mathematics

MDPI

Article

Special Type Routing Problems in Plane Graphs

Tatiana Makarovskikh *,† **and Anatoly Panyukov** †

Department of System Programming, South Ural State University, 454080 Chelyabinsk, Russia; paniukovav@susu.ru
* Correspondence: makarovskikh.t.a@susu.ru
† These authors contributed equally to this work.

Abstract: We considered routing problems for plane graphs to solve control problems of cutting machines in the industry. According to the cutting plan, we form its homeomorphic image in the form of a plane graph G. We determine the appropriate type of route for the given graph: *OE*-route represents an ordered sequence of chains satisfying the requirement that the part of the route that is not passed does not intersect the interior of its passed part, *AOE*-chain represents *OE*-chain consecutive edges which are incident to vertex v and they are neighbours in the cyclic order $O^{\pm}(v)$, *NOE*-route represents the non-intersecting *OE*-route, *PPOE*-route represents the Pierce Point *NOE*-route with allowable pierce points that are start points of *OE*-chains forming this route. We analyse the solvability of the listed routing problems in graph G. We developed the polynomial algorithms for obtaining listed routes with the minimum number of covering paths and the minimum length of transitions between the ending of the current path and the beginning of the next path. The solutions proposed in the article can improve the quality of technological preparation of cutting processes in CAD/CAM systems.

Keywords: routing; plane graph; polynomial algorithm

Citation: Makarovskikh, T.; Panyukov, A. Special Type Routing Problems in Plane Graphs. *Mathematics* **2022**, *10*, 795. https://doi.org/10.3390/math10050795

Academic Editor: Alexander A Lazarev

Received: 10 February 2022
Accepted: 26 February 2022
Published: 2 March 2022

1. Introduction

Simulation of some control and automation design problems [1,2] explains interest in routing problems for CAD/CAM systems. Lots of them devoted to finding the routes satisfying certain constraints have arisen from specific practical situations. All kinds of trajectory problems are universal mathematical models of optimization and control tasks. The examples of them are the following: (1) heuristic algorithms for constructing routes (N.A. Eapen and R.B. Heckendorn [3], S.Q. Xie [4], Y. Jing and C. Zhige [5], M.K. Lee and K.B. Kwon [6], J. Hoeft and U.S. Palekar [7]); (2) trajectory stabilization of mobile robots (V.A. Utkin [8]); (3) management of routing process and optimization (A.A. Lazarev [9]); (4) problems of obtaining the routes in graphs (H. Fleischner [10]); (5) the routing problem for cutting blanks from sheet material (V.M. Kartak [11], A.A. Petunin [12,13], A.G. Chentsov [14,15], I. Landovskaya [16]).

The capabilities of modern equipment for cutting sheet material allow using the cutting plans with the combining of contours for cut-out of separate parts. This combining of cuts allows reducing the material loss, the cutting length and the number of idle passes.

Algorithms for obtaining the cutting plans for tasks with combined cuts do not fundamentally differ from algorithms that do not allow any combining. However, the algorithms for finding the routes of the cutter moving are fundamentally different. Therefore, the development of algorithms for finding the route of the cutter for plans allowing the combining of the cut parts contours is still an open task.

In this paper, we consider the routing problems in plane graphs. These graphs are homeomorphic images of cutting plans. The cutter path is defined as a path covering all the boundaries of the cut parts. The main constraint on this path is that the faces of the route's initial part do not intersect with the edges of the remaining part. For flame cutting,

we need the following additional constraints on a path: (1) absence of self-intersections of the cutting path (NOE-condition), and (2) allowance to start cutting from allowable pierce vertices (PPOE-condition).

In practice, the most common approach does not involve a combination of the contours of the cut parts. This method is material and energy-consuming [13,17,18]. In one of such papers [3], the shapes are considered to be polygons. There are two different ways to cut each polygon: (1) entirely (complete cutting approach) or (2) partially (partial cutting approach) before cutting the next one. The authors of [3] proposed the approximation algorithm that uses such concepts as matching, spanning tree, and triangulation (MASTRI). This algorithm runs a time not greater than $O(n \log n)$, where n is the total number of all the polygons vertices. The cutting path computed by algorithm [3] is guaranteed to be within a factor of 3/2 of the optimum distance of the cutter. Hence, MASTRI algorithm can be used for computing cutting paths in industries like sheet metal cutting. It should be noted that the possibility of using concepts of matching, spanning tree, and triangulation was noted in our article [19].

The first attempts of constructing the routes in which the passed part of the route does not cover the edges of the remaining part were made in the work of U. Manber and S. Israni [20] where the image of the cutting plan is represented as an equivalent graph. The objective of this research is to cover this graph with a minimum number of chains starting in the pierce points or breakthroughs. Since the graph has $2k$ vertices of odd degree, then k pierce points are necessary and sufficient to traverse the graph. The cutter path problem formulation includes such parameters as manufacturing cost, efficiency, and distortion considerations. Some algorithms solving this task are considered in [20]. However, these algorithms do not have sufficient formalization, and the formulation of the problem does not take into account some technological constraints for flame cutting. Later, U. Manber and S.W. Bent [21] noted the need to construct a self-intersecting route and provided proof that this task belongs to the $\mathcal{NP}$ class. This proof is a compilation H. Fleischner's results [22] introducing the concept of an A-chain. This chain has the allowed transitions between edges, that are specified in a cyclic order at each vertex of the graph. H. Fleischner also proved that the task of constructing the A-chain is $\mathcal{NP}$-hard in general, but there are some special cases for which this task is solvable in polynomial time. One of such cases is a 4-regular graph. U. Manber and S.W. Bent in fact use A-chain instead of a self-intersecting chain. If we consider the partial case when the cutting plan is a plane Euler graph, then it is known that its dual face graph is bichromatic. For this case, S.B. Bely [23] proved the existence of an Euler cycle homeomorphic to a plane Jordan curve without self-intersections. However, it is unclear how to use this possibility for CAD/CAM systems for technological preparation of cutting processes.

The listed above problems have been solved by the authors. A rigorous formalization of these problems in terms of OE-chains is given in our paper [24]; however, the OE-chain allows the possibility of self-intersection of the trajectory. Representation of a plane Euler graph in the form of a self-intersecting Jordan curve has been announced at conferences [25,26]. We proved the necessary and sufficient conditions of $PPOE$-routes existence and built the polynomial algorithm $PPOE$-routing constructing such routes for any plane graph. The correctness of this algorithm has been announced at the conference [27]. The purpose of this article is to present the results obtained using a single terminology.

2. Methods

2.1. Abstracting the Cutting Plan to a Plane Graph

The information on the part shape is not used when we determine the sequence of cutting the fragments of the cutting. Hence, all the curves without self-intersections and contacts on the plane, representing the shape of the parts, are interpreted as the edges of the graph. All the points of intersection and contact are represented as the vertices of the graph. So, it is necessary to introduce additional functions on the set of vertices, faces and edges of the resulting graph to analyse the implementation of the given technological constraints.

We consider a plane S as a model of cutting plan, then a plane graph $G(V, E)$ with outer face $f_0 \subset S$ be the cutting plan model. The set of edges $E(G) \subset S$ of this graph is the Jordan curves with pairwise disjoint interiors and homeomorphic to open segments. Hence, the set of vertices $V(G) \subset S$ is the set of bounding points of these segments. For any part of the graph $J \subseteq G$, we denote the set-theoretic union of its interior faces (the union of all connected components of $S \setminus J$ that do not contain an outer face) by $\mathrm{Int}(J)$. Then, $\mathrm{Int}(J)$ can be interpreted as a part cut off a sheet. The sets of vertices, edges and faces of the graph G we denote as $V(G)$, $E(G)$ and $F(G)$, respectively. Since we consider graph G as a model of a cutting plan, there is no case when G is non-planar.

Theorem 1. *The topological representation of plane graph $G = (V, E)$ on plane S up to homeomorphism is defined by the following functions for each edge $e \in E$, $k = 1, 2$:*

- *$v_k(e)$ is the pair of vertices incident to e,*
- *$l_k(e)$ is the edges obtained by rotating edge e counter-clockwise around a vertex v_k,*
- *$r_k(e)$ is the edges obtained by rotating edge e clockwise around a vertex v_k,*
- *$f_k(e)$ is the face placed on the left when moving along the edge e from the vertex $v_k(e)$ to the vertex $v_{3-k}(e)$.*

Proof. An illustration of the functions from the Theorem 1 is given in Figure 1.

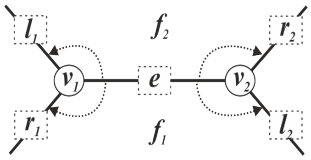

Figure 1. Functions representing graph edges.

Since functions $v_k(e)$, $f_k(e)$, $l_k(e)$, $k = 1, 2$ for graph G edges define incident vertices, faces, and adjacent edges for each $e \in E(G)$ this statement is obvious. □

Figure 2 illustrates an example of a cutting plan. Its homeomorphic image is given in Figure 3 and the named functions for its computer representation are given in Table 1. We can interpret any path obtained in graph G as a trajectory of the cutter since we know the inverse images of all the vertices.

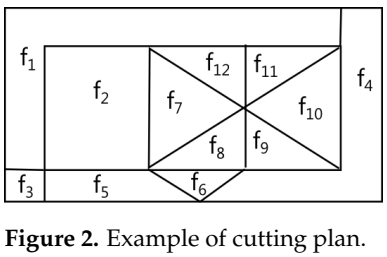

Figure 2. Example of cutting plan.

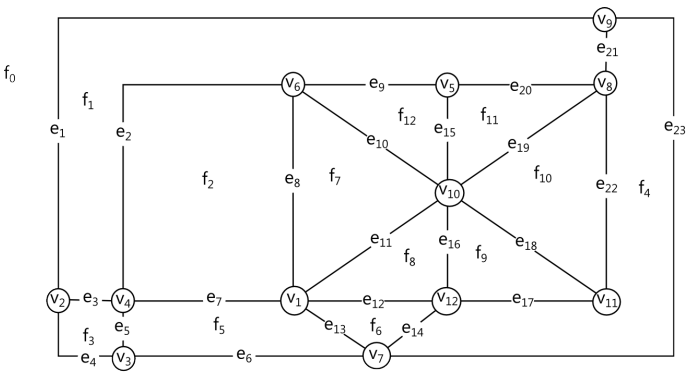

Figure 3. Geomorphic image of cutting plan in Figure 2.

Table 1. Encoding of the plane graph in Figure 3.

e	$v_1(e)$	$v_2(e)$	$l_1(e)$	$l_2(e)$	$r_1(e)$	$r_2(e)$	$f_1(e)$	$f_2(e)$
e_1	v_2	v_9	e_4	e_{21}	e_3	e_{23}	f_0	f_1
e_2	v_4	v_6	e_3	e_8	e_7	e_9	f_1	f_2
e_3	v_2	v_4	e_1	e_5	e_4	e_2	f_1	f_3
e_4	v_2	v_3	e_3	e_6	e_1	e_5	f_3	f_0
e_5	v_3	v_4	e_4	e_7	e_6	e_3	f_3	f_5
e_6	v_3	v_7	e_5	e_{23}	e_4	e_{13}	f_5	f_0
e_7	v_4	v_1	e_2	e_{13}	e_5	e_8	f_2	f_5
e_8	v_6	v_1	e_{10}	e_7	e_2	e_{11}	f_7	f_2
e_9	v_6	v_5	e_2	e_{15}	e_{10}	e_{20}	f_1	f_{12}
e_{10}	v_6	v_{10}	e_9	e_{11}	e_8	e_{15}	f_{12}	f_7
e_{11}	v_1	v_{10}	e_8	e_{16}	e_{12}	e_{10}	f_7	f_8
e_{12}	v_1	v_{12}	e_{11}	e_{14}	e_{13}	e_{16}	f_8	f_6
e_{13}	v_1	v_7	e_{12}	e_6	e_7	e_{14}	f_6	f_5
e_{14}	v_7	v_{12}	e_{13}	e_{17}	e_{23}	e_{12}	f_6	f_4
e_{15}	v_5	v_{10}	e_{20}	e_{10}	e_9	e_{19}	f_{11}	f_{12}
e_{16}	v_{10}	v_{12}	e_{18}	e_{12}	e_{11}	e_{17}	f_9	f_8
e_{17}	v_{12}	v_{11}	e_{16}	e_{22}	e_{14}	e_{18}	f_9	f_4
e_{18}	v_{10}	v_{11}	e_{19}	e_{17}	e_{16}	e_{22}	f_{10}	f_9
e_{19}	v_{10}	v_8	e_{15}	e_{22}	e_{18}	e_{20}	f_{11}	f_{10}
e_{20}	v_5	v_8	e_9	e_{19}	e_{15}	e_{21}	f_1	f_{11}
e_{21}	v_9	v_8	e_{23}	e_{20}	e_1	e_{22}	f_4	f_1
e_{22}	v_8	v_{11}	e_{21}	e_{18}	e_{19}	e_{17}	f_4	f_{10}
e_{23}	v_9	v_7	e_1	e_{14}	e_{21}	e_6	f_0	f_4

2.2. OE-Routing

2.2.1. Basic Definitions

Let us consider the formulation and solution of the problem for constructing the routes in a plane graph that satisfy the condition that the interior faces of any their initial parts do not intersect with the edges of the remaining part. Formally, such routes are defined as an ordered sequence of *OE*-chains (ordered enclosing chains) of the graph $G = (V, E)$ and form a class of *OE*-paths. The definitions, proofs and notations of the theory of routes with ordered enclosing (*OE*-routes) are introduced in [19]. Let us give these definitions to avoid the loss of generality.

Definition 1. *Chain $C = v_1e_1v_2e_2 \dots v_k$ in plane graph G has ordered enclosing (is an OE-chain), if for any its initial part $C_l = v_1e_1v_2e_2 \dots e_l$, $l \leq (|E|)$ the condition* $\mathrm{Int}(C_l) \cap E = \emptyset$ *holds.*

Theorem 2 ([19])**.** *Let $G = (V, E)$ be a plane Euler graph. For any vertex $v \in V(G)$ incident to outer (infinite) face of graph G there exists Euler OE-cycle $C = ve_1v_1e_2v_2 \dots v_{|E|-1}e_{|E|}v$.*

The proof of this theorem gives the recursive algorithm of *OE*-cycle constructing. This algorithm has computing complexity $O(|E|^2)$. However, there exists non-recursive approach with computing complexity $O(|V| \cdot \log|E|)$ [28].

Let's generalize Definition 1 up to the notion of *OE*-route plane graphs (it is possible non-Eulerian and disconnected).

Definition 2. *The ordered sequence of edge-disjoint OE-chains*

$$\begin{aligned} C^0 &= v^0e_1^0v_1^0e_2^0 \dots e_{k_0}^0v_{k_0}^0, \quad C^1 = v^1e_1^1v_1^1e_2^1 \dots e_{k_1}^1v_{k_1}^1, \dots, \\ C^{n-1} &= v^{n-1}e_1^{n-1}v_1^{n-1}e_2^{n-1} \dots e_{k_{n-1}}^{n-1}v_{k_{n-1}}^{n-1}, \end{aligned}$$

covering graph G and such that

$$(\forall m : m < n), \quad \left(\bigcup\nolimits_{l=0}^{m-1} \mathrm{Int}(C^l)\right) \cap \left(\bigcup\nolimits_{l=m}^{n-1} C^l\right) = \emptyset$$

is a route with ordered enclosing (OE-route).

Definition 3. *Let a route consisting of a minimal (in cardinality) ordered sequence of edge-disjoint OE-chains in a plane graph G be called an Euler route with ordered enclosing (Euler OE-route), and OE-chains forming it be the Euler OE-cover.*

Theorem 3 ([27])**.** *Let G connected plane graph, $V_{odd}(G)$ be the set of its odd vertices, then the cardinality N of Euler OE-cover of G satisfies the inequality*

$$k = \frac{|V_{odd}(G)|}{2} \leq N \leq |V_{odd}(G)| = 2k$$

holds. The upper and lower bounds are reachable.

The cover capacity is significantly influenced by the presence of bridges in the graph. In their absence, the lower bound is reached, in the case of the existence of vertices of odd degree incident to the outer face; or, if there are no such vertices, the cardinality of the cover is one higher than the lower bound.

The construction of the *OE*-route of the graph *G* solves the considered cutting problem in the absence of restrictions on self-intersections and the placement of starting (i.e., pierce) points for all chains.

2.2.2. Algorithms Constructing *OE*-Chains for Connected Graph *G*

Algorithms for constructing *OE*-routes in plane Eulerian graphs are known [28]. The possibility of constructing an *OE* route in an arbitrary plane graph demonstrates Theorem 4 [24].

Theorem 4. *Let $G = (V, E)$ be plane connected graph without bridges on S. There exists the set of edges $H : (H \cap S) \backslash V = \varnothing$ so that graph $\hat{G} = (V, E \cup H)$ be Euler, and there exists Euler cycle in graph $\hat{G}$, such that $C = v_1e_1v_2e_2...e_nv_1$, $n = |E| + |H|$, for any its initial part $C_l = v_1e_1v_2e_2...v_l$, $l \leq |E| + |H|$ the condition $\mathsf{Int}(C_l) \cap G = \varnothing$ holds.*

We use the concept of the edge *e* rank while considering the algorithms of *OE*-routes constructing.

Definition 4. *The rank of edge $e \in E(G)$ be the value of function $\mathrm{rank}(e) : E(G) \to \mathbb{N}$ recursively defined as following:*

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

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

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

Definition 5 ([28])**.** *Let rank of face $f \in F(G)$ be a value of function $\mathrm{rank} : F(G) \to \mathrm{N}_0$:*

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

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

Definition 6 ([28])**.** *Let rank of vertex $v \in V(G)$ be a value of function $\mathrm{rank} : V(G) \to \mathrm{N}$: $\mathrm{rank}(v) = \min_{e \in E(v)} \mathrm{rank}(e)$ where $E(v)$ is a set of edges incident to vertex $v \in V$.*

We developed the following polynomial time algorithms for constructing OE-routes for plane graphs:

- Euler OE-cycle in plane Euler graph (algorithm `OE-CYCLE`, computing complexity $O(|V|^2)$ [29]);
- connected OE-route of Chinese postman for any plane connected graph; removing re-traversed edges will result in a OE-route; this route is not optimal either in terms of the number of covering chains or the length of idle passes (algorithm `CPP_OE`, computing complexity $O(|E(G)| \cdot |V(G)|)$ [30]);
- a route in plane connected graph without bridges being the OE-cover optimal by the number of chains, the length of idle passes may not be optimal (algorithm `OECover`, computing complexity $O(|E| \cdot \log |V|)$ [19]);
- OE-route in plane connected graph without bridges with additional edges, connecting the odd vertices (algorithm `M-COVER` with computing complexity $O(|E| \cdot \log |V|)$); this algorithm appended by algorithm of finding the shortest matching between odd vertices allows to obtain OE-cover with minimal summary length of additional edges (computing complexity $O(|E| \cdot \sqrt{|V|})$) [19].

In this paper, we describe in details those of them that are not published in open access for the convenience of the reader.

Algorithm 1 `OECover` covers the plane graph G by an ordered sequence of OE-chains. The graph G is encoded by the list of edges, and for each edge e the functions considered in Theorem 1 are defined.

Algorithm 1 Algorithm OECover

Require: $G = (V, E)$ be a plane graph; $V_{odd} \subseteq V$ be the set of odd vertices;
Ensure: $first \in E$, $last \in E$, $\mathrm{mark}_1 : E \to E$;

```
Initiate();                                  ▷ Assign the initial values of all used variables
Order();                                     ▷ Define the ranks of edges, and form the ordered lists for vertices
SortOdd();                                   ▷ Sorting of odd vertices by decreasing of their rank
if {∃v ∈ V_odd | v ∈ f_0} then               ▷ Define the starting value of a chain
    v^0 ← arg max_{v∈V_odd} rank(v);   V_odd ← V_odd\{v^0};
else   v^0 ← v | v ∈ f_0;
end if
while (true) do
        v ← FormChain(v^0);                  ▷ Form a chain from the defined vertex
        V_odd ← V_odd\{v};                   ▷ Exclude the starting vertex of current chain from the list
    if ( then V_odd = ∅)                     ▷ Check the possibility to construct one more chain
        break;
    end if
        v^0 ← arg max_{v∈V_odd} rank(v);
end while
```

In the body of the procedure `Initiate`, the initial values of all used variables are assigned, and the first edge $e_0 \in E$ belonging to the boundary of the outer face f_0 is defined.

Procedure `Order` Algorithm 2 functional purpose of the `Order` procedure is in:

(1) defining the value $\mathrm{rank}(e)$ for each edge $e \in E$ (note that the rank of any edge of a plane graph can be determined in time $O(|E|)$ using this procedure);

(2) forming the list $Q(v)$ of incident edges for each vertex (the edges are ordered in descending order of the $\mathrm{rank}()$ value).

Algorithm 2 Procedure Order

```
1: procedure ORDER
2:     while first ≠ ∞ do
3:         while (mark(ne) = ∞) and (last ≠ ne) do
4:             M1:                                  ▷ Forming the queue of M1-marked edges
5:             rank(ne) ← k;                        ▷ Define the rank of an edge
6:             mark_1(last) ← ne;
7:             if v_2(ne) ≠ v then
8:                 REPLACE(ne);
9:             end if
10:            v ← v_1(ne); last ← ne; ne ← l_1(ne);
11:        end while
12:        e ← first; first ← mark_1(first); v ← v_2(e); ne ← l_2(e);
13:        M2:          ▷ Placing the M1-marked edges to the lists of the corresponding vertices
14:        k ← rank(e) + 1; mark_1(e) ← Stack(v_1(e)); mark_2(e) ← Stack(v);
15:        if mark_1(e) ≠ 0 then
16:                        ▷ Form queue of M1-marked edges of all unmarked edges bounding f_1(e)
17:            if v_1(e) = v_1(mark_1(e)) then
18:                prev_1(mark_1(e)) ← e;
19:            else
20:                prev_2(mark_1(e)) ← e;
21:            end if
22:        end if
23:        if mark_2(e) ≠ 0 then          ▷ Pushing of edge to stacks of vertices v_1(e) and v_2(e)
24:            if v = v_1(mark_2(e)) then
25:                prev_1(mark_2(e)) ← e;
26:            else
27:                prev_2(mark_2(e)) ← e;
28:            end if
29:            Stack(v) ← e; Stack(v_1(e)) ← e;
30:        end if
31:    end while
32: end procedure
```

After executing the `Initiate` and `Order` procedures, the odd vertices $v \in V_{odd}$ are ordered in ascending order of their rank using the `SortOdd` procedure. The rank of the vertex v is the value of the function $\mathrm{rank}(Stack(v))$. Then, the loop `do...while` is executed using the `FormChain` procedure (see Algorithm 3). This cycle constructs a sequence of $|V_{odd}|/2$ simple paths between pairs of odd vertices. If none of the odd vertices is adjacent to the outer face, then it is necessary to construct a $|V_{odd}|/2+1$ chain, where the first of the constructed paths C^0 starts at vertex of even degree $v^0 \in f_0$, adjacent outer face, and ends at an odd vertex. All the chains of the cover $C^1, \dots C^{n-1}$ are connecting the odd vertices, and the last one C^n starts at odd vertex, and ends at vertex $v^0 \in f_0$.

The aim of `FormChain` procedure is to obtain the OE-chain starting in a given vertex w and ending in some odd vertex $v \in V_{odd}$, $v \neq w$. As a result of the procedure, a simple chain will be obtained $C^i = v_0^i e_1^i v_1^i e_2^i \dots e_k^i v_k^i$, for which $v_1^i, v_2^i, \dots v_{k-1} \notin V_{odd}^i$, and for $i \neq 0$ and $i \neq n$ vertices $v_0^i, v_k^i \in V_{odd}$, if $i = 0$ vertex $v_k^i \in V_{odd}$, and if $i = n$ vertex $v_0^i \in V_{odd}$,

$$e_i = \arg \max_{e \in E(v_i) \setminus \{e_l | l < i\}} \mathrm{rank}(e),\ v_{i+1} = \overline{v_1}(e_i),\ i = 1, 2, \dots, k,$$

moreover, for any initial part $C_l = v^0 e_1 v_1 e_2 v_2 \dots e_l$, $l \leq k$ and for any vertex $v \in V$ the inequality

$$\min_{e \in E(v) \bigcap E(C_l)} \mathrm{rank}(e) > \max_{e \in E(v) \setminus E(C_l)} \mathrm{rank}(e)$$

holds.

Algorithm 3 Procedure FormChain

procedure FORMCHAIN(In: w starting vertex of a chain; Out: v ending vertex of a chain)
  $v \leftarrow w$; $e \leftarrow Q(v)$;
  do
  $e_1 = \arg\max_{e \in Q(v)} \text{rank}(e)$;
  $e_2 = \arg\max_{e \in Q(v): f_1(e) = f_2(e)} \text{rank}(e)$;
  if $\text{rank}(e_1) = \text{rank}(e_2)$ **then** ▷ Find the edge of maximal rank, a bridge if possible
    $e = e_2$;
  else
    $e = e_1$;
  end if
  if $v = v_1(e)$ **then**
    REPLACE(e); ▷ Change the indexes of functions for edge e from k to $3 - k$, $k = 1, 2$
  end if
  $E(G) \leftarrow E(G) \setminus \{e\}$; ▷ Delete edge e and delete faces divided by edge e
  $Trail \leftarrow Trail \cup \{e\}$;
  $v \leftarrow v_1(e)$;
  while $(v \notin V_{odd} \& Q(v) \neq \emptyset)$;
  return v;
end procedure

Theorem 5. *Let $G = (V, E)$ be a plane connected bridgeless graph on S, and $V_{odd} \subset V$ be the set of odd vertices. For any matching M on set V_{Odd} in graph $\hat{G} = (V, E \cup M)$, there exists Euler cycle $C = v_1 e_1 v_2 e_2 ... e_n v_1$, $n = |E| + |M|$, for any initial part $C_l = v_1 e_1 v_2 e_2 ... v_l$, $l \leq |E| + |M|$ of which, the condition $\text{Int}(C_l) \cap G = \emptyset$ holds.*

The proof of Theorem 5 is constructive and consists in proving the efficiency of the algorithm M-`Cover` (see Algorithm 4) for constructing a cover for any matching on the set of odd vertices [19].

Algorithm 4 Algorithm M-`Cover`

Require: plane connected graph G, functions $v_k(e)$, $l_k(e)$, $e \in E(G)$, $k = 1, 2$; vertex $v_0 \in V(G)$ incident to outer face; matching M on set of odd vertices V_{Odd}; boolean function $\texttt{Idle}_M : V_{Odd} \to \{\textbf{false}, \textbf{true}\}$ on set of odd vertices V_{Odd};
Ensure: almost ordered set C of OE-chains of graph G, being the OE-cover of graph G;
`Order` (G); ▷ Define rank() for all $e \in E(G)$, $v \in V(G)$
$v := v_0$; ▷ Constructing
while $Q(v) \neq \emptyset$ **do**
  `FormChain`(v, v);
  if $\texttt{Idle}_M(v) \vee (Q(v) = \emptyset)$ **then**
    $u \leftarrow M(v)$; ▷ Vertex u is a pair for vertex v in matching M
    $V_{Odd} \leftarrow V_{Odd} \setminus \{u, v\}$ ▷ Delete vertices u, v from V_{Odd}
    $v \leftarrow u$; ▷ Finish constructing the current chain
  end if
end while
End of algorithm

The main difference of this algorithm from OE-Cover is that for each vertex $v \in V_{Odd}$ the next one $u = M(v) \in V_{Odd}$ is fixed. It is the vertex to which the transition is made. Algorithm `M-Cover` can finish constructing the current chain both at the first visit to the vertex $v \in V_{Odd}$, and at the moment when the vertex becomes dead-end (i.e., $Q(v) = \emptyset$). To determine at what moment to finish the constructing of the chain, the values of

$$\texttt{Idle}_M(v) = (\text{rank}(v) \leq \text{rank}(M(v))) \wedge \left(f_{M(v)} \succeq f_v\right),\ v \in V_{Odd},$$

are used, where $f_w = \arg\min_{f: v \in f \subset F(G)} \text{rank}(f), w \in V_{Odd}$. Here $\succeq$ is partial ordering on $F(G)$ induced by tree $T^{G'}_{f0}$ of shortest paths to vertex $f_0 \in F$:

$$(f_i \succeq f_j) \leftrightarrow \left(f_j \text{ belongs to chain } T^{G'}_{f_0} \text{ between } f_i and f_0\right).$$

To construct the optimal cover (i.e., cover with a minimal length of additional edges) it is enough to take the shortest matching on set of odd vertices V_{odd} as M. This task is realized by the following Algorithm 5.

Algorithm 5 `OptimalCover`

Require: plane graph G represented by the list of edges with defined functions $v_k(e)$, $l_k(e)$, $f_k(e)$, $k = 1, 2$
Ensure: cover of graph G by OE-chains C_j, $j = 1, ..., |V_{odd}|/2$
1: Define the shortest matching M on set V_{odd}
2: Run algorithm `M-Cover` for graph G and matching M
3: Stop

Obviously, Algorithm 5 allows us to construct the optimal OE-cover, and its computing complexity is not greater than $O(|V|^3)$ (but by using special data structures and algorithms, it is possible to run this algorithm by the time not exceeding $O(|E(G)| \cdot \sqrt{|V(G)|})$). This estimation is defined by the computing complexity of Step 1.

2.3. Constructing of Routes Satisfying the Combination of Constraints

During the technological preparation of the cutting process, various constraints on the trajectory of the cutting tool may appear. One of them is the task considered above, where the cut off part of the sheet of the obtained route does not require additional cuts. However, in practice, it is required to fulfil additional constraints on the absence of intersection of cuts and on allowable pierce points that are start points of OE-chains forming this route.

To solve a problem of the cuts intersection absence at each vertex of the graph, a cyclic order of traversing the edges is specified, and the continuation of the traversal along the chain is carried out only by this cyclic order. In the general case, the problem of finding such a chain in a graph belongs to the class of $\mathcal{NP}$-complete problems, but there are effective algorithms of its solution for some special cases.

2.3.1. *AOE*-Routs

Let us consider Euler chain

$$T = v_0, k_1, v_1, \ldots, k_n, v_n,\ v_n = v_0$$

in graph $G = (V, E)$. Let we know the cyclic order $O^{\pm}(v)$ defining the transitions system $A_G(v) \subseteq O^{\pm}(v)$ for each vertex $v \in V$. In the case when $\forall v \in V(G)\ A_G(v) = O^{\pm}(v)$ the transitions system $A_G(v)$ is called the full transitions system, and chain satisfying this system is A_G-compatible.

Definition 7. *A_G-compatible chain T is called **A-chain**. Thus, consecutive edges in the chain T incident to vertex v are the neighbours in the cyclic order $O^{\pm}(v)$ [31].*

Definition 8. *The chain is called AOE-chain if it is OE-chain and A-chain simultaneously [32].*

Theorem 6. *If there is A-chain in a plane graph G then there is also AOE-chain in this graph [32].*

Theorem 7. *Plane connected 4-regular graph G has AOE-chain.*

To prove this theorem, we need to introduce some definitions and prove some propositions.

Definition 9. *The partial graph G_k of graph G for which $E(G_k) = \{e \in E(G) : \mathrm{rank(e)} \geq \mathrm{k}\}$ is called partial graph of rank k.*

Preliminarily, the "correct" splitting of all cut-vertices of partial graphs G_k is performed, so that as a result of the splitting, we get a graph for which any partial graph G_k has no cut-vertices. The vertices splitting is a local operation, hence the sequence of splitting does not affect the total result. The "correct" transition is one between arcs corresponding of a cyclic order and incident to the different pairs of faces (see Figure 4b)). The splitting result, in this case, is shown in Figure 4b).

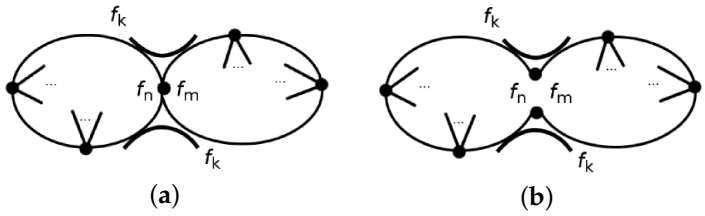

Figure 4. Splitting of cut-vertices of rank *k*. (**a**) The correct transitions system for splitting the cut-vertex of partial graph G_k. (**b**) Splitting according to the transitions system in cut-vertex for partial graph G_k.

These propositions imply the effectiveness of the `CUT-POINT-SPLITTING` Algorithm 6 running in time not greater than $O(|E(G)| \log |V(G)|)$.

Theorem 8. *Algorithm AOE-`CHAIN` constructs AOE-chain for plane connected 4-regular graph G any partial graph G_k, $k = 1, 2, \ldots$ of which has no cut-vertices. Algorithm solves the problem by the time $O(|E(G)| \cdot \log |V(G)|)$.*

The proof of this algorithm's effectiveness [32] finishes the proof of Theorem 7.

Proposition 1. *Vertex incident to four edges bounding outer face is cut-vertex.*

Proposition 2. *The outer face of a partial graph G_k is the union of all faces of rank k in graph G.*

Let us consider the Algorithm 7 for constructing the *AOE*-chain for plane connected 4-regular graph [32,33].

Algorithm 6 CUT-POINT-SPLITTING

Require: plane connected 4-regular graph $G = (V, E)$ represented for all $e \in E(G)$ by functions v_s, l_s, r_s , $s = 1, 2$.
Ensure: homeomorphic image of graph $G = (V, E)$ for which any partial graph G_k has no cut-vertices.
Supplementary data $\forall v \in V(G)$:
point(v) is the array of pointers to one of edges incident to vertex v;
rank(v) is the array of vertices ranks;
count(v) is the counter of incident edges of one rank for each vertex;
Supplementary data $\forall f \in F(G)$: array rank(f).

Initiate():
for all $v \in V(G)$ **do** ▷ Zero the counter of edges of the same rank incident to a vertex
    point$(v) := 0$; count$(v) := 0$
end for
Ranking(G) ▷ Determining the rank of all vertices, edges and faces of a graph
Finding(): ▷ Defining the cut-vertices
for all $e \in E(G)$ **do**
    point$(v_1(e))$: point$(v_2(e)) := e = e$
end for
for all $v \in V(G)$ **do** ▷ To look through all the vertices

    $e :=$ point(v); $k :=$ rank(v) ▷ Save the rank value k of the incident edge e
    if $v = v_1(e)$ **then** ▷ Define the direction of edge e
        $s := 1$
    else
        $s := 2$
    end if
    $e := l_s(e)$ ▷ Counting the number of rank k edges incident to vertex v
    for $i = 1$ **up to** 4 **do**
        if rank$(e) = k$ **then**
            count$(v) :=$ count$(v) + 1$
            if $i < 4$ **then**
                $e := l_s(e)$
            end if
        end if
    end for
    if count$(v) = 4$ **then** ▷ Split the vertex if it is cut-vertex
        if $(f_s(e) = f_s(l_s(e))$ **and** $f_{3-s}(e) = f_{3-s}(l_s(e)))$ **or**
            or $(f_s(e) = f_{3-s}(l_s(e))$ **and** $f_{3-s}(e) = f_s(l_s(e)))$ **then**
            $e^* := l_s(e)$, $l_s(e) := r_s(e)$, $r_s(r_s(e)) := e$,
            $r_s(e^*) := l_s(e^*)$, $l_s(l_s(e^*)) := e^*$
        else
            $e^* := r_s(e)$, $r_s(e) := l_s(e)$, $l_s(l_s(e) := e$,
            $l_s(e^*) := r_s(e^*)$, $r_s(r_s(e^*)) := e^*$
        end if
    end if
end for

2.3.2. *NOE*-Routes

Algorithm 6 *AOE*-CHAIN is used for running the algorithm NOE-CHAIN (see Algorithm 8) to obtain the non-intersecting *OE*-chain for plane connected graph [26].

Definition 10 ([26]). *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.*

Algorithm 7 Algorithm *AOE*-CHAIN

Require: plane connected 4-regular graph $G = (V, E)$ defined by functions v_k, l_k, r_k, $k = 1, 2$ (see Theorem 1); starting vertex $v \in V(f_0)$.
Ensure: *AChain* – output stream containing *AOE*-chain obtained by the algorithm.

Initiate(G, v_0);
Ranking(G);
CUT_POINT_SPLITTING (G); ▷ Deleting of cut-vertices in partial graphs of each rank
▷ Constructing
$e = \arg\max_{e \in E(v)} \text{rank}(e)$ ▷ Choose the edge of maximal rank incident to vertex v
repeat
  if $v \neq v_1(e)$ **then**
    REPLACE(e)
  end if ▷ If necessary, adjust the numbering of functions for the edge e
  AChain ←**Print**(v, e) ▷ Add edge e to the resulting sequence *AChain*
  mark(e) := **false**; counter:=counter+1; $v := v_2(e)$ ▷ Mark the current edge as passed
  if (rank($r_2(e)$) $\geq$ rank($l_2(e)$)) **then** ▷ Choose the next edge of maximal possible rank
    if mark($r_2(e)$) **then** ▷ Check if the chosen edge is already passed
      $e := r_2(e)$ ▷ The passed edges have False value in the arra *mark*
    else
      $e := l_2(e)$
    end if
  else
    if (mark($l_2(e)$) **then** ▷ Choose the not passed edge
      $e := l_2(e)$
    else
      $e := r_2(e)$
    end if
  end if
until (counter $> |E(G)|$) ▷ Finish the cycle when all the edges are scanned
End of Algorithm

Algorithm 8 NOE-CHAIN (G)

Require: plane Euler graph G defined by functions $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$ (see Theorem 1) and rank(e);
Ensure: C as *NOE*-chain in graph G;

$\widehat{G} = \text{NonIntersecting}(G)$; ▷ Split all vertices of degree higher than 4
C^*=AOE_CHAIN($\widehat{G}$); ▷ Obtain *AOE*-chain in graph $\tilde{G}$
C=Absorb(C^*); ▷ Absorb all split vertices and obtain the resulting *NOE*-chain

Its execution means transforming the initial graph to a plane connected 4-regular graph by splitting the vertices of degree greater than 4. To obtain the Euler *NOE*-cycle in a plane Euler graph without given transitions system, we can act as follows. Let us define boolean function

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

on the set of vertices $V(G)$. When performing initialization, declare all vertices not viewed. Function NonIntersecting (G) (Algorithm 9) splits all vertices $v \in V(G)$ of degree more than $2k - 1$ ($k \geq 3$) to k fictive vertices of degree 4 and introduces k fictive edges incident to the vertices obtained as a result of splitting and forming a cycle (see Figure 5).

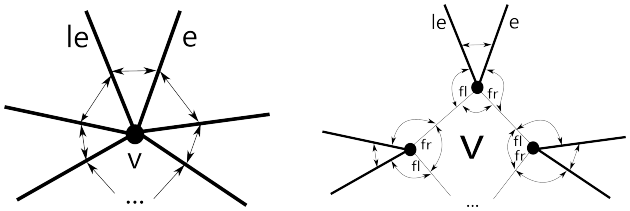

Figure 5. Splitting of vertex (the edges of graph G are bold lines, and the fictive ones are thick lines) and modification of the pointers according to the splitting processed.

Algorithm 9 Function `NonIntersecting (G)`

Require: plane Euler graph G defined by functions $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$ (see Theorem 1) and rank(e);
Ensure: plane connected 4-regular graph G^* defined as the same;
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** ▷ Searching of vertices of degree greater than 4 and their splitting
5: $k := 1$; ▷ Consider vertex with index 1, then vertex with index 2
6: **while** $(k \leq 2)$ **do**
7: **if** (! $Checked(v_k(e))$) **then** ▷ Process only a previously unprocessed vertex
8: **if** $(k = 2)$ **then** ▷ Improve the indexes
9: REPLACE(e); ▷ Process vertices $v_1(e)$
10: **end if**
11: Handle (e); ▷ Call the function to process vertex $v_1(e)$
12: $Checked(v_1(e)) :=$ **true**; ▷ Mark the vertex as considered
13: **end if**
14: $k := k + 1$;
15: **end while**
16: **end for**
End of function

In the body of function we use the procedure `Handle (`e`,` $v_k(e)$`,` k`)`, which processes each unconsidered graph vertex.

Procedure Algorithm 10 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. Here the handled vertex is split to $d/2$ fictive vertices, and d fictive edges incident to these vertices. There fictive edges form a cycle.

In lines 18–23, we not only change 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 [26].

Definition 11. *The rank of fictive edge (line 20) is equal to the rank of the initial graph face incident to the entered fictive edge.*

The introduced by `Handle` procedure $k/2$ fictive vertices and k fictive edges incident to these vertices are forming a cycle. As a result of processing all graph G vertices, we obtain the modified plane connected 4-regular graph G^*. Algorithm `AOE-CHAIN()` constructing AOE-chain T^* can be implemented to graph G^*. The considered procedure is realized in algorithm `NonIntersecting` (see Algorithm 9). If then in T^* all the fictive edges and the incident vertices obtained by splitting the vertex v are replaced by v, then we obtain the NOE-chain T in the original graph G.

Algorithm 10 Procedure Handle (e)

procedure HANDLE(e)
  $v := v_1(e)$; ▷ Splitting vertex
  $e_{first} := e$; ▷ Save the first considered edge
  $d := 0$; ▷ Initialization of a counter for vertex degree d
  $F := FaceNum() + 1$; ▷ Define the number of a new face
  repeat ▷ Pass 1: Defining the degree of v
    $le := l_1(e)$;
    if ($v_1(le) \neq v$) **then** REPLACE(le);
    end if ▷ Change the indexing of functions if necessary
    $e := le$; $d := d + 1$; ▷ Consider the edge when calculating the degree and move on to the next one
  until ($e = e_{first}$); ▷ Repeat until all edges incident v have been considered
  if ($d > 4$) **then** ▷ If the degree of current vertex is greater than 4
    $e := e_{first}$; ▷ Begin from the first considered edge
    $le := l_k(e)$; ▷ Define the number of its left neighbour
    $e_{next} := l_k(le)$; ▷ Save the edge for the next iteration
    $fl :=$ **new** EDGE; $fle := fl$; $e_{first} := e$; ▷ Introduce a fictive edge adjacent to le
    repeat ▷ Put the pointers for edges
      $e := e_{next}$; $le := l_k(e)$; $fr := fl$;
      $f_1(fl) := F$; $f_2(fl) := f_2(e)$; ▷ Define faces adjacent to a fictive edge
      rank(fl) := facerank($f_2(fl)$); ▷ Define "rank" of fictive edge
      ▷ Function facerank() defines the rank of a face according to the definition
      $fl :=$ **new** EDGE; $e_{next} := l_k(le)$;
    until ($l_k(le) = e_{first}$);
  end if
end procedure

Theorem 9. *Algorithm* `NOE-CHAIN` *solves the task of constructing the NOE-chain for plane Euler graph by the time* $O(|E(G)|^2)$ *[26].*

Note that this algorithm constructs a *NOE*-chain in a plane Euler graph. In the case of a plane non-Euler (generally disconnected) graph G, it is necessary to split all vertices of degree higher than 4 by the Algorithm 10. As a result, we get a graph with vertex degrees equal to 3 or 4. For this graph, we apply the algorithm for constructing an *AOE*-cover. In the chains of the resulting cover, remove all artificial edges and absorb all split vertices. As a result, we get *NOE*-cover.

2.3.3. *PPOE*-Routes

Let us consider a problem arising in the case of intrusion of constraints on the location of pierce points. Obviously, the number of pierce points is determined by the number of covering chains. According to Theorem 3, the number of pierce points is at least $|V_{odd}|/2$. This problem can be formalized as following.

- Let faces $F_{in}(G) \subset F(G)$ allow piercing.
- Let odd vertices $v^- \in V_{in}(G) \subset V(G)$ be incident to face $F_{in}(G)$.
- Let for odd vertices $v^+ \in V_{out} = V_{odd} \setminus V_{in}$ piercing is forbidden.

If the constructed route in the graph is an *OE* route and all the initial vertices of the covering chains belong to $V_{in}(G)$, then this route can be used as a basis for constructing a route for the cutter trajectory for laser cutting process. Let these routes be called *PPOE*-routes [27].

Definition 12. *Let chain $C = v_1e_1v_2e_2 \dots v_k$ be called PPOE-**chain**, if it is OE-chain and starts from vertex $v_1 \in V_{in}(G)$.*

Definition 13. *Let PPOE-**cover** of graph G be such an OE-cover of G, consisting of PPOE-chains.*

Definition 14. *An ordered sequence of edge-disjoint PPOE-chains in a plane graph G of the minimal cardinality is called an Euler PPOE-cover.*

The problem of determining the realizability of a cutting plan can be formulated as determining the existence of an Euler *PPOE*-cover for a plane graph that is a homeomorphic image of the corresponding cutting plan. Following the existing restrictions, we can formulate the following necessary condition for the existence of a *PPOE*-cover.

Theorem 10. *Plane connected graph $G(V, F, E)$ without bridges has PPOE-cover if and only if the cardinality of minimal $\{V_{in},\ V_{out}\}$-cut is at least $|V_{out}|$.*

Proof. The validity of the necessary condition is obvious, sufficiency follows from the effectiveness of Algorithm 11 solves the problem of constructing *PPOE*-cover for a plane graph $G(V, E)$ without bridges. □

To find minimal $\{V_{in},\ V_{out}\}$-cut let us construct a network

$$N(V \cup \{w\},\ A \cup (\{w\} \times V_{in}))$$

(i.e., directed graph with source w), in which

- a pair of arcs $(u, v), (v, u) \in A(N)$ of capacity 1 corresponds to edge $e = \{u, v\} \in E(G)$;
- vertices $v^+ \in V_{out}(N)$, i.e., points of possible end of chain, are the sinks of a unit power flow;
- vertices $v^- \in V_{in}(N)$, i.e., possible pierce points, may be source of the unit.

Cardinality of minimal $\{V_{in},\ V_{out}\}$-cut can be obtained as optimal value of problem

$$\sum_{(u,v)\in A(N)} x(u,v) \to \min, \tag{1}$$

$$\sum_{v:\,(u,v)\in A(N)} x(u,v) - \sum_{v:\,(v,u)\in A(N)} x(v,u) = 1, \qquad u \in V_{out}(N), \tag{2}$$

$$-\sum_{v:\,(u,v)\in A(N)} x(u,v) + \sum_{v:\,(v,u)\in A(N)} x(v,u) = -x(w,u), \qquad u \in V_{in}(N), \tag{3}$$

$$\sum_{v:\,(u,v)\in A(N)} x(u,v) - \sum_{v:\,(v,u)\in A(N)} x(v,u) = 0, \qquad u \in V\backslash(V_{out}(N) \cup V_{in}(N)), \tag{4}$$

$$\sum_{v\in V_{in}(N)} x(w,v) = |V_{out}(N)|, \tag{5}$$

$$0 \le x(u,v) \le 1, \quad (u,v) \in A(N), \tag{6}$$

$$0 \le x(w,u) \le 1, \quad u \in V_{in}(N), \tag{7}$$

where w is a common source with capacity $|V_{out}|$ adjacent to all $v \in V_{in}$ to network N.

Let $x: A \to \{0,1\}$ be optimal solution of problem (1)–(7). Let us construct a sequence of disjoint chains $C_1, C_2, \ldots C_{|V_{out}|}$, containing all the flow holders of x and only them. It is possible to «correctly» split each vertex $v \in V(G)$ to the dummy vertices with «correct» union of active arcs lists, while it is possible (i.e., taking into account the cyclic order on the set of arcs and their orientations). The examples of «correct» splitting and uniting are shown in Figure 6. The result of this step is a sequence of disjoint chains $C_1, C_2, \ldots C_{|V_{out}|}$, containing all the flow holders and only them. The above allows us to propose the Algorithm 11 *PPOE-covering*.

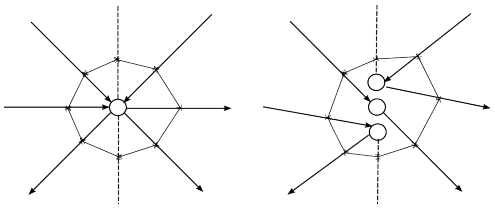

Figure 6. Example of the «correct» splitting, where $\to$ is flow hold arc, and $\dashrightarrow$ is arc without flow. (**a**) A vertex and arcs incident to it. (**b**) The «correct» splitting.

Algorithm 11 Algorithm *PPOE-covering*

Require: plane graph $G(V,F,E)$ without bridges, defined for all $e \in E(G)$
functions $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1,2$ (see Theorem 1) rank(e),
functions rank(v), $v \in V(G)$, rank(f), $f \in F(G)$;
subsets $V_{out}, V_{in} \subset V: |V_{in}| \geq |V_{out}|$;
subset $\{F_{in} \subset F\}$ of faces that allow piercing.
Ensure: PPOE-cover of graph $G(V,E)$: $\tilde{C}_1, \tilde{C}_2, \ldots \tilde{C}_{|V_{out}|}, C_{|V_{out}|+1}, \ldots, C_M$.

1: Construct a network $N(V \cup \{w\}, A \cup (\{w\} \times V_{in}))$.
2: **if** $(V_{out} = \emptyset)$ **then**
3: Run Algorithm 5 `OptimalCover` for graph G
4: **Return**(Cover of graph G by OE-chains C_j, $j = 1, ..., |V_{odd}(G)|/2$)
5: **end if**
6: **if** problem (2)–(7) is unsolvable **then**
7: **Rteurn**(*PPOE*-cover does not exist)
8: **else**
9: Let $x: A(N) \to \{0,1\}$ be optimal solution of problem (2)–(7)
10: For each active arc $(u,v): x(u,v) = 1$ create a list, including this arc and only it
11: Find a sequence of disjoint chains $C_1, C_2, \ldots C_{|V_{out}|}$, containing all the flow holders and only them with usage for each vertex $v \in V(G)$ of «correctly» splitting to the dummy vertices (see Figure 6)
12: Construct a partial graph

$$\tilde{G} = G \setminus \left(\bigcup_{i=1}^{|V_{out}|} C_i \right), E(\tilde{G}) = \left(E(G) \setminus \left(\bigcup_{i=1}^{|V_{out}|} C_i \right) \right),$$

in which all vertices $v \in V_{out}$, for which piercing is forbidden, are the vertices of even degree.
13: **end if**
14: For $\tilde{G}$ run algorithm `OptimalCover` 5. The result of this step is a sequence of disjoint chains

$$C_{|V_{out}|+1}, \ldots, C_{|V_{out}|+|M|}.$$

15: **Return**($\tilde{C}_1, \tilde{C}_2, \ldots \tilde{C}_{|V_{out}|}, C_{|V_{out}|+1}, \ldots, C_M$.)
16: **End of algorithm**

Theorem 11. *Algorithm PPOE-*`covering` *solves the problem of constructing PPOE-cover for a plane graph $G(V, E)$ without bridges by the time not exceeding $O(|V|^3)$.*

Proof. The route consisting of chains C_1, C_2, $\dots C_{|V_{out}|}$ is the edge-disjoint *OE*-route (due to unit carrying capacity of arcs). Partial graph $\tilde{G}$ does not contain any edges belonging to chains C_i, $i = 1, \dots, |V_{out}|$ by definition. All graph $\tilde{G}$ vertices avoiding piercing have even degree due to constructions. As a result of running Step 9, we get the continuation

$$C_{|V_{out}|+1}, \dots, C_{|V_{out}|+|M|}$$

of route which is the *OE*-route in graph $\tilde{G}$ covering all edges of graph $\tilde{G}$, and starting vertex $v \in V_{in}$ of each chain C_i, $i = |V_{out}| + 1, \dots |V_{out}| + |M|$ is permissible for piercing. Hence, the route

$$C_1, C_2, \dots C_{|V_{out}|}, C_{|V_{out}|+1}, \dots, C_{|V_{out}|+|M|}$$

is *PPOE*-cover of initial graph *G*.

Let us estimate the computing complexity of this algorithm. Step 1 allows to get the network by time $O(|E|)$. Step 2 verifies the condition and it is completed in $O(1)$. Circulation in step 3 may be obtained by time not exceeding $O(|V|^3)$ [34]. Step 4 verifies the condition and it is completed in $O(1)$. In step 5, we introduce a sequence of chains along with a set of active arcs. This operation is performed at a time not exceeding $O(|E|)$. In step 6, at each vertex v, a "merging" of lists is performed in a time not exceeding $O(|V| \cdot \deg(v))$. Thus, the computing complexity of step 6 does not exceed the value $O(|V| \cdot |E|)$. Step 7 runs by time not exceeding $O(|E|)$. The complexity of Step 7 is defined by the complexity of algorithm *OE-*`Cover` [19] and amounts to $O(|E(G)| \cdot \log_2 |V(G)|)$. Obtaining the partial graph $\tilde{G}$ at Step 7 claims the time not exceeding $O(|E|)$. The complexity of Step 9 does not exceed $O(|V|^3)$ used for the shortest matching obtaining. Thus, the complexity of algorithm *PPOE-*`routing` does not exceed the value of $O(|V|^3)$. □

Let us consider the application of algorithm *PPOE-Routing* to cutting plan in Figure 2 with geomorphic image presented in Figure 3 and in Table 1. We have $V_{out}(N) = \{v_1, v_5, V_{11}\}$, $V_{in}(N) = \{v_2, v_3, v_7, v_9\}$, $V(N) \backslash (V_{out}(N) \cup V_{in}(N)) = \{v_4, v_6, v_8, v_{10}, v_{12}\}$. Figure 7 demonstrates network $N(V \cup \{w\}, A \cup (\{w\} \times V_{in}))$ constructed in **Step 1** and optimal solution of problem (1)–(7) found in **Step 2**. Bold lines highlight carriers of non-zero flow.

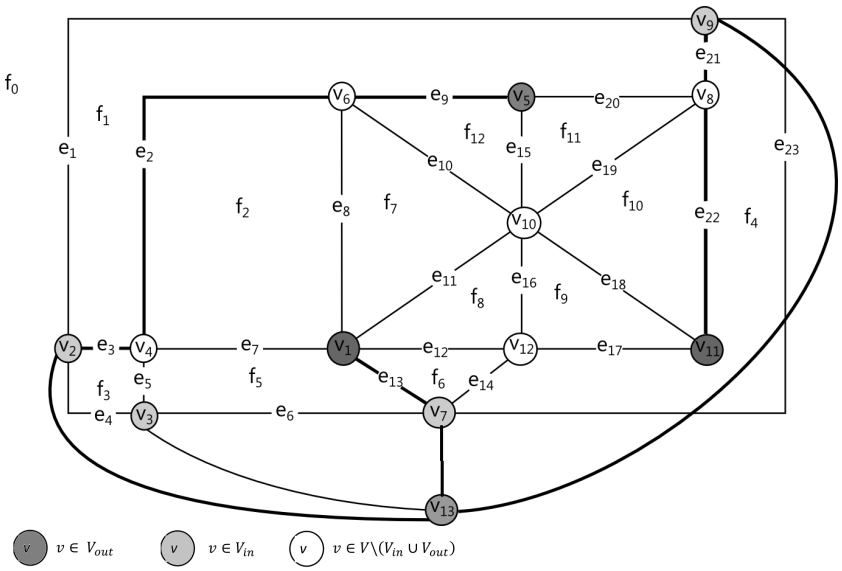

Figure 7. Network *N* for graph in Figure 3.

After running steps 3–6, we obtain the chains $C_1 = e_3e_2e_9$, $C_2 = e_{21}e_{22}$, $C_3 = e_{13}$. Partial graph constructed by step 7 is shown in Figure 8.

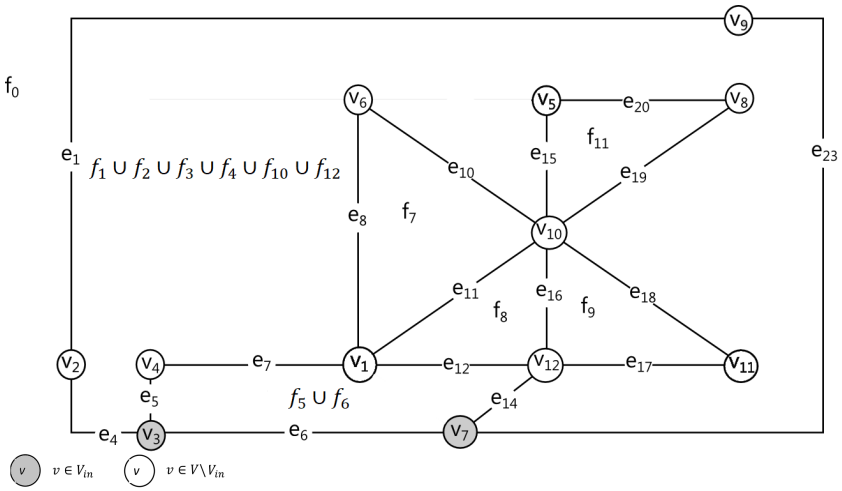

Figure 8. Rest of network N after running steps 3–6.

In this graph algorithm M-Cover constructs the only chain

$$C_4 = e_5e_7e_{11}e_{16}e_{12}e_8e_{10}e_{15}e_{20}e_{19}e_{18}e_{17}e_{14}e_{23}e_1e_4e_6.$$

So, the $PPOE$-cover for this graph is $\{C_1, C_2, C_3, C_4\}$.

Thus, the construction of the $PPOE$ -cover of the G graph allows us to solve the problems of the cutter movement routing for a realizable cutting plan with restrictions on possible pierce points.

2.4. Algorithms for Disconnected Graphs

The problem of constructing OE-routes in disconnected graphs is also of practical value. In this case, the task of finding the OE-covering of the graph by chains can be reduced to several tasks of lower dimension (to construct a cover for each connected component separately). This approach is reasonable if the resulting graph does not contain nested components. However, in the presence of nesting of the connected components, the problem becomes somewhat more complicated and the following restrictions on the order of traversing the connected components arise: the connected components consisting of edges of higher rank must be traversed before the components consisting of edges of lower rank. To solve the problem in common for plane disconnected graph algorithms `MultiComponent` (computing complexity $O(|E(G)| \cdot \log_2 |V(G)|)$), `Bridging`, `DoubleBridging`) and `FaceCutting` (Figure 9) are developed.

The proofs of these results [19] are constructive and, in fact, are reduced to describing and proving the effectiveness of algorithms for constructing the desired cycles (routes).

Algorithms `Bridging` and `DoubleBridging` use the approach of reducing the initial disconnected graph to a connected one.

Definition 15. *Let face $f \in F(G)$ be called separating, if it is incident to two or more connected components.*

Let graph $\tilde{G}$ be obtained from graph G by adding bridges belonging to separating faces between the components. Obviously, the obtained graph $\tilde{G}$ be a plane connected graph and it is possible to construct the OE-route $R(\tilde{G})$ for it. This OE-route $R(G)$ can be obtained from route $R(\tilde{G})$ if vertices incident to the introduced bridges are to be the ends of the current chain and the beginnings of the next ones (i.e., introduced bridges are the idle passes).

Let us consider the way of constructing the bridges connecting graph G and having a minimal summary length (see Algorithm 12).

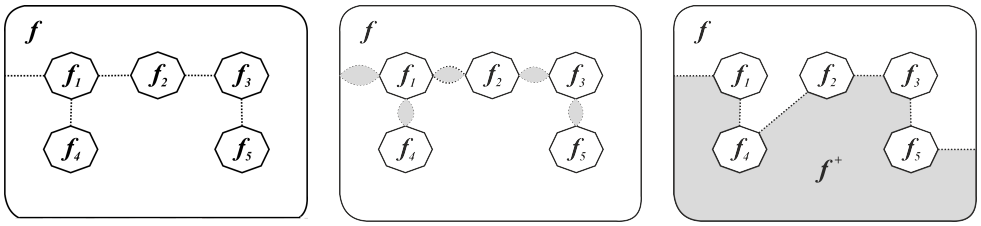

Figure 9. Examples of combining separated components. (**a**) Bridging; (**b**) DoubleBridging; (**c**) Face-Cutting.

Algorithm 12 `Bridging`

Require: plane disconnected graph G
Ensure: plane connected graph $\tilde{G}$ and set B of introduced bridges
1: $\tilde{G} := G$; $B = \emptyset$;
2: Define the set C_F of all separating faces.
3: **for all** $f \in C_F$ **do**
4: Find the set $S(f)$ of connected components of graph G incident to face f.
5: Construct the full abstract graph $\mathcal{T}$ the vertices of which are the components $S(f)$, and lengths of edges are equal to the distance between the components.
6: Obtain the minimal spanning tree $T(\mathcal{T})$ in $\mathcal{T}$.
7: Add the edges of the obtained spanning tree to graph $\tilde{G}$: $E(\tilde{G} := E(\tilde{G}) \cup E(T(\mathcal{T}))$, $B := B \cup E(T(\mathcal{T}))$.
8: **end for**
9: **end**

Plane graph $\tilde{G}$ obtained by algorithm `Bridging` contains bridges, hence it is possible to apply only algorithm `CPP_OE` [30] constructing the Chinese postman OE-route for plane graph [30]. Note that both the `OECover` algorithm and the M-`Cover` algorithm require no bridges in the graph. To avoid errors in the execution of the algorithms, it is necessary to add the edges of the spanning tree $T(\mathcal{T})$ to the graph twice (see line 6 of the Algorithm 12). The algorithm adjusted in this way is called `DoubleBridging`. The complexity of the `Bridging` and `DoubleBridging` algorithms is polynomial, depending on the method used to determine the distances between the connectivity components. If the distances are given it can be estimated as $O(|E(G)| \cdot \log |V(G)|)$.

Theorem 12. *If, for each component G_k of graph G, the degrees of vertices incident to separating faces of G are even, then the path of the minimal length of additional edges can be realized by algorithm* `DoubleBridging`.

Algorithm `FaceCutting` [35] is one other way to obtain graph $\tilde{G}$ without bridges. It consists in splitting the separating face using the Hamiltonian cycle. In fact, if for abstract graph $\mathcal{T}$ we use the minimal weight Hamiltonian cycle $H(\mathcal{T})$ instead of spanning tree $T(\mathcal{T})$, then the resulting graph $\tilde{G}$ does not contain any bridges and, hence, we can use algorithm M-`Cover` (see Algorithm 4) to obtain the OE-route with a minimal length of idle passes.

3. Discussion

In our article, we provide an overview of the plane graph routing problem statements since the 1980s. In the early works [20,21], the formalization of the tasks posed is not accurate enough, there is no classification of routes by the type of restrictions imposed. In this regard, mainly heuristic algorithms were developed. In the works of the authors (2007–2020), an exact formalization of some previously considered statements is given, classes of routes in plane graphs satisfying constraints are introduced, and polynomial exact algorithms for solving these problems are described.

Of particular note is the proof of the $\mathcal{NP}$-completeness of the problem of finding non-intersecting Euler chains [21], where the author takes an A-chain as a self-non-intersecting Euler chain. The authors have shown the existence of a polynomial algorithm for finding an Euler self-non-intersecting chain in a plane graph. The question of $\mathcal{NP}$-completeness of the problem of finding a self-non-intersecting chain remains open.

4. Conclusions

In our paper, we introduced a class of routes with ordered enclosing (OE-routes) in plane graphs. The routes of this class represent an ordered sequence of paths that satisfy the requirement that the inner edges of the traversed part of the route do not intersect with the edges of its non-traversed part.

We showed that the presence of bridges in the graph has a significant effect on the cover cardinality. If there are no bridges in a graph, the minimum number of OE-chains covering the graph is equal to the minimum number of paths covering the given graph (in the case of the existence of vertices of odd degree incident to the outer face). If there are no vertices of odd degree adjacent to the outer face, then the cardinality of the covering is one higher than the minimum number of paths covering the given graph. In general, the cardinality N of Euler OE-cover of graph G satisfies the inequality $k = \frac{|V_{odd}(G)|}{2} \leq N \leq |V_{odd}(G)| = 2k$. The upper and lower bounds are reachable.

We discussed the polynomial algorithms for constructing the OE-cover for different cases: plane Euler graph, any plane connected graph (CPP and OE-cover constructing problems), any disconnected plane graph. We developed the polynomial time algorithm for obtaining an OE-route with the minimum number of covering paths and an algorithm for constructing an OE-route with a minimum length of transitions between the end of the current path and the beginning of the next path.

To discover the chains with the complex groups of restrictions we introduced classes of AOE-chains, NOE-chains, and $PPOE$-chains. (1) Class of AOE-chains includes the chains with additional local restriction according to which the neighbouring edges need to satisfy the transitions system of A-chain. Algorithm `AOE-CHAIN` allows obtaining a chain belonging to class AOE for a plane connected 4-regular graph. The algorithm allows to obtain it by the time $O(|E(G)| \cdot \log |V(G)|)$. (2) Class of NOE is the extension of class AOE and contains all OE-chains with non-intersecting transitions. Algorithm `Non-intersecting` allows obtaining such a chain. Its implementation consists in reducing the original plane graph to a plane connected 4-regular graph by splitting vertices of degree higher than 4 and further executing the `AOE-CHAIN` algorithm. (3) Class of $PPOE$-chains contains OE-chains with fixed sets of starting and ending vertices, and algorithm $PPOE$`-covering` allows for correctly solving the problem of constructing this type of cover for a plane graph $G(V, E)$ without bridges by the time not exceeding $O(|V|^3)$ [36].

All our algorithms are implemented using the C++ programming language, and the initial data can be read either from text files with the table of functions for edges (see Theorem 1) or by conversion of JSON-files used in known CAD/CAM systems to the table of these functions [36].

Author Contributions: Conceptualization, T.M. and A.P.; Formal analysis, A.P.; Methodology, T.M.; Software, T.M.; Validation, A.P.; Writing—original draft, T.M.; Writing—review & editing, A.P. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported by Act 211 Government of the Russian Federation, contract No. 02.A03.21.0011. The work was supported by the Ministry of Science and Higher Education of the Russian Federation (government order FENU-2020-0022).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Velayutham, K.; Waran, V.; Gurusamy, S. Optimisation of laser cutting of SS 430 plate using advanced Taguchi entropy weighted-based GRA methodology. *Int. J. Mechatron. Manuf. Syst.* **2018**, *11*, 148. [CrossRef]
2. Liang, F.; Kang, C.; Fang, F. Tool path planning on triangular mesh surfaces based on the shortest boundary path graph. *Int. J. Prod. Res.* **2021**, 1–20. [CrossRef]
3. Eapen, N.A.; Heckendorn, R.B. Cutting path optimization for an automatic cutter in polynomial time using a 3/2 approximation algorithm. *Int. J. Adv. Manuf. Technol.* **2021**, *113*, 3667–3679. [CrossRef]
4. Xie, S.; Gan, J. Optimal process planning for compound laser cutting and punch using Genetic Algorithms. *Int. J. Mechatron. Manuf. Syst.* **2009**, *2*, 20–38. [CrossRef]
5. Jing Y.; Chen, Z.C. An Optimized Algorithm of Numberical Cutting-Path Control in Garment Manufacturing. *Adv. Mater. Res.* **2013**, *796*, 454–457. [CrossRef]
6. Lee, M.K.; Kwon, K.B. Cutting path optimization in CNC cutting processes using a two-step genetic algorithm. *Int. J. Prod. Res.* **2006**, *44*, 5307–5326. [CrossRef]
7. Hoeft, J.; Palekar, U.S. Heuristics for the plate-cutting traveling salesman problem. *IIE Trans.* **1997**, *29*, 719–731. [CrossRef]
8. Kochetkov, S.A.; Utkin, V. Method of decomposition in mobile robot control. *Autom Remote Control* **2011**, *72*, 2084–2099. [CrossRef]
9. Arkhipov, D.; Battaia, O.; Lazarev, A. Long-term production planning problem: Scheduling, makespan estimation and bottleneck analysis. *IFAC-PapersOnLine* **2017**, *50*, 7970–7974. [CrossRef]
10. Fleischner, H. Eulerian Graphs and Related Topics. Part 2. *Ann. Discret. Math.* **1991**, *50*, 336.
11. Kartak, V.M.; Ripatti, A.V.; Scheithauer, G.; Kurz, S. Minimal proper non-IRUP instances of the one-dimensional Cutting Stock Problem. *Discret. Appl. Math.* **2015**, *187*, 120–129. [CrossRef]
12. Tavaeva, A.; Petunin, A.; Ukolov, S.; Krotov, V. A Cost Minimizing at Laser Cutting of Sheet Parts on CNC Machines. In *Mathematical Optimization Theory and Operations Research, Proceedings of the 18th International Conference, MOTOR 2019, Ekaterinburg, Russia, 8–12 July 2019;* Springer International Publishing: New York, NY, USA, 2019; Volume 1090, pp. 422–437. [CrossRef]
13. Petunin, A.A.; Polishchuk, E.G.; Ukolov, S.S. On the new Algorithm for Solving Continuous Cutting Problem. *IFAC-PapersOnLine* **2020**, *52*, 2320–2325. [CrossRef]
14. Chentsov, A.G.; Khachay, M.Y.; Khachay, D.M. Linear Time Algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem. *IFAC-PapersOnLine* **2016**, *49*, 651–655. [CrossRef]
15. Chentsov, A.G.; Grigoryev, A.M.; Chentsov, A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme. *Bull. South Ural State Univ. Ser. Math. Model. Program. Comput. Softw.* **2018**, *11*, 60–74. [CrossRef]
16. Landovskaya, I. A processing algorithm of fabric particle interaction with solid object faces during computer simulation. *Proc. Russ. Higher Sch. Acad. Sci.* **2016**, *2*, 78–93. (In Russian) [CrossRef]
17. Dewil, R.; Vansteenwegen, P.; Cattrysse, D.; Laguna, M.; Vossen, T. An improvement heuristic framework for the laser cutting tool path problem. *Int. J. Prod. Res.* **2015**, *53*, 17611776. [CrossRef]
18. Dewil, R.; Vansteenwegen, P.; Cattrysse, D. A review of cutting algorithms for laser cutters. *Int. J. Manuf. Technol.* **2016**, *87*, 1865–1884. [CrossRef]
19. Panyukov, A.V.; Makarovskikh, T.A.; Savitskiy, E.A. Mathematical models and routing algorithms for economical cutting tool paths. *Int. J. Prod. Res.* **2018**, *56*, 1171–1188. [CrossRef]
20. Manber, U.; BentIsrani, S. Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting. *J. Manuf. Syst.* **1984**, *3*, 1. [CrossRef]
21. Manber, U.; Bent, S.W. On Non-intersecting Eulerian Circuits. *Discret. Appl. Math.* **1987**, *18*, 87–94.
22. Fleischner, H.; Beineke, L.; Wilson, R. *Selected Topics in Graph Theory. Part 2*; Academic Press: London, UK, 1983.
23. Bely, S. On self-non-intersecting and non-intersecting chain. *Math. Notes* **1983**, *34*, 625–628.
24. Panioukova, T. Eulerian cover with ordered enclosing for flat graphs. *Electron. Notes Discret. Math.* **2007**, *28*, 17–24. [CrossRef]
25. Makarovskikh, T.A.; Panyukov, A.V. The Cutter Trajectory Avoiding Intersections of Cuts. *IFAC PapersOnLine* **2017**, *50*, 2284–2289. [CrossRef]
26. Makarovskikh, T.A.; Panyukov, A.V. Mathematical model for a cutting path avoiding intersections. *IFAC-PapersOnLine* **2020**, *53*, 10455–10460. [CrossRef]

27. Makarovskikh, T.A.; Panyukov, A.V. Construction of a Technologically Feasible Cutting with Pierce Points Placement Constraints. *Commun. Comput. Inf. Sci.* **2020**, *1340*, 186–197. [CrossRef]
28. Panioukova, T. Chain sequences with ordered enclosing. *J. Comput. Syst. Sci. Int.* **2007**, *46*, 83–92. [CrossRef]
29. Panioukova, T.A.; Panyukov, A.V. Algorithms for Construction of Ordered Enclosing Traces in Planar Eulerian Graphs. In Proceedings of the International Workshop on Computer Science and Information Technologies' 2003, Ufa, Russia, 16–18 September 2003; Ufa State Technical University: Ufa, Russia, 2003; Volume 1, pp. 134–138.
30. Panyukova, T.A. Constructing of OE-postman Path for a Planar Graph. *Bull. South Ural State Univ. Ser. Math. Model. Program. Comput. Softw.* **2014**, *7*, 90–101. [CrossRef]
31. Fleischner, H. *Eulerian Graphs and Related Topics*; Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990; 450p.
32. Makarovskikh, T.A.; Panyukov, A.V. *AOE*-Trails Constructing for a Plane Connected 4-Regular Graph. In Proceedings of the Supplementary Proceedings of the 9th International Conference on Discrete Optimization and Operations Research and Scientific School (DOOR 2016), Vladivostok, Russia, 19–23 September 2016; Volume 1623, pp. 62–71.
33. Makarovskikh, T.A.; Panyukov, A.V. Algorithm for constructing *AOE* circuit in a connected flat 4-regular graph. In *Proceedings of the XII International Scientific Seminar "Discrete Mathematics and Its Applications" Academician Lupanov*; Mechanics and Mathematics Faculty of Moscow State University: Moscow, Russia, 2016, p. 293296. (In Russian)
34. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization. Algorithms and Complexity*, Unabridged edition; Dover Publications: Mineola, NY, USA, 1998.
35. Makarovskikh, T.A.; Panyukov, A.V.; Savitskiy, E.A. Mathematical Models and Routing Algorithms for CAD Technological Preparation of Cutting Processes. *Autom. Remote Control* **2017**, *78*, 868–881. [CrossRef]
36. Makarovskikh, T.; Panyukov, A.; Savitsky, E. Software Development for Cutting Tool Routing Problems. *Procedia Manuf.* **2019**, *29*, 567–574. [CrossRef]