i_4, i_8, i_{10}, i_{12}, i_{14}\},$$

$$J_6 = \{j_3, j_6, j_9, j_{11}, j_{13}, j_{16}\},$$

$$\bar{I}_6 = \{j_3, j_6, j_9, j_{11}, j_{13}, j_{16}\} \subset X,$$

$$\bar{J}_6 = \{i_2, i_4, i_8, i_{10}, i_{12}, i_{14}\} \subset Y,$$

$$R_1 = \{i_1, i_5, i_7, i_{15}, i_{17}\},$$

$$S = \{j_1, j_5, j_7, j_{15}, j_{17}\}.$$

$$k = 6 < \lfloor n/2 \rfloor = 8.$$

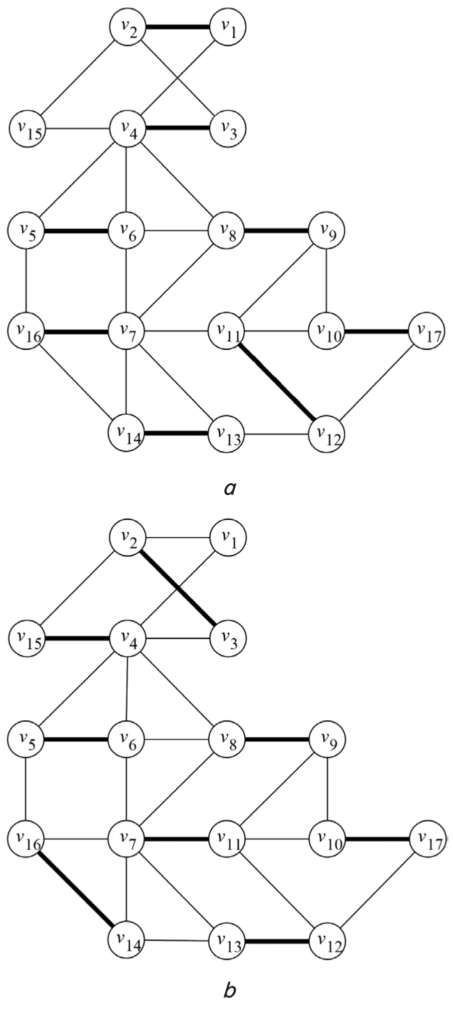

Fig. 4. Graph $H=(V, U)$: *a* — maximal matching derived by the Edmons algorithm; *b* — maximal matching derived by the proposed algorithm

At step $S2$, a subgraph D_6 is formed (Fig. 5), $r=13$, $k=7$. Step $S4$ builds a subgraph $D_{x_{13}}$, $x_{13} = i_1 \in R$, in which the vertex i_1 is isolated. Therefore, the PATH procedure establishes that there is no path from x_{13} to the vertex of the set $S - \{y_{13}\}$, $y_{13} = j_1$; $R = \{i_5, i_7, i_{15}, i_{17}\}$, r=14, in the subgraph

$D_{x_{13}}$. Now a subgraph D_{14} is built for the vertex $x_{14}=i_5$. D_{14} is shown in Fig. 6. The PATH procedure builds the prolonged path $P_{x_{14}}=p_{i_5}=(i_5,j_6,i_4,j_{15})$ relative to the matching $M_6(D)$ and, therefore, determines

$$M_7(D)=M_6(D)\oplus P_{i_5}=$$

$$=\left\{\begin{matrix}\left\{\begin{matrix}[i_2,j_3],\ [i_4,j_6],\ [i_8,j_9],\\ [i_{10},j_{11}],\ [i_{12},j_{13}],\ [i_{14},j_{16}]\end{matrix}\right\}-\\ -\{(i_5,j_6),\ [i_4,j_6],(i_4,j_{15})\}\end{matrix}\right\}\cup$$

$$\cup\left\{\begin{matrix}\{(i_5,j_6),\ [i_4,j_6],(i_4,j_{15})\}-\\ -\left\{\begin{matrix}[i_2,j_3],\ [i_4,j_6],\ [i_8,j_9],\\ [i_{10},j_{11}],\ [i_{12},j_{13}],\ [i_{14},j_{16}]\end{matrix}\right\}\end{matrix}\right\}=$$

$$=\left\{\begin{matrix}[i_2,j_3],\ [i_4,j_{15}],[i_5,j_6],\ [i_8,j_9],\\ [i_{10},j_{11}],\ [i_{12},j_{13}],\ [i_{14},j_{16}]\end{matrix}\right\}.$$

At step S7 $k=7<\lfloor n/2\rfloor=8$,

$$I_7=\{i_2,i_4,i_5,i_8,i_{10},i_{12},i_{14}\},$$

$$J_7=\{j_3,j_6,j_9,j_{11},j_{13},j_{15},j_{16}\},$$

$$\bar{I}_7=J_7\subset X,\quad \bar{J}_7=I_7\subset Y,\quad R=\{i_7,i_{17}\},\quad S=\{j_7,j_{17}\}.$$

Subgraph D_7 is shown in Fig. 7, $r=15$, and the subgraph $D_{x_{15}}$, $x_{15}=i_7$ – in Fig. 8; $k=8$.

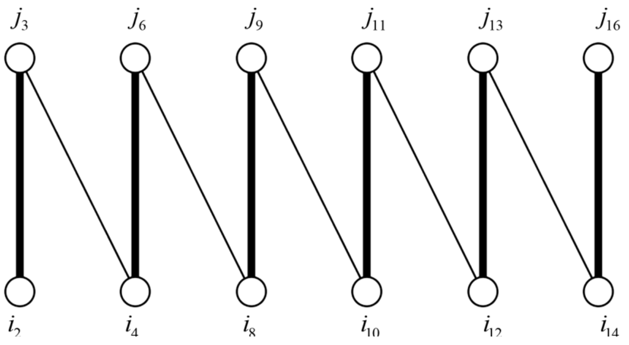

Fig. 5. Subgraph D_6 formed at step S2

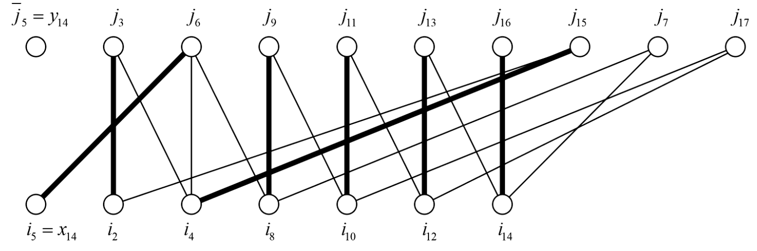

Fig. 6. Subgraph D_{14} for vertex $x_{14}=i_5$

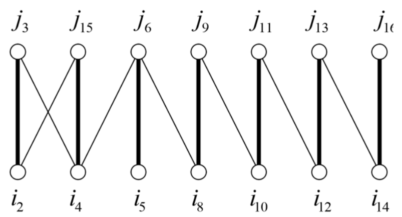

Fig. 7. Subgraph D_7

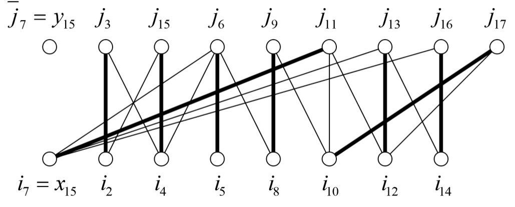

Fig. 8. Subgraph $D_{x_{15}}$, $x_{15}=i_7$

Thus, the PATH procedure builds, in the subgraph $D_{x_{15}}$, the prolonged path $p_{x_{15}}=(x_{15}=i_7,j_{11},i_{10},j_{17})$ relative to the matching $M_7(D)$ from the vertex i_7 to the vertex in the set $S=\{j_{17}\}$ and is derived

$$M_8(D)=\left\{\begin{matrix}[i_2,j_3],\ [i_4,j_{15}],[i_5,j_6],\ [i_7,j_{11}],\\ [i_8,j_9],\ [i_{10},j_{17}],\ [i_{12},j_{13}],\ [i_{14},j_{16}]\end{matrix}\right\},$$

coinciding with an accuracy to designations with the matching $M_{\max}=M_8$, shown in Fig. 4, *b*).

8. Discussion of results of constructing an algorithm with a quadratic time complexity to find the maximal matching

Known methods from [5–8] make it possible to derive a solution to the similar problem over time $O(n^4)$. The algorithm for solving the matching problem outlined in work [8] could be implemented as a computer software with a temporal complexity of $O(n^3)$ provided data structures are optimally utilized.

Existing algorithms from studies [5–8] are based on the execution of the algorithm for bypassing the graph in width, at the steps of which the detection and packaging of flowers are made (cycles of odd length).

The results reported in our article provide a solution to the matching problem over time $O(n^2)$. Winning in speed is achieved by moving from an arbitrary to a bipartite graph, on which there is no need to find and pack flowers. Thus, the algorithm that we proposed would provide for the acceleration in solving the problem by at least *n* times.

However, in practice, there is often a need to solve the problem on a weighted graph. In this case, one should talk about the problem of weighted matching. The approach proposed in this article cannot be applied directly to solve the problem of weighted matching. Therefore, we plan to advance the ideas proposed in the current paper to build an algorithm to solve the specified problem.

Solving a problem of weighted matching could be used as a lower estimate of the cost in accurate and approximate methods for finding Hamilton cycles on graphs and routing problems, which are reduced to the problem of a salesman, which would make it possible to bring down computational costs and speed up the time it takes to find solutions to these problems.

9. Conclusions

1. Reducing the problem on building the maximal matching, solved in an arbitrary graph, to the corresponding bipartite variant makes it possible to avoid the need to process flowers [7] and reduce the computational com-

plexity of the algorithm to solve an MP. By reducing the complexity of the algorithm, the speed of solving an MP is increased compared to known methods. The proof of mutually unambiguous correspondence between a matching in the arbitrary graph and the matching in a bipartite graph allows us to assert the correctness of the proposed reduction.

2. The proposed PATH procedure with a linear time assessment of complexity makes it possible to build a prolonged path in a bipartite graph relative to the current matching. The procedure greatly simplifies and, as a result, accelerates the derivation of a new matching, which has one edge more than the original.

3. Based on the iterative execution of the PATH procedure, we have proposed and substantiated an algorithm for building the maximal matching, which has a quadratic temporal complexity. Solving an MP by proposed algorithm as a relaxation within the branch and boundary method makes it possible to save computing resources of computers and servers used for computations and, as a result, to gain a win in the speed of building accurate solutions to certain NP-complete routing problems.

References

1. Toth, P., Vigo, D. (Eds.) (2014). Vehicle Routing: Problems, Methods, and Applications. SIAM. doi: https://doi.org/10.1137/1.9781611973594
2. Brusco, M. J., Stahl, S. (2005). Branch-and-Bound Applications in Combinatorial Data Analysis. Springer. doi: https://doi.org/10.1007/0-387-28810-4
3. Coste, P., Lodi, A., Pesant, G. (2019). Using Cost-Based Solution Densities from TSP Relaxations to Solve Routing Problems. Lecture Notes in Computer Science, 182–191. doi: https://doi.org/10.1007/978-3-030-19212-9_12
4. Matsiy, O. B., Morozov, A. V., Panishev, A. V. (2016). Fast Algorithm to Find 2-Factor of Minimum Weight. Cybernetics and Systems Analysis, 52 (3), 467–474. doi: https://doi.org/10.1007/s10559-016-9847-9
5. Zenklusen, R. (2019). A 1.5-Approximation for Path TSP. Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, 1539–1549. doi: https://doi.org/10.1137/1.9781611975482.93
6. Papadimitriu, H., Stayglits, K. (1985). Kombinatornaya optimizatsiya: Algoritmy i slozhnost'. Moscow: Mir, 510.
7. Edmonds, J. (1965). Paths, Trees, and Flowers. Canadian Journal of Mathematics, 17, 449–467. doi: https://doi.org/10.4153/cjm-1965-045-4
8. Lovas, L., Plammer, M. (1998). Prikladnye zadachi teorii grafov. Teoriya parosochetaniy v matematike, fizike, himii. Moscow: Mir, 653.
9. Sharifov, F. A. (2008). Sovershennye parosochetaniya i rasshirenniy polimatroid. Kibernetika i sistemniy analiz, 3, 173–179.
10. Öncan, T., Şuvak, Z., Akyüz, M. H., Altınel, İ. K. (2019). Assignment problem with conflicts. Computers & Operations Research, 111, 214–229. doi: https://doi.org/10.1016/j.cor.2019.07.001
11. Naser, H., Awad, W. S., El-Alfy, E.-S. M. (2019). A multi-matching approximation algorithm for Symmetric Traveling Salesman Problem. Journal of Intelligent & Fuzzy Systems, 36 (3), 2285–2295. doi: https://doi.org/10.3233/jifs169939