Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Mathematical model for a cutting path avoiding intersections ⋆

Tatiana A. Makarovskikh * **Anatoly V. Panyukov** **

* *South Ural State University, pr. Lenina, 76, Chelyabinsk, 454080, Russian Federation (e-mail: Makarovskikh.T.A@susu.ru).*
** *South Ural State University, pr. Lenina, 76, Chelyabinsk, 454080, Russian Federation (e-mail: paniukovav@susu.ru).*

Abstract: During the technological preparation of the cutting process, it is necessary to determine the path of the cutter when there are no self-intersections of the cutting path and the part cut off from the sheet does not require any cuts, then arise the problems: first, representation of the cutting plan as a planar graph which is the homeomorphic image of the cutting plan; second, algorithms to find the cutter routes in this graph. The paper is devoted to a polynomial-time algorithm for constructing a non-intersecting ordered enclosing chain (NOE-chain) for a plane Eulerian graph. The proposed approach consists in splitting of all original graph vertices with degree higher than 4 by introducing fictive vertices and edges and, thus, reducing the considered earlier problem to the problem of finding an A-chain with ordered enclosing in a plane connected 4-regular graph. A test example of constructing NOE-chain with ordered enclosing is considered.

Keywords: CAD/CAM, Graph theoretic models, Graph theory, Path planning, Routing algorithms, Software performance

1. INTRODUCTION

During the technological preparation of the cutting process, it is necessary to determine the path of the cutter when there are no self-intersections of the cutting path and the part cut off from the sheet does not require any cuts, then arise the problems: first, representation of the cutting plan as a planar graph which is the homeomorphic image of the cutting plan; second, algorithms to find the cutter routes in this graph.

The paper by Silva et al. (2019) is devoted to the CPDP (Cutting Path Determination Problem), which consists in determining the optimal path for cutting according to a given cutting plan with one or more tools. The authors assume that there are two obvious restrictions: 1) all parts must be cut out; 2) none of the cut out parts should require further cuts, i.e. OE (Odered Eclosing) constraint by Makarovskikh et al. (2015) is fulfilled. To solve the CPDP problem, more detailed statements are known: GTSP (General Traveling Salesman Problem, see Xie et al. (2009); Jing et al. (2013); Dewil et al. (2015, 2016); Hoeft et al. (1997); Dewil et al. (2014); Petunin et al. (2016); Chentsov et al. (2018); Khachay et al. (2018); Chentsov et al. (2016)), CCP (Continuous Cutting Problem Point), ECP (Endpoint Cutting Problem), see Manber et al. (1984); Lee et al. (2006), and ICP (Intermittent Cutting Problem, see Makarovskikh et al. (2015)). Note that ECP and ICP allow the combination of the parts borders, which reduces material waste, cutting length and idle lengths (see Dewil et al. (2015)). The problems of reducing material waste and maximizing the combination of the contours fragments of the cut out parts are solved at the stage of the cutting plan design.

Despite the noted advantages of the computer technologies ECP and ICP most publications are currently devoted to the development of GTSP and CCP technologies, which use obvious cutting path algorithms consisting in contour-by-contour cutting.

The development of ECP and ICP computer technologies are considered, for example,in papers by Makarovskikh et al. (2015); Manber et al. (1984); Panyukova (2007). The polynomial algorithms for OE routing (when the part cut off from a sheet does not require further cuts) are given there.

For industrial enterprises related by their activity to the tasks of cutting sheet material, there is a need to use CAD/CAM systems for the technological preparation of cutting processes. Taking into account the capabilities of modern equipment for cutting parts from sheet material allows you to make cutting plans that allow combining the contours of the cut parts, which reduces material waste, cutting length, and the number of idle passes. Algorithms for cutting plans design for tasks that allow combination of cuts do not fundamentally differ from algorithms that do not allow combination. However, the algorithms for finding the paths of the cutting tool movement are fundamentally different. Therefore, the development of algorithms for finding the route of the cutting tool for cutting plans that allow the combination of the contours of the cut parts is an open task.

⋆ The work was supported by Act 211 Government of the Russian Federation, contract No. 02.A03.21.0011.

Our research is devoted to routing problems in plane graphs, which are homeomorphic images of cutting plans. Makarovskikh et al. (2019) give the description of the format used for representation of the cut-out parts and a cutting plan. The route covering all the borders of the cut parts determines **the path of the cutting tool**. The technological constraint is the absence of intersection of the internal faces of any route initial part with the edges of its remaining part. When constructing manipulator control systems using an undirected graph as a model of cutting plan we may display various elements of the manipulator trajectory by it. In this case, problems of constructing routes that satisfy various constraints arise.

As it was mentioned before, the most used approach is one not involving the combination of the contours of the cut parts. This method is material intensive and energy consuming.

If the cutting plan represents a plane Euler graph, then it is known that its dual face graph is bichromatic. For this case, Bely (1983) proved the existence of an Euler cycle homeomorphic to a plane Jordan curve without self-intersections. However, the possibility of using this result for CAD/CAM systems for the technological preparation of cutting processes remained unclear. The algorithm proposed by Bely (1983) paper is polynomial. Subsequently, Manber et al. (1987) gave the proof of the $\mathcal{NP}$-completeness of this problem. Further Fleischner (1990) introduced the concept of a A-chain in which the allowed transitions between edges are given in a cyclic order at each vertex of the graph. He also showed that the problem of determining the A-chain is $\mathcal{NP}$-hard in the general, he presented some particular cases for which the problem is solvable in polynomial time. One such special case is a 4-regular graph.

Attempts to construct routes in which the covered part does not cover edges that have not yet been completed were made by Manber et al. (1984). A mathematical statement of this problem in terms of OE-chains was given by Panyukova (2007), however, the OE-chain allows the possibility of self-intersections of the trajectory. The construction of non-intersecting OE-chains (NOE-chains) in plane Eulerian graphs has the great theoretical and practical value.

We say a chain obtained by splitting the vertices of the original graph is non-intersecting one if it homeomorphic image is a plane Jordan curve without self-intersections. Obviously, AOE-chain is non-intersecting one. Makarovskikh et al. (2019) propose the polynomial algorithm for constructing AOE-chain for the particular case when the cutting plan is homeomeric to a plane connected 4-regular graph.

In this paper we propose a polynomial algorithm for constructing a non-intersecting chain in a plane connected Eulerian graph. The algorithm proposed here solves the routing problem when two technological restrictions are fulfilled: the part cut from the sheet does not require additional cuts (some papers on this problem are Panyukova (2014); Makarovskikh et al. (2016, 2015)) and there are no intersections in the cutting path (the first announcement of these results is made in Makarovskikh et al. (2017)).

Next section 2 contains the necessary definitions and describes the notation used to represent the data. Section 3 considers a class of non-intersecting OE-chains (or NOE-chains). It is shown that the problem to construct NOE-chain in a plane connected Eulerian graph can be reduced in polynomial time to the problem of constructing the AOE-chain for a plane connected 4-regular graph. An algorithm for such reduction is given. Section 4 considers a test example of constructing the NOE-chain. In conclusion, the results obtained in the work are listed.

2. MAIN DEFINITIONS AND DESIGNATIONS

In this paper we use the graph representation from earlier works (see Makarovskikh et al. (2019, 2016, 2017); Panyukova (2014); Makarovskikh et al. (2015)). Instead of data cutting plan we use it homeomorphic image which is a plane graph G with outer face f_0 on plane S. Let for any part J of graph G (i.e. $J \subseteq G$) $\mathsf{Int}\,(J)$ designates the set-theoretic union of its inner faces (union of all connected components $S \setminus J$ without outer face). If J is an initial part of the rote then $\mathrm{Int}(J)$ may be interpreted as a parts cut off a sheet.

The topological representation of plane graph G on plane S up to homeomorphism can be defined by the following eight functions for each edge $e \in E(G)$: $v_k(e)$, $k = 1, 2$ be the vertices incident to e; $l_k(e)$, $k = 1, 2$ be the edges obtained by rotating of e counter clockwise around $v_k(e)$; $r_k(e)$, $k = 1, 2$ be the edges obtained by rotating e clockwise around $v_k(e)$; $f_k(e)$ be a face at right when we move by e from $v_k(e)$ to $v_{3-k}(e)$, $k = 1, 2$ (see Makarovskikh et al. (2016, 2015)).

In fact, graph representation gives the orientation of its edges. Later we suppose that we move by edge from vertex $v_1(e)$ to vertex $v_2(e)$. As soon as we do not know the direction of passing an edge when we code graph G then we need a function changing the indices of functions $v_k(e)$, $r_k(e)$ $l_k(e)$, $f_k(e)$, $k = 1, 2$ for some edges. Function `REPLACE` makes it for the presented algorithms. Its aim is changing the indices of functions $v_k(e)$, $l_k(e)$, $r_k(e)$, and $f_k(e)$ to $3 - k$, $k = 1, 2$ (see Makarovskikh et al. (2016)).

Let's assume that off the considered plane graphs are represented by these functions. The space complexity of this representation is $O(|E(G)|\cdot\log_2|V(G)|)$ (see Makarovskikh et al. (2019)). In further we wil use the following definitions from papers Panyukova (2007); Szeider (2003); Fleischner (1990). Here are the main ones for the readers convenience.

Definition 1. (Panyukova (2007)). We say that cycle $C = v_1e_1v_2e_2\dots v_k = v_1$ of Eulerian graph G has ordered enclosing (or being OE-trail for short) if for any its initial part $C_i = v_1e_1v_2e_2\dots e_i$, $i \le |E(G)|$ condition

$$\mathrm{Int}\,(C_i) \cap G = \emptyset$$

holds.

Definition 2. (Fleischner (1990)). Eulerian chain T is called **A-chain** if sequent edges of T incident to vertex v be the neighbours in cyclic order $O^{\pm}(v)$.

Definition 3. Let **rank of edge** $e \in E(G)$ be called a value of function $\mathrm{rank}(e) : E(G) \to N$ recursively defined:

- let $E_1 = \{e \in E : e \subset f_0\}$ be a set of edged bounding outer face f_0 of graph $G(V, E)$ then $(\forall e \in E_1)\,(\mathrm{rank}(e) = 1)$;

Algorithm 1 NOE-CHAIN (G)

Require: plane Eulerian graph G, presented by functions $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$ rank(e);
Ensure: NOE-chain $T(G)$;
1: $\widehat{G}$ = NonIntersecting(G);▷ Split vertices of degree greater than 4
2: $\widetilde{G}$ = CutPointSplitting$(\widehat{G})$; ▷ Split cut-vertices at each rank
3: C^*=AOE_TRAIL$(\widetilde{G})$; ▷ Obtain AOE-chain for graph $\tilde{G}$
4: C=Absorb(C^*); ▷ Absorb all splitted vertices

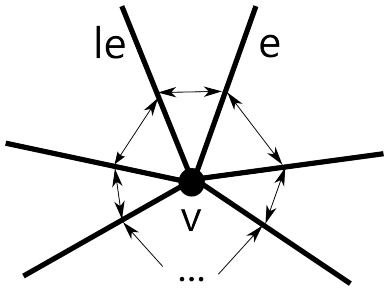

Fig. 1. The initial pointers to the neighbouring edges of the splitting vertex

- let $E_k(G)$ be a set of edges having rank = 1

$$G_k\left(V, E\backslash\left(\bigcup_{l=1}^{k-1} E_l\right)\right)$$

then $(\forall e \in E_k)\,(\text{rank}(e) = k)$.

Edge rank defines the remoteness of an edge from the outer face and shows the minimal number of faces to be passed to reach this edge from the outer face f_0.

Definition 4. (Makarovskikh et al. (2015)). Let **rank of face** $f \in F(G)$ be the value of function rank : $F(G) \to \mathrm{Z}^{\geq 0}$:

$$\text{rank}(f) = \begin{cases} 0, & \text{if } f = f_0, \\ \min\limits_{e \in E(f)} \text{rank}(e), & \text{otherwise,} \end{cases}$$

where $E(f)$ be the set of edges incident to face $f \in F$.

3. *NOE*-CHAINS CONSTRUCTING ALGORITHM

The considered by Makarovskikh et al. (2019) class of AOE-chains is rather narrow. Moreover, there are no known effective algorithms for constructing of these chains. As a matter of fact, for practice it is enough to obtain non-intersecting OE-chains in spite of AOE-chains.

Definition 5. Let Eulerian cycle C of plane graph G be non-intersecting if it is homeomorphic to a closed Jordan curve without intersections obtained from graph G by applying of $O(|E(G)|)$ splittings of its vertices.

To obtain the non-intersecting Eulerian OE-chain (or cycle) for plane connected Eulerian graph (in further we call whis chain as NOE-chain) we may use the following algorithm 1.

Function `Non-intersecting (G)` (algorithm 2) constructs the 4-regular graph $\widehat{G}$ by splitting all vertices $v \in V(G)$ of degree $d = 2l$ $(l \geq 3)$ to l fictive vertices of degree 4 and adds l fictive edges incident to vertices obtained by splitting and forming the cycle (see fig. 1 and fig.2).

Algorithm 2 Function `Non-intersecting (G)`

Require: plane Eulerian graph G presented by functions $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$, and rank(e);
Ensure: plane connected 4-regular graph G^* defined the same way;
1: **for all** $v \in V(G)$ **do** ▷ Initialization of $Checked(v)$ function
2: $Checked(v) :=$ **false**;
3: **end for**
4: **for all** $(e \in E(G)\ )$ **do** ▷ Search $v : d(v) > 4$ and split it them
5: $k := 1$; ▷ Consider vertex with $k = 1$, then $k = 2$
6: **while** $(k \leq 2)$ **do**
7: **if** (! $Checked(v_k(e))$) **then** ▷ Is it unprocessed earlier?
8: **if** $(k = 2)$ **then** ▷ Correct the indices
9: REPLACE(e); ▷ later only $v_1(e)$ handles
10: **end if**
11: Handle (e); ▷ Run the function for handling $v_1(e)$
12: $Checked(v_1(e)) :=$ **true**; ▷ Mark vertex as viewed
13: **end if**
14: $k := k + 1$;
15: **end while**
16: **end for**
End of function

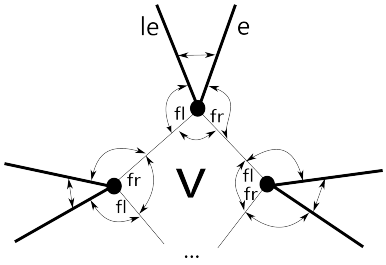

Fig. 2. Splitting the vertex (bold lines show the edges of G, and the thin ones show the fictive ones) and modifying the pointers according to splitting

For all considered modifications we need to look through the functions $v_k(e)$, $k = 1, 2$ for all edges $e \in E(G)$, and modify the graph encoding system. To achieve this goal we need boolean function

$$\text{Checked}(v) = \begin{cases} \texttt{true}, & \text{if vertex is handled;} \\ \texttt{false}, & \text{otherwise} \end{cases}$$

on set $V(G)$. At initialization stage (lines 1–3 of algorithm 2) all vertices are said to be not checked, i. e. Checked(v) = `false` for all $v \in V(G)$. Handling the vertex $v = v_1(e)$ so that Checked(v) = `false` consists in running the procedure `Handle (`e`)` (algorithm 3).

Algorithm 3 during cycle repeat–until (lines 6–11) counts the degree d of current vertex v. If $d > 4$, then the second cycle repeat–until (lines 12–23) runs. IN this cycle the handled vertex is splitted to $d/2$ fictive vertices, d fictive edges incident to these vertices and forming a cycle are introduced.

Let us admit that lines 18–23 are dealing not only with changing the pointers to edges but also create a new (fictive) face F, incident to all fictive vertices and edges, and also define the ranks of fictive edges.

Definition 6. The rank of fictive edge (line 20) is equal to rank of initial graph face incident to the entered fictive edge.

After obtaining plane connected 4-regular graph $\widehat{G}$ with defined ranks of fictive edges and introduced fictive faces we may consequentially run:

Algorithm 3 Procedure Handle (e)

```
1: procedure HANDLE(e)
2:     v := v_1(e);                                  ▷ Splittable vertex
3:     e_first := e;                                 ▷ Save the first considered edge
4:     d := 0;                                       ▷ Initialization of degree counter d
5:     F := FaceNum() + 1;                           ▷ Define the number of a new face
6:     repeat                                        ▷ Pass 1: Defining the degree of v
7:         le := l_1(e);
8:         if (v_1(le) ≠ v) then REPLACE(le);
9:         end if                                    ▷ Change the indices if necessary
10:        e := le;                                  ▷ Take the edge into account
11:        (d := d + 1;)                             ▷ and go to the next one
12:    until (e = e_first);                          ▷ All edges incident to v are viewed
13:    if (d > 4) then                               ▷ If the degree of vertex is greater than 4
14:        e := e_first;                             ▷ Begin from the first considered edge
15:        le := l_k (e);                            ▷ Define its left neighbour
16:        e_next := l_k (le);                       ▷ Save the edge for the next iteration
17:        fl := new EDGE;                           ▷ Add the fictive edge adjacent to le
18:        (fle := fl; e_first := e;)
19:        repeat                                    ▷ Put the edge pointers
20:            e := e_next; le := l_k (e); fr := fl;
21:            f_1(fl) := F; f_2(fl) := f_2(e);
22:                          ▷ Define the faces incident to a fictive edge
23:            rank(fl) := facerank(f_2(fl));
24:                                  ▷ Define "rank" of fictive edge
25:            Function facerank()
26:                ▷ defines the face rank according to the definition
27:            fl := new EDGE; e_next := l_k (le);
28:        until (l_k(le) = e_first);
29:    end if
30: end procedure
```

- algorithm CUT-POINT-SPLITTING($\widehat{G}$) for obtaining graph $\widetilde{G}$ with splitted cut-vertices of each rank;
- algorithm AOE-TRAIL($\widetilde{G}$) (see Makarovskikh et al. (2019)) for constructing the AOE-chain C^* for graph $\widetilde{G}$.

When constructing a chain C^* algorithm AOE-TRAIL($\widetilde{G}$) having opportunity to take one of two adjacent unpassed edges of the same rank firstly takes the fictive edge to guarantee the fulfilment of ordered enclosing condition.

Procedure Absorb(*) replaces all the fictive edges and their incident vertices (obtained while splitting v) from * (runs the operation of absorbing the fictive vertices). As a result of running this procedure we obtain NOE-chain C for initial graph G. Chain C obtained after deleting the fictive edges and absorbing the fictive vertices belongs to class OE as soon as procedure of edges deleting does not vanish the order of the remaining edges of a chain. That excludes the appearance of a cycle covering still not passed edges.

As soon as procedure Handle consists of two consecutive scans of incident to current vertex v edges then itscomputing complexity is equal to $O(|E(G)|)$. Function Non-Intersecting consists in a single scan of all edges, that is, its computational complexity is also $O(|E(G)|)$. Hence, algorithm for reducing a plane connected Eulerian graph to a plane connected 4-regular graph solves the problem in time $O(|E(G)|^2)$. Since the algorithm AOE-TRAIL and function CUT-POINT-SPLITTING Makarovskikh et al. (2019) preceding its call solve the problem of obtaining AOE-chain C by time $O(|E(G^*)| \cdot \log_2 |V(G^*)|)$, then the problem of NOE-chain constructing for graph G is solved by polynomial time $O(|E(G)|^2)$.

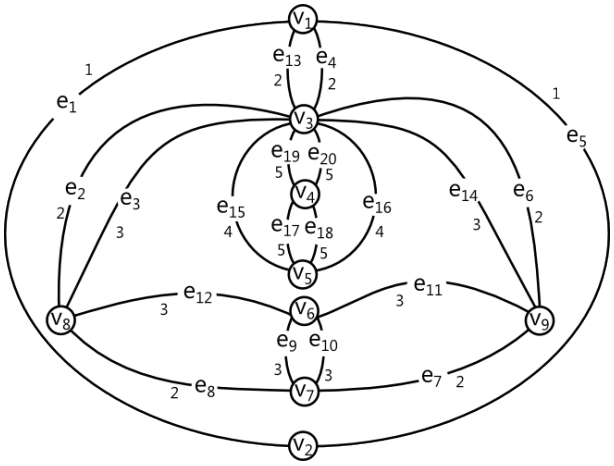

Fig. 3. The initial graph for the example of NOE-chain constructing with a vertex of degree greater than 6 not incident to outer face

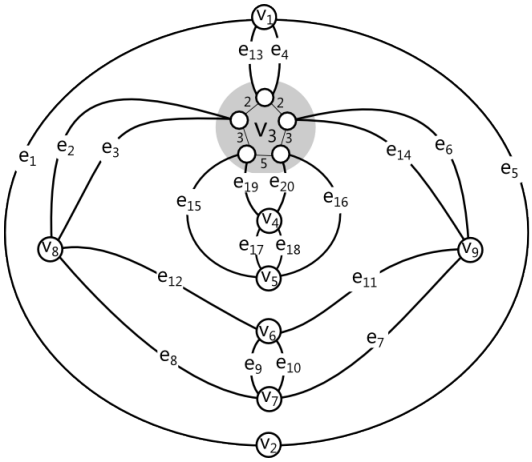

Fig. 4. Obtained 4-regular graph $\widehat{G}$

The above proves the following theorem.

Theorem 7. Algorithm NOE-CHAIN solves the problem of constructing the NOE-chain for plane Eulerian graph by time $O(|E(G)|^2)$.

4. EXAMPLE

Let us consider the running of algorithm on the example pf graph in figure 3. This example involves the common case when graph has a vertex of degree 6 or more not incident to outer face and also graph has some cut-vertices (and receives some after splitting process) of different ranks. This graph is Eulerian, hence, the constructing of NOE-chain may start from any vertex incident to outer face. Let it be v_2.

After procedure Handle() running we get graph in figure 4. As it is easy to see all vertices of this graph have degree 4. Graph in figure 4 besides the split vertex v_3 has cut-vertices v_4 and v_6 of ranks 3 and 2 correspondingly, and also cut-vertex of rank 2 in the split vertex v_3 incident to edges e_{13}, e_4, and two fictive edges of the same rank. These vertices are to be split by algorithm CUT-POINT-SPLITTING. The result of running this algorithm is shown in figure 5. Algorithm AOE-Trail allows to define AOE-chain for the obtained graph. The symbol

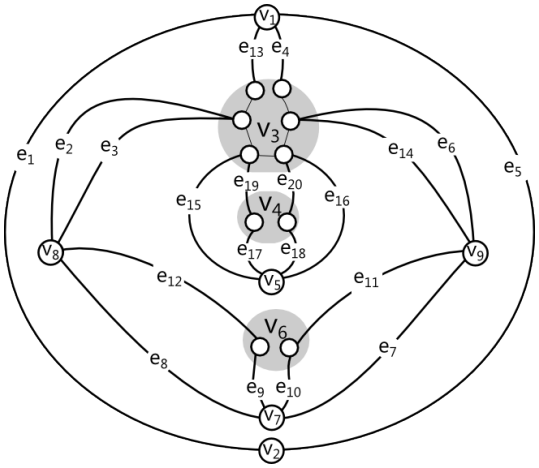

Fig. 5. 4-regular graph $\widetilde{G}$ without cut-vertices at any rank

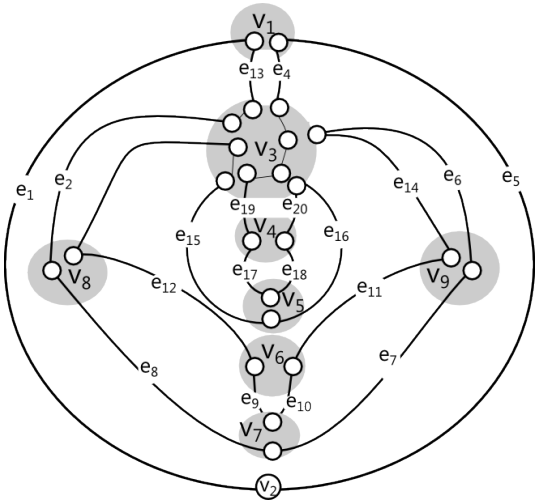

Fig. 6. The resulting non-intersecting OE-chain C

"*" means the transition by fictive edge(s). The chain is following

$$T^* = v_2e_5v_1e_4v_3 * * * e_{19}v_4e_{17}v_5e_{18}v_4e_{20}v_3e_{16}v_5e_{15}v_3 * e_3v_8e_{12}v_6e_9v_7e_{10}v_6e_{11}v_9e_{14}v_3e_6v_9e_7v_7e_8v_8e_2v_3 * e_{13}v_1e_1v_2,$$

After absorbing the split vertices we get the following NOE-chain for initial graph (see figure 6):

$$T^* = v_2e_5v_1e_4v_3e_{19}v_4e_{17}v_5e_{18}v_4e_{20}v_3e_{16}v_5e_{15}v_3 e_3v_8e_{12}v_6e_9v_7e_{10}v_6e_{11}v_9e_{14}v_3e_6v_9e_7v_7e_8v_8 e_2v_3e_{13}v_1e_1v_2,$$

5. CONCLUSION

The algorithm proposed in the paper solves the routing problem for cutting parts, when the following technological restrictions are simultaneously imposed on the path of the tool: (1) the part cut off from the sheet does not require additional cuts (OE-routes), (2) there are no self-intersections of the cutting path (NOE-chains).

The presented algorithm allows to obtain a NOE-chain in a plane Eulerian graph. In the case of a plane non-Eulerian (in the general case a graph is disconnected) graph G without end-vertices, it is necessary to split all vertices of degree higher than 4 in accordance with algorithm 3. As a result, we obtain a graph having the degrees of vertices are 3 or 4 (w.l.o.g. vertices of degree 2 are not considered). For this graph, the same sequence of actions is applicable as described by Makarovskikh et al. (2019) for constructing the AOE-cover. The chains of the resulting covering do not contain any fictive edges, absorbing together all the split vertices. As a result, we obtain NOE-covering.

REFERENCES

Silva E.F., Oliveira L.T., Oliveira J.F., Toledo F.M.B. Exact approaches for the cutting path determination problem // Computers & Operations Research. 2019. Vol. 112, 104772. DOI: 10.1016/j.cor.2019.104772

Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAD Technological Preparation of Cutting Processes // Automation and Remote Control. 2017. Vol. 78, No. 4. P. 868–882.

Xie S. Q., et al. Optimal process planning for compound laser cutting and punch using Genetic Algorithms. // International Journal of Mechatronics and Manufacturing Systems. 2009. Vol. 2 (1/2). P 20-38.

Jing Y., Zhige C. An Optimized Algorithm of Numberical Cutting-Path Control in Garment Manufacturing. // Advanced Materials Research. 2013. Vol. 796. P. 454-457.

Dewil R., Vansteenwegen P., Cattrysse D. A Review of Cutting Path Algorithms for Laser Cutters // International Journal Adv Manuf. Technol. 2016. Vol. 87, P. 1865–1884. DOI: 10.1007/s00170-016-8609-1

Dewil R., Vansteenwegen P., Cattrysse D., Laguna M., Vossen T. An Improvement Heuristic Framework for the Laser Cutting Tool Path Problem // International Journal of Production Research. 2015. Vol. 53, Issue 6, P. 1761–1776. DOI: 10.1080/00207543.2014.959268

Hoeft J., Palekar U. Heuristics for the Plate-cutting Travelling Salesman Problem. // IIE Transactions. 1997. Vol. 29(9). P. 719–731.

Dewil R., Vansteenwegen P., Cattrysse D. Construction Heuristics for Generating Tool Paths for Laser Cutters. // International Journal of Production Research. 2014. Vol. 52(20). P. 5965–5984.

Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme // Bulletin of South Ural State University. Series: Mathematical Modelling and Programming. 2018. Vol. 11, No. 1. P. 60–74. DOI: 10.14529/mmp180106

Petunin A., Stylios C. Optimization Models of Tool Path Problem for CNC Sheet Metal Cutting Machines. // 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016 Troyes, France, 28–30 June 2016. IFAC-PapersOnLine. 2016. Vol. 49. P. 23–28.

Chentsov A., Khachay M., Khachay D. Linear Time Algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem // 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016, Troyes, France, 28–30 June 2016. IFAC-PapersOnLine. 2016. Vol. 49. P. 651–655.

Khachay M., Neznakhina K. Towards Tractability of the Euclidean Generalized Travelling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height // Communications in Computer and Information Science. 2018. Vol. 871. P. 68–77.

Manber U., Israni S. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting // J. Manuf. Syst. Vol 3(1). 1984. P. 81–89. DOI: 10.1016/0278-6125(84)90024-4

Lee M. K., Kwon K. B. Cutting path optimization in CNC cutting processes using a two-step genetic algorithm. // International Journal of Production Research. 2006. Vol. 44(24). P 5307-5326.

Panyukova T. Chain Sequences with Ordered Enclosing // Journal of Computer and System Sciences International. 2007. Vol. 46, No. 1(10). P. 83–92. DOI: 10.1134/S1064230707010108

Makarovskikh T., Panyukov A., Savitsky E. Software Development for Cutting Tool Routing Problems // Procedia Manufacturing. 2019. Vol. 29. Pp. 567–574.

Bely S.B. On self-disjoint and disjoint chains // Matematicheskie Zametki, 1983. Vol. 34. No. 4. P. 625–628.

Makarovskikh T.A., Panyukov A.V. Software for Constructing A-chains with Ordered Enclosing for a Plane Connected 4-regular Graph // IFAC-PapersOnLine. 2019. Vol. 52. Iss. 13. Pp. 2650–2655. DOI: 10.1016/j.ifacol.2019.11.607

Manber U., Bent S.W. On Non-intersecting Eulerian Circuits // Discrete Applied Mathematics. 1987. Vol. 18. P. 87–94. DOI: 10.1016/0166-218X(87)90045-X

Fleischner H. Eulerian Graphs and Related Topics. Part 1, Vol.1.: Ann. Discrete Mathematics, 1990. 45.

Panyukova T.A. Constructing of OE-postman Path for a Planar Graph // Bulletin of South Ural State University. Series: Mathematical Modelling and Programming. 2014. Vol. 7, No. 4. P. 90–101. DOI: 10.14529/mmp140407

Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAM of Technological Support of Cutting Processes // ScienceDirect IFAC-PapersOnLine 49–12. 2016. P. 821–826. DOI: 10.1016/j.ifacol.2016.07.876

Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts // IFAC-PapersOnLine. 2017. Vol. 50, Issue 1. P. 2284-2289.

Szeider S. Finding Paths in Graphs Avoiding Forbidden Transitions // Discrete Applied Mathematics. 2003. No. 126. P. 261–273. DOI: 10.1016/S0166-218X(02)00251-2