MSC 05C38 DOI: 10.14529/mmp140407

CONSTRUCTING OF *OE*-POSTMAN PATH FOR A PLANAR GRAPH

T.A. Panyukova, South Ural State University, Chelyabinsk, Russian Federation, kwark@mail.ru

The model of cutting plan can be presented as a planar graph for automated system of sheet material cutting process preparation. The aim of such modelling is a definition of the shortest path of a cutter having no parts requiring any additional cuttings. The paper is devoted to a problem of chines postman path constructing for a planar graph representing a cutting plan. This path has a restriction of ordered enclosing (i.e. cycle of passed edges does not contain inside not passed ones). The path satisfying this restriction is also called *OE*-path. This kind of restriction means the lack of additional cuttings of details. The recursive algorithm for constructing of this type of paths is considered in the paper. It is proved that this algorithm has a polynomial complexity. The developed software allows to solve the problem for an arbitrary planar graph. The software is tested for the typical cases of planar graphs.

Keywords: planar graph; Chinese postman problem; path; ordered enclosing; algorithm; software.

Introduction

Great number of industrial branches is dealing with cutting the material such as metal, wood, plywood, glass and others. These materials are presented in industrial flow as sheets, boards, pipes, profiled rolls etc. Obviously, the usage of these materials implies their separation (or cutting) on parts of the given size and form (the samples of some details).

That is why the significance of industrial cutting is a source of economy and it is mentioned in technical literature and some industrial journals [1]. Theoretically this field is rather unexplored. There are some researches devoted to maximization of wood volume at lumbering process. The well known problem of circles placement on a plane is also similar to a cutting problem for an infinite sheet and equal round billets.

The experience of advanced engineering plants shows that accurate planning of cutting process allows to achieve the economy of materials [1].

Nowadays the problems of creation of the high-effective technologies of social sphere development, flexible automated enterprises on the base of information technologies, particularly, clothes production for individuals, particularly, the development of software for personified high quality clothes projecting for individual customer are actual. The brunch connected with flexible automated enterprises of consumer goods is officially called as one of prior branches of science development.

The interest to routing problems can be explained by their usage as mathematical models of different control problems and automation of projection. Particularly, for automated system of sheet material cutting process preparation the model of cutting plan can be presented as a planar graph. The aim of such modelling is a definition of the shortest path of a cutter. This path is to have no parts requiring any additional cuttings. However, to solve this problem one needs a formal statement in terms of a problem of

path constructing as a planar graph. Due to the lack of a statement the algorithms of such rational trajectories definition should be developed.

One of the researches on special problems on graph theory is the monograph by Herbert Fleischner "Eulerian Graphs and Related Topics" [2,3] where some special types of Eulerian trails (for example, trails without forbidden transitions, A-trails, pairwise-compatible trails) are considered in rather detailed and systematised way.

There are also some papers of other authors where some problems on special type Eulerian trails are considered. For example, For example, the extension of forbidden transitions class [4], non-intersecting trails, bidirectional double paths [2,3], Petrie walks [5], straight-ahead walks [6], and edge-ordered paths [7] are among them.

Let's solve the problem of constructing of the Eulerian trail with considered restriction as a cutting problem. Let the model of cutting sheet be presented as a plane S, and the model of cutting plan be a planar graph G with outer face f_0 on plane S. For any part $J \subseteq G$ of this graph (a part of cutter movement trajectory) let's define by $\mathsf{Int}\,(J)$ a set-theoretic union of its inner faces (the union of all its connected components $S \setminus J$ not containing outer face). Then $\mathrm{Int}(J)$ can be interpreted as a part cut off a sheet. Let the sets of vertices and edges of graph J be denoted by $V(J)$ and $E(J)$ correspondingly, and $|M|$ be a number of set M elements.

Let any path in graph G be considered as a part of graph consisting of all vertices and edges belonging to a path. This allows to formalize requirement to a cutter path as no intersection of inner faces of any initial path part for a fixed graph G with unpassed edges [8]. Let's call such paths as ordered enclosing paths [9] (or OE-paths for short).

Definition 1. *A cycle $C = v_1e_1v_2e_2\ldots v_k$ of Eulerian graph G has an ordered enclosing (be an OE-cycle) if for any its initial part $C_i = v_1e_1v_2e_2\ldots e_i$, $i \leq (|E(G)|)$ the condition*

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

holds.

For example, let's consider a plane Eulerian graph on fig. 1. Cycle

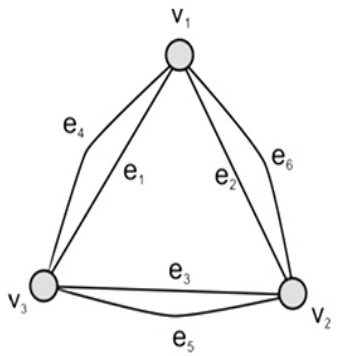


Fig. 1. Example of Eulerian graph

$v_1e_1v_3e_3v_2e_2v_1e_4v_3e_5v_2e_6v_1$ satisfies the condition of ordered enclosing and cycle $v_1e_4v_3e_5v_2e_6v_1e_1v_3e_3v_2e_2v_1$ does not because $\mathsf{Int}\,(v_1e_4v_2e_5v_3e_6v_1) \supset \{e_1, e_2, e_3\}$.

If a cutting plan is presented as plane Eulerian graph G then it can be passed without any additional (idle) cuts [10]. If this graph G is not Eulerian and has $2k$ odd vertices

then it is possible to use Listing–Luke algorithm [2, 3] to cover this graph by k trails. An algorithm for constructing of a cover with defined restrictions (OE-cover) is proposed in [11, 12]. For an arbitrary chosen planar graph G the problem of OE-path constructing can be considered in terms of Chinese postman problem where a path to be defined consists only of graph G edges. Let's consider this problem in details.

1. Algorithm Specification

Consider a planar graph $G = (V, E)$ and a problem of constructing of a path C on the set of its edges $E(G)$. To get a closed path we need some edges to be passed twice. Later we shall consider that edges passed twice are duplicated by ***additional*** edges belonging to set $H(G)$.

Let's show that algorithm for constructing of closed path P for non-Eulerian graph G be the algorithm for constructing of Eulerian OE-cycle C for Eulerian graph $\tilde{G}$ being the modification of G received by addition of edges from set $H(G)$.

Definition 2. *A path $C = v_1e_1v_2e_2 \dots v_k$ of graph G has an ordered enclosing (be an OE-path) if for any its initial part $C_i = v_1e_1v_2e_2 \dots e_i$, $i \le (|E(G)| + |H(G)|)$ the condition*

$$\mathrm{Int}(C_i) \cap (E(G) \cup H(G)) = \emptyset$$

holds. I.e. the intersection of inner faces C_i with set of edges is empty.

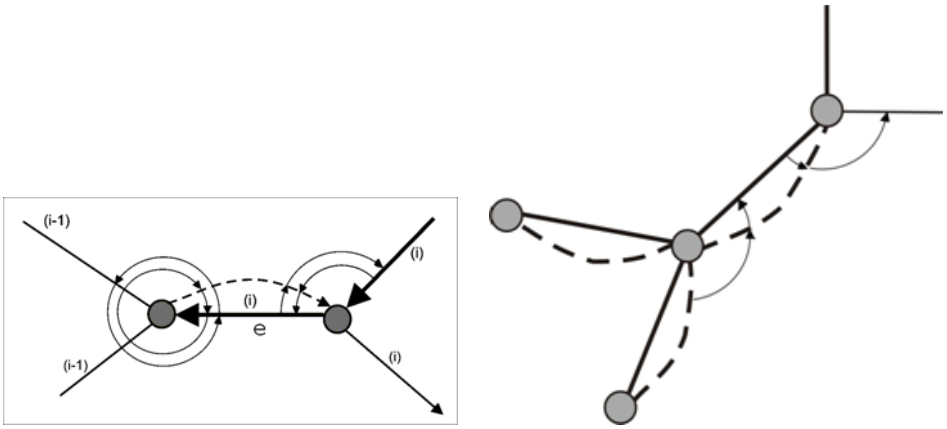
In terms of cutting problem (for model of cutting plan as plane non-Eulerian graph) the edges of set $H(G)$ are interpreted as idle passes of a cutter.

Using approach similar to Chinese Postman Problem one can define a set of edges to be duplicated. Obviously, the computing complexity of an algorithm may increase.

Here we shall add the duplicates in such a way that the recursive algorithm presented in [8] for Eulerian graphs will have less modifications. This modification can be as the following.

In the first part of algorithm when we need decision which edges bound the outer face it occurs that field $Mark$ pointing to a next edge in this cycle consists of edge pointing on itself (terminal vertex of a current component) or this edge does not coincide with the initial one for this component and points to marked edges (bridge). In both cases one needs duplication of these edges. After adding these edges as it is shown on fig. 2 a) for a terminal vertex and on fig. 2 b) for a bridge some pointers are to be modified. By the way, each edge $e \in E$ of graph G is presented by pointers (functions) $v_i(e)$ (vertices incident to e) and $l_i(e)$ (adjacent edges-neighbours laying first in rotation of e around $v_i(e)$), $i = 1, 2$.

After making such constructions the field $Mark$ on a first stage of algorithm execution is to be formed such a way that all other functions for Eulerian graphs algorithm do not require any modifications. So, after adding all idle edges one has Eulerian graph, and a cycle found in it is to be an OE-cycle. That is why we have a path in initial graph where any cycle of passed edges does not contain inside any unpassed ones. Let's call this modification of recursive algorithm as algorithm `CPP_OE` (see algorithm 5) (this algorithm solves the OE-Postman problem). The recursive call of function for each unmarked edge incident to vertices of a cycle received on a previous stage is made without changes. After constructing of a path for the considered component the path obtained is included to a resulting path.



а) б)

Fig. 2. Pointers modification after adding the edges: a) for a terminal vertex; b) for a bridge

Let's generalize all above considerations as a following theorem.

Theorem 1. *A path constructed by algorithm* `CPP_OE` *has to be an OE-path. The complexity of this algorithm is* $O(|E(G)| \cdot |V(G)|)$.

The **proof** of this theorem is obvious because all additional edges are to be lacking parts of cycles determined by algorithm. We need less than $O(|E(G)|)$ edges additions, and each addition needs changing of four pointers (fuctions).

Let's organize the first part of algorithm as function `ExternCycle` (see algorithm 1) to make algorithm more descriptive. **Input data** for this function are:

- the first edge of a graph (it's the edge from which the searching of edges bounding outer face of the current component starts);
- edge next to the starting one;
- the initial vertex for a currtnt component (for organising the correct edges orientation);
- some additional variables.

This function calls the process of edges addition in all considered above cases. The duplicating of a bridge occurs directly in a function `ExternCycle`, for the remained two cases of different modifications of function `Add` are called (see algorithm 3 and 4). The first one is used to drop-out the terminal vertex of a current component, and the second one is to finish a cycle in a current component.

The example of this algorithm execution is shown on fig. 3. The dashed line corresponds to the additional (idle) edges. The considered algorithm allows to find path

$$v_3v_1v_3v_7v_{10}v_8v_{10}v_7v_8v_6v_4v_6v_5v_6v_8v_9v_{11}v_{12}v_{11}v_{13}v_{11}v_9v_7v_3v_2v_1v_4v_5v_{12}v_{13}v_2.$$

Algorithm 1. ExternCycle (Part 1)

Require: $G = (V, E)$ be a planar graph where $\forall e \in E$ the functions $v_k(e)$ (vertices incident to e) and $l_k(e)$ (edges neighbouring to e received by its rotation counter-clockwise around $v_k(e)$), $k = 1, 2$; $First$ be the first considered edge; $Next$ be a pointer to the next edge; $Vertex$ be a current vertex; $Number$ be the number of edges in graph

Ensure: $NewFirst$ be the number of additional edge finishing a cycle

```
1: procedure ExternCycle(In: G = (V, E), First, Next, Vertex, Number; Out: NewFirst)
2:     NewFirst = 0;
3:     while true do
4:         First = Next; Vertex = v_1(First); Next = l_1(First);
5:         if Mark(Next) ≠ ∞ then
6:             if Next = Start then
7:                 if v_1(First) = v_1(Next) then
8:                     Add(G, Next, Mark(Next)); Number = Number + 1;
9:                     Mark(First) = Number; Mark(Number) = Next;
10:                    Level(First) = L; Level(Number) = L; NewFirst = Number;
11:                    return NewFirst;
12:                end if
13:                Mark(First) = Next; Level(First) = L; return NewFirst;
14:            else
15:                e = l_2(Mark(Next));
16:                if e ≠ Start then
17:                    while Mark(e) ≠ ∞ do
18:                        e = l_2(l_1(e));
19:                        if e = Start then
20:                            break;
21:                        end if
22:                    end while
23:                end if
24:                if e ≠ First then
25:                    if Mark(Next) ≠ ∞ and Level(Next) = L then
26:                        Number = Number + 1;
27:                        v_1(Number) = v_2(Next); v_2(Number) = v_1(Next);
28:                        l_1(Number) = l_2(Next); r_1(l_2(Next)) = Number;
29:                        r_1(Number) = Next; l_2(Next) = Number;
30:                        if v_1(r_1(Next)) = v_2(Number) then
31:                            l_1(r_1(Next)) = Number;
32:                        else
33:                            l_2(r_1(Next)) = Number;
34:                        end if
35:                        r_2(Number) = r_1(Next); r_1(Next) = Number; l_2(Number) = Next;
36:                        Next = Number;
37:                    end if
38:                    Next = e;
39:                else
40:                    Number = Number + 1; Add(G, Number, First);
41:                    Next = l_1(First);
42:                end if
43:            end if
44:        end if
```

Algorithm 2. ExternCycle (Part 2)

```
45:         if Vertex ≠ v2(Next) then
46:             REPLACE(Next);
47:         end if
48:         Mark(First) = Next; Level(First) = L;
49:         if Next = Start then
50:             break;
51:         end if
52:     end while
53:     return NewFirst;
54: end procedure
```

Algorithm 3. Add (Function for adding of an idle edge for a terminal vertex)

```
1: procedure ADD(In: G = (V, E) be a planar graph; Number be a number of additional (idle)
   edge; First be an edge leading to this terminal vertex)
2:     v1(Number) = v2(First); v2(Number) = v1(First);
3:     l1(Number) = l2(First); r1(Number) = First;
4:     if v1(l2(First)) = v2(First) then
5:         r1(l2(First)) = Number;
6:     else
7:         r2(l2(First)) = Number;
8:     end if
9:     l2(First) = Number;
10:    r2(Number) = First;
11:    if v1(r1(First)) = v1(First) then
12:        l1(r1(First)) = Number;
13:    else
14:        l2(r1(First)) = Number;
15:    end if
16:    r1(First) = Number; l2(Number) = First; l1(First) = Number;
17: end procedure
```

Algorithm 4. Add (Function of adding of the idle edge for finishing a cycle)

```
1: procedure ADD(In: G = (V, E) be a planar graph; Number be the number of added idle
   edge; Next be the edge belonging to an outer cycle)
2:     v1(Number) = v2(Next); v2(Number) = v1(Next);
3:     l1(Number) = Next; l2(Next) = Number; l2(Number) = Mark(Next);
4:     if v1(Mark(Next)) = v2(Number) then
5:         r1(Mark(Next)) = Number;
6:     else
7:         r2(Mark(Next)) = Number;
8:     end if
9:     r2(Number) = Next; r1(Number) = Next; r2(Next) = Number;
10: end procedure
```

Algorithm 5. CPP_OE (OE-chinese postman problem)

Require: $G = (V, E)$ be a planar graph; $Number$ be the number of graph vertices; $First$ be a first considered edge

Ensure: Queue $Mark$, the first its edge be $Ret.First$ and the last one be Ret.Last;

```
1: procedure CPP_OE(In: G = (V,E); Number; First; Out: Mark, Ret)
2:     for all e ∈ E do
3:         Mark(e) = ∞;
4:     end for
5:     Start = Next = First;
6:     NewFirst=ExternCycle (G, Start, Next, First, Vertex, Number); Mst = 0;
7:     while true do
8:         if l2(Next) ≠ First and Mark(l2(Next)) = ∞ then
9:             if Mst = 0 then
10:                Mst = l2(Next);
11:            end if
12:            if v ≠ v2(l2(Next)) then
13:                REPLACE(l2(Next));
14:            end if
15:            Ret=CPP_OE (G, l2(Next), Number);
16:            if Mark(First) ≠ ∞ then
17:                Mark(Ret.Last) = Mark(First);
18:                if v2(Ret.First) = v1(First) then
19:                    Mark(First) = l2(Next);
20:                else
21:                    Mark.First = l2(Next);
22:                end if
23:            end if
24:            First = Next; Next = Mark(First); Vertex = v1(e);
25:            if Next = Ret.First or Next = Start then
26:                break;
27:            end if
28:        end if
29:    end while
30:    if Mst = 0 then
31:        Ret.First = Start;
32:    else
33:        if v2(Ret.First) ≠ v1(First)and NewFirst = 0 then
34:            Ret.First = Mst;
35:        end if
36:        if v2(Ret.First) ≠ v1(First)and NewFirst ≠ 0 then
37:            Ret.First = Next;
38:        end if
39:    end if
40:    if NewFirst = 0 then
41:        Ret.Last = First;
42:    else
43:        Ret.Last = NewFirst;
44:    end if  return Ret;
45: end procedure
```

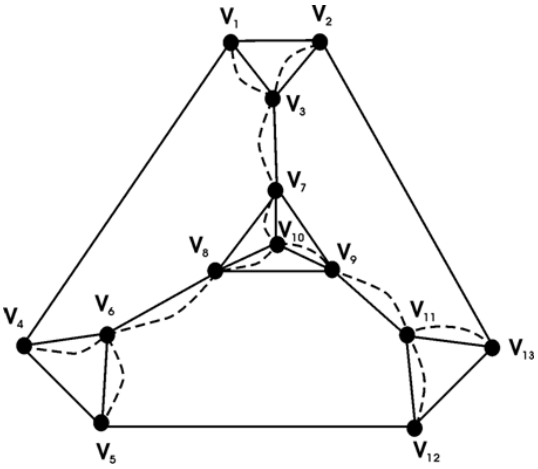


Fig. 3. The example of algorithm execution for a plane non-Eulerian graph

This algorithm allows to define $|\tilde{V}|$ different paths for a given graph G. Here $\tilde{V}$ is a set of graph G vertices adjacent to outer face of this graph. In fact it is only the low bound for number of solutions. However this algorithm allows to define only $|\tilde{V}|$ of different solutions due to certainty of choosing of the next edge of the path.

2. The Software for Constructing of Postman Path with Ordered Enclosing

Consider some examples demonstrating how the developed algorithm runs for non-Eulerian graphs. Figure 4 demonstrates the simplest example when graph has only two odd vertices (vertices 2 and 4) and both of them are adjacent to the outer face.

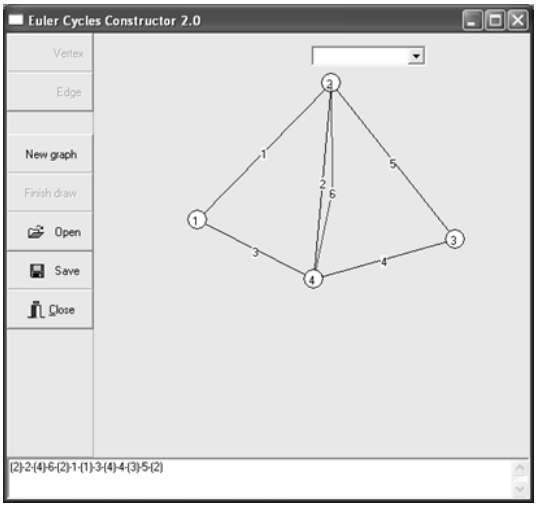


Fig. 4. One of the simplest examples of planar non-Eulerian graph

The additional (idle) edges are shown by a polyline. The significance of these idle edges is laid at a stage of model projecting (in terms of cutting problem they are understood,

for example, as idle way of a cutter), not at a stage of solution finding. In a bottom of the window we see the defined path. Numbers of vertices are embraced.

For graph on fig. 4 algortighm constructs the additional edge 6 connecting vertices 2 and 4.

Consider the example of graph having no even vertices (see fig. 5).

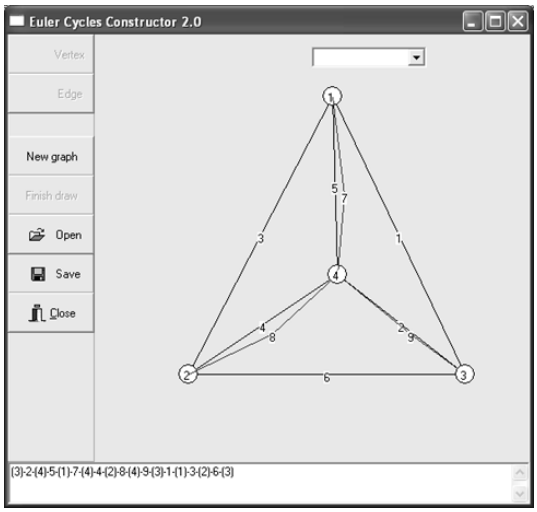


Fig. 5. Raph without even vertices

Obviously, after marking the outer cycle of edges 1, 3, and 6 we have a subgraph with three terminal vertices. To get an inner cycle we need a construction of three additional edges 7, 8, and 9. Such a construction is not optimal by the number of added edges because graph with four odd vertices needs only two additional edges to become Eulerian. Nevertheless, the received construction allows to get Eulerian graph while algorithm `CPP_OE` is running.

Consider more complex example (fig. 6) where one needs duplication of edges (that are bridges) not adjacent to exterior face.

Note that the condition of ordered enclosing holds because idle edges are the lacking parts of the inner cycles, and as the received Eulerian graph has an OE-cycle then a path where some edges are replaced by idle ones also has ordered enclosing.

Conclusion

The considered algorithm allows to find one of possible postman paths satisfying the condition of ordered enclosing corresponding to a chosen initial vertex. Algorithm has polynomial complexity. The developed software allows to solve the problem for an arbitrary planar graph. The software is tested for the typical cases of planar graphs. In all of these cases graph was completed to become Eulerian using considered above functions and OE-path for initial graph is defined.

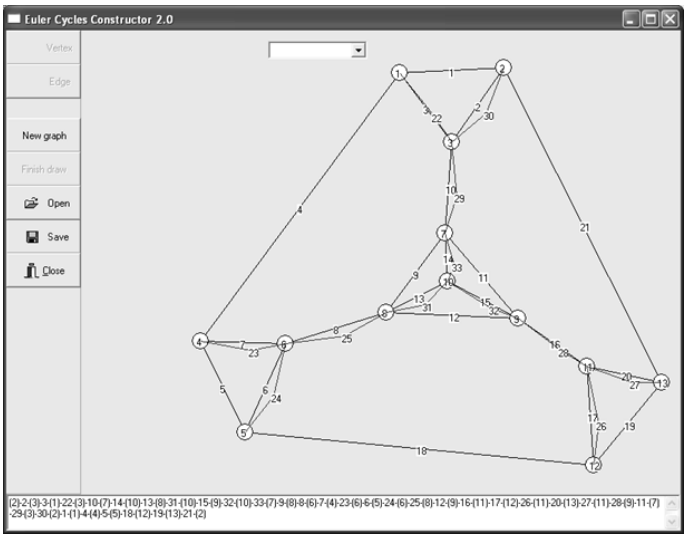


Fig. 6. Plane non-Eulerian graph where there are no planar matching of odd vertices

References

1. Kantorovich L.V., Zalgaller V.A. *Ratzionalniy raskroy promyshlennikh materialov* [Rational Cutting of Industrial Materials]. St.Peterburg, 2012. 304 p.
2. Fleischner H. *Eulerian Graphs and Related Topics. Part 1, vol. 1.* Ann. Discrete Mathematics, 1990, no. 48. 420 p.
3. Fleischner H. *Eulerian Graphs and Related Topics. Part 1, vol. 2.* Ann. Discrete Mathematics, 1991, no. 50. 356 p.
4. Szeider S. Finding Paths in Graphs Avoiding Forbidden Transitions. *Discrete Applied Mathematics*, 2003, no. 126, pp. 261–273. DOI: 10.1016/S0166-218X(02)00251-2
5. Zitnik A. planar graphs with Eulerian Petrie Walks. *Discrete Mathematics*, 2002, vol. 244, pp. 539–549. DOI: 10.1016/S0012-365X(01)00061-9
6. Pisanski T., Tucker T.W., Zitnik A. Straight-Ahead Walks in Eulerian Graphs. *Discrete Mathematics*, 2004, no. 281, pp. 237–246. DOI: 10.1016/j.disc.2003.09.011
7. Chebikin D. On k-Edge-Ordered Graphs. *Discrete Mathematics*, 2004, no. 281, pp. 115–128. DOI: 10.1016/j.disc.2003.09.004
8. Panioukova T.A., Panyukov A.V.Algorithms for Construction of Ordered Enclosing Traces in Planar Eulerian Graphs. *The International Workshop on Computer Science and Information Technologies' 2003. Proceedings of Workshop, Ufa, September 16 – 18, 2003*, Ufa, Ufa State Technical University, 2003, vol. 1, pp. 134–138.
9. Panyukova T. Chain Sequences with Ordered Enclosing. *Journal of Computer and System Sciences International*, 2007, vol. 46, no. 1, pp. 83–92. DOI: 10.1134/S1064230707010108
10. Panyukov A.V., Panioukova T.A. The Algorithm for Tracing of Flat Euler Cycles with Ordered Enclosing. *Proceedings of Chelyabinsk Scientific Center*, 2000, no. 49), pp. 18–22. Available at: http://elibrary.ru/download/18130929.pdf (accessed November 20, 2000).

11. Panyukova T.A. [Trails with Ordered Enclosing for planar graphs]. *Diskretny analys i issledovaniye operaziy* [Discrete Analysis and Operation Research], 2006, part 2, vol. 13, no. 2, pp. 31–43. (in Russian)

12. Panyukova T.A. [Optimal Eulerian Covers for planar graphs]. *Diskretny analys i issledovaniye operaziy* [Discrete Analysis and Operation Research], 2011, vol. 18, no. 2, pp. 64–74. (in Russian)

Received August 15, 2014

УДК 519.178 **DOI: 10.14529/mmp140407**

ПОСТРОЕНИЕ *OE*-МАРШРУТА КИТАЙСКОГО ПОЧТАЛЬОНА В ПЛОСКОМ ГРАФЕ

Т.А. Панюкова

При автоматизированной подготовке процесса раскроя раскройный план можно представить в качестве плоского графа. Целью такого моделирования является определение кратчайшего пути режущего инструмента, при условии, что отрезанная от листа часть не требовала бы дополнительных разрезаний. В статье рассматривается задача построения пути китайского почтальона в плоском графе, являющемуся моделью раскройного плана. На этот путь наложено условие упорядоченного охватывания (т.е. цикл из пройденных ребер не охватывает еще не пройденных). Такой путь еще будем называть *OE*-путем. Данное ограничение и означает отсутствие дополнительных разрезаний для деталей. В статье рассматривается рекурсивный алгоритм построения таких цепей. Доказано, что алгоритм имеет полиномиальную сложность. Разработанное программное обеспечение позволяет решить задачу для произвольного плоского графа. Программа протестирована для различных типов плоских графов.

Ключевые слова: плоский граф; задача китайского почтальона; маршрут; упорядоченное охватывание; алгоритм; программная реализация.

Литература

1. Канторович, Л.В. Рациональный раскрой промышленных материалов / Л.В. Канторович, В.А. Залгаллер. – СПб.: Невский Диалект, 2012. – 304 с.

2. Фляйшнер, Г. Эйлеровы графы и смежные вопросы / Г. Фляйшнер. – М.: Мир, 2002. – 335 с.

3. Fleischner, H. Eulerian Graphs and Related Topics. Part 1, V. 2 / H. Fleischner. – Ann. Discrete Mathematics, 1991. – № 50. – 356 p.

4. Szeider, S. Finding Paths in Graphs Avoiding Forbidden Transitions / S. Szeider // Discrete Applied Mathematics. – 2003. – № 126. – P. 261–273.

5. Zitnik, A. Planar graphs with Eulerian Petrie walks / A. Zitnik // Discrete Mathematics. – 2002. – V. 244. – P. 539–549.

6. Pisanski, T. Straight-ahead walks in Eulerian graphs / T. Pisanski, T.W. Tucker, A. Zitnik // Discrete Mathematics. – 2004. – № 281. – P. 237–246.

7. Chebikin, D. On k-edge-ordered graphs / D. Chebikin // Discrete Mathematics. – 2004. – № 281. – P. 115–128.

8. Panioukova, T.A. Algorithms for Construction of Ordered Enclosing Traces in Planar Eulerian Graphs / T.A. Panioukova, A.V. Panyukov // The International Workshop on Computer Science and Information Technologies' 2003, Proceedings of Workshop, Ufa, September 16 – 18, 2003/ Ufa State Technical University. – Ufa, 2003. – V. 1. – P. 134–138.
9. Panyukova, T. Chain sequences with ordered enclosing / T. Panyukova // Journal of Computer and System Sciences International. – 2007. – V. 6, № 1. – P. 83–92.
10. Panyukov, A.V. The Algorithm for Tracing of Flat Euler Cycles with Ordered Enclosing / A.V. Panyukov, T.A. Panioukova // Известия Челябинского научного центра. – 2000. – № 4 (9). – P. 18–22. Available at: http://elibrary.ru/download/18130929.pdf (accessed November 20, 2000)
11. Панюкова, Т.А. Цепи с упорядоченным охватыванием в плоских графах / Т.А. Панюкова // Дискретный анализ и исследование операций. Часть 2. – 2006. – Т. 13, № 2. – С. 31–43.
12. Панюкова, Т.А. Оптимальные Эйлеровы покрытия для плоских графов / Т.А. Панюкова // Дискретный анализ и исследование операций. – 2011. – Т. 18, № 2. – С. 64–74.

Татьяна Анатольевна Панюкова, кандидат физико-математических наук, доцент, кафедра «Экономико-математические методы и статистика», Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), kwark@mail.ru.

Поступила в редакцию 15 августа 2014 г.