Open Journal of Discrete Mathematics, **2013, 3, 86-92**
http://dx.doi.org/10.4236/ojdm.2013.32017 Published Online April 2013 (http://www.scirp.org/journal/ojdm)

The Software for Constructing Trails with Local Restrictions in Graphs

Tatyana Panyukova, Igor Alferov
Department of Mathematical Methods in Economics and Statistics, South Ural State University, Chelyabinsk, Russian Federation
Email: kwark@mail.ru

Received December 18, 2012; revised March 15, 2013; accepted April 20, 2013

ABSTRACT

The present research considers the problem of covering a graph with minimal number of trails satisfying the pre-defined local restrictions. The research is devoted to the problem of graph covering by minimal number of trails satisfying some local restrictions. Algotithm of allowed Eulerian cycle construction is considered. The authors showed that it is possible to recognize the system of transitions and solve the problem of constructing the allowable path by linear time. It's also possible to find allowable Eulerian cycle for Eulerian graph or to proclaim that such a cycle does not exist by the time $O\left(\left|V\left(G\right)\right|\cdot\left|E\left(G\right)\right|\right)$. All presented algorithms have the software realization.

Keywords: Eulerian Graph; Trail; Transition; Compatible Path; Algorithm

1. Introduction

Lots of problems of finding paths satisfying the different restrictions can be applied to some practical problems. For example for sheet material cutting problem plane graph represents the model of cutting plan, and a path covering all its edges defines the trajectory of cutter. The restriction defined for this problem is lack of intersection of any initial part of path with edges that are not passed yet [1]. Creating the control systems using non-oriented graphs the following problems of constructing the paths with different restrictions can arise. Among them are straight-ahead paths [2]; paths the next edge of which is defined by the given cyclic order on the set of incident edges [3-5]; paths for which it's necessary to pass some edges in pre-defined order [5].

The restrictions on the order of vertices and edges can be classified as local (the next edge of a path is defined by conditions established at the current vertex or edge [2-8]), and global (Eulerian, Hamiltonian cycles, bidirectional double tracing etc.). Most of researches are devoted to algorithms with local restrictions of edges order in a path. The present research considers the problem of covering a graph with minimal number of trails satisfying the pre-defined local restrictions.

2. Constructing of T_G-Compatible Path

The generalization of most of particular cases for problem of simple trail with local restrictions construction and analysis of its computing complexity is made by S.Szeider [7].

Let's quote the basic definitions and results of this research to make the further statements clear. Let's confine with finite simple graphs. Let's designate as $V\left(G\right)$ and $E\left(G\right)$ the sets of vertices and edges of graph G correspondingly. For vertex $v\in V\left(G\right)$ let's define the set $E_G\left(V\right)$ of all graph G edges incident to vertex v. The degree of vertex v let be designated as $d\left(v\right)$; for $d>0$ let $V_d\left(G\right):=\left\{v\in V\left(G\right)\middle|d\left(v\right)=d\right\}$. Let $H\le G$ if H be vertex-induced subgraph of graph G *i.e.* the subgraph received of graph G by deleting of one set of vertices and only all edges incident to vertices of this set.

Restrictions for paths in graph G can be defined in terms of allowed transitions graph.

Definition 1. Let transition graph $T_G\left(v\right)$ for vertex $v\in V\left(G\right)$ be a graph vertices of which are the edges incident to vertex v *i.e.* $V\left(T_G\left(v\right)\right)=E_G\left(v\right)$, and set of edges consists of allowed transitions.

Definition 2. The system of allowed transitions (or shortly, system of transitions) T_G is called the set $\left\{T_G\left(v\right)\middle|v\in V\left(G\right)\right\}$ where $T_G\left(v\right)$ be the transition graph for vertex v.

Definition 3. The path $P=v_0e_1v_1\cdots e_kv_k$ for graph G is called T_G -compatible if $\left(e_i,e_{i+1}\right)\in E\left(T_G\left(v_i\right)\right)$ for each i $\left(1\le i\le k-1\right)$.

Theorem 1 [S. Szeider]. If all graphs of transitions belong either class M of full multipartite graphs or class P of matchings then the problem of T_G-compatible trail constructing can be solved by the time $O\left(\left|E(G)\right|\right)$. Otherwise this problem is *NP*-complete.

If the system of transitions for a vertex $v \in V(G)$ is a matching then this problem can be reduced to the problem for graph

$$G' : V(G') = V(G) \setminus \{v\},$$

$$E(G') = \left(E(G) \setminus E_G(v)\right) \cup \left\{\left\{v_i v_j\right\} : \left\{v_i v, v v_j\right\} \in E\left(T_G(v)\right)\right\}.$$

If for any vertex $v \in V(G)$ graph $T_G(v)$ is full multipartite graph then a trail can be constructed by the following algorithm.

Algorithm T_G-Compatible Path

Input:

- Graph $G = (V, E)$;
- Vertices *x, y* the end-vertices of T_G-compatible trail;
- System of transitions T_G: $\left(\forall v \in V(G)\right)\ T_G(v) \in M$.

Output:

- The sequence of edges corresponding to T_G-compatible trail between vertices *x* and *y* or the message that such a path does not exist.

Step 1. If vertex *x* or vertex *y* is isolated then stop: path does not exist.

Step 2. Delete all isolated vertices from graph *G*.

Step 3. Construct the supplementary graph G' as following (**Figure 1**):

- Each vertex $v \in V(G)$ should be split into vertices $v_1, v_2, \cdots, v_{p(v)}$ where $p(v)$ be the number of parts of graph $T_G(v)$. The edges of corresponding part of graph $T_G(v)$ and one additional vertex $v'_{p(v)}$ are incident to vertex v_p;
- Add two new vertices $w_1(v)$ and $w_2(v)$, edge $w_1(v)w_2(v)$, and edge $v'_{p(v)}w_j(v)$ for each part of graph $T_G(v)$, $1 \le j \le 2$.

Step 4. Construct the initial matching for graph G'

$$M(G') = \bigcup_{v \in V(G)} \left(\bigcup_{p=1,2,\cdots,p(v)} \left\{v_p v'_p\right\} \cup \left\{w_1(v) w_2(v)\right\} \right).$$

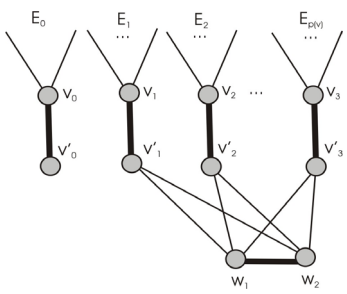


Figure 1. Illustration how supplementary graph G' is constructed.

Step 5. Find the alternate sequence between vertices *x* and *y* that enlarges the cardinality of matching for graph G'. If it's impossible to find such a sequence then stop (matching $M(G')$ has maximal cardinality and graph has no T_G-compatible path). Otherwise all the edges of found enlarging path except of additional edges of graph G' produce the T_G-compatible path between vertices *x* and *y*. **Stop**.

Let's admit that there is open question in research [7]. This question is about recognition the multipartiteness of graphs $T_G(v)$. Problems of constructing the allowed path or set of paths covering all the edges of given graph are not also considered.

Let's illustrate an example of graph *G* (**Figure 2**) that algorithm T_G-COMPATIBLE PATH cannot be used for constructing of paths covering all edges of graph *G*. Let the following system of transitions T_G is defined for the graph:

$$\left\{\{v_2v_1\},\{v_1v_5\}\right\},\ \left\{\{v_6v_1\},\{v_1v_4\}\right\},\ \left\{\{v_4v_3\},\{v_3v_7\}\right\},$$
$$\left\{\{v_8v_3\},\{v_3v_2\}\right\},\ \left\{\{v_3v_2\},\{v_2v_8\}\right\},\ \left\{\{v_5v_2\},\{v_2v_1\}\right\},$$
$$\left\{\{v_1v_4\},\{v_4v_6\}\right\},\ \left\{\{v_7v_4\},\{v_4v_3\}\right\},\ \left\{\{v_2v_5\},\{v_5v_8\}\right\},$$
$$\left\{\{v_2v_8\},\{v_8v_5\}\right\},\ \left\{\{v_3v_8\},\{v_8v_7\}\right\},\ \left\{\{v_3v_7\},\{v_7v_8\}\right\},$$
$$\left\{\{v_4v_7\},\{v_7v_6\}\right\},\ \left\{\{v_4v_6\},\{v_6v_7\}\right\},\ \left\{\{v_1v_6\},\{v_6v_5\}\right\},$$
$$\left\{\{v_1v_5\},\{v_5v_6\}\right\}.$$

Supplementary graph G' for finding of T_G-compatible path between vertices v_1 and v_7 which construction is reviewed on **step 3** of algorithm is shown at **Figure 3**.

The initial matching $M(G')$ is marked by thick lines. The alternate enlarging sequence of edges for this matching be

$$\left\{v_{1,1}v_{5,2}\right\},\ \left\{v_{5,2}v'_{5,2}\right\},\ \left\{v'_{5,2}w_{5,2}\right\},\ \left\{w_{5,2}w_{5,1}\right\},\ \left\{w_{5,1}v'_{5,1}\right\},$$
$$\left\{v'_{5,1}v_{5,1}\right\},\ \left\{v_{5,1}v_{6,2}\right\},\ \left\{v_{6,2}v'_{6,2}\right\},\ \left\{v'_{6,2}w_{6,2}\right\},\ \left\{w_{6,2}w_{6,1}\right\},$$
$$\left\{w_{6,1}v'_{6,1}\right\},\ \left\{v'_{6,1}v_{6,1}\right\},\ \left\{v_{6,1}v_{7,2}\right\}.$$

Edges of this sequence not belonging the initial matching are represented by dash line. These edges form the set

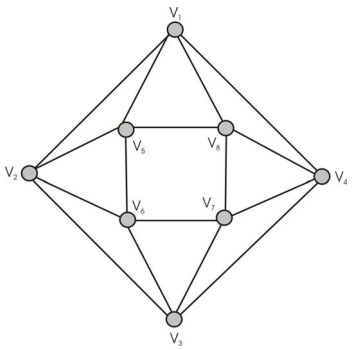


Figure 2. Example of graph.

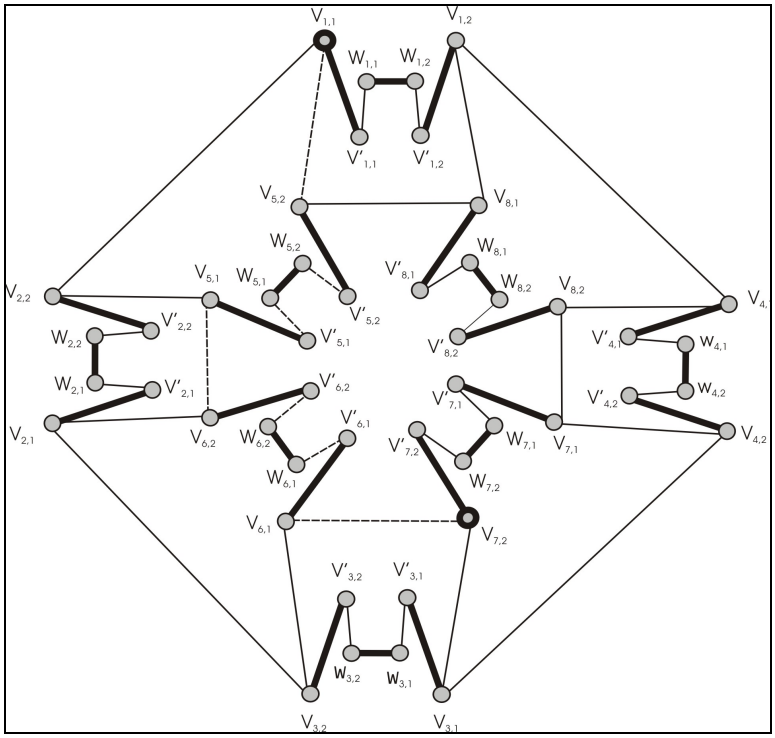


Figure 3. Graph *G'* received of graph *G* by additional constructions.

$\{v_{1,1}v_{5,2}\}$, $\{v'_{5,2}w_{5,2}\}$, $\{w_{5,1}v'_{5,1}\}$, $\{v_{5,1}v_{6,2}\}$, $\{v'_{6,2}w_{6,2}\}$, $\{v_{6,1}w_{6,1}\}$, $\{v_{6,1}v_{7,2}\}$.

All edges of this set belonging to graph *G i.e.* $\{v_1, v_5\}$, $\{v_5, v_6\}$, $\{v_6, v_7\}$ form T_G -compatible path from vertex v_1 to vertex v_7.

Using algorithm T_G -COMPATIBLE PATH it's possible to construct only a simple trail between two different vertices (*i.e.* a trail where each vertex is presented only once).

Figure 4 shows the software realization of the represented algorithm. The bold line marks the found trail between vertices 1 and 4. This trail corresponds the system of allowed transitions (see the additional window at right bottom side).

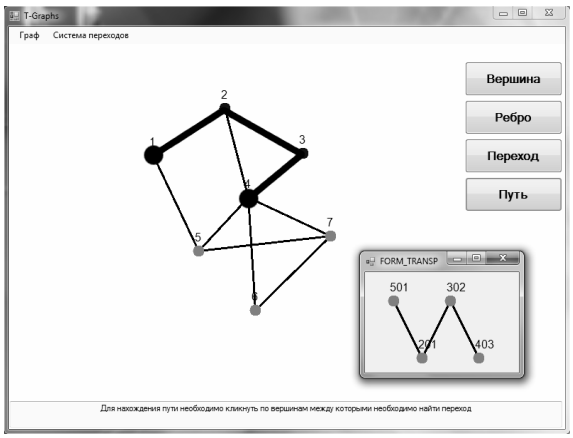


Figure 4. Software for compatible path algorithm.

However in common the direct use of this algorithm does not allow to solve the problem of T_G -compatible path with maximal number of edges constructing. Actually the matching of maximal cardinality for graph G' cannot contain the pairs of edges forming forbidden transition because these edges are incident to one common vertex of graph G'. At the same time, in common there may exist T_G -compatible path containing such a pair of edges.

For example, for graph *G* presented on **Figure 2** the path

$$\{v_2v_1\}, \{v_1v_4\}, \{v_4v_8\}, \{v_8v_1\}, \{v_1v_5\}, \{v_5v_2\}$$

in principle cannot be received by constructing the matching of maximal weight for graph G'. This path begins from edge v_2v_1 and ends by edge v_5v_2. These edges form forbidden transition $\{v_5v_2\}$, $\{v_2v_1\}$, consequently, graph G' does not contain the alternate path with both of these edges.

Thus, the question of multipartite $T_G(v)$ graph recognition is still open as well as the problem of allowed path constructing or finding the set of paths covering all the edges of initial graph *G*.

3. Algorithm for Constructing of Compatible Eulerian Trails

In the previous section the restrictions for paths were

stated in the terms of allowed transitions system [7]. It's shown that the problem of constructing the allowed path in graph *G* can be solved by polynomial time if a system of transitions T_G consists only of matchings and full multipartite graphs. It's trivial to recognize if graph of allowed transitions belongs to the class of matchings. If we want recognize if transitions graph belongs to class of full multipartite graphs it's expedient to use the definition of partition system [3-5,8].

The conception of partition system is used for definition of allowed trail in terms of forbidden transitions.

Definition 4. Let $G=(V,E)$ be a graph. Let $P_G(v)$ be some partition of set $E_G(v)$. Then *the partition system* of graph *G* be the system of sets $P_G := \{P_G(v) | v \in V(G)\}$.

Definition 5. Let $p \in P_G(v)$, $e, f \in p$. A trail not containing the transitions $e \to v \to f$ and $f \to v \to e$ can be called P_G*-compatible*, and transitions $e \to v \to f$ and $f \to v \to e$ are *forbidden*.

Let's admit that graph of allowed transitions $T_G(v)$ unambiguously defines the graph of forbidden transitions $T_G(v)$ which is the complement of allowed transitions graph to full graph. Thus, using definitions 1-3 the problem either with use of allowed transitions or forbidden transitions can be stated.

So the partition system is defined on the set of $E(v)$ (the set of vertices incident to *v*). If edges e_1 and e_2 belong to one subset then edge e_2 cannot be placed after the edge e_1 in a trail. Let graph $G(V,E)$ be defined by the adjacency list. Its elements are the structures. Each element of this structure consists of two fields: vertex number v_i (this vertex is adjacent to the current one); the number of partition element c_i. To define the degree of the current vertex it is enough to count the number of elements of adjacency list.

Let's admit that each edge *e* belongs to two adjacency lists of vertices v_i and v_j (the ends of an edge). But for each vertex edge *e* belongs to different partition systems.

Input data are represented by the following list

vector < list < pair <string, int> > > Graph;

All data are represented by a vector of vertices, each element of this vector is a list of pairs. The first pair of each list is a vertex number, the second one is its degree. The other pairs represent the numbers of adjacent vertices and number of corresponding partition set.

On the other side, graph of allowed transitions defined by partition system P_G cannot be arbitrary, and belongs to class *M* of full multipartite graphs: the elements of partition $P_G(v)$ define the parts of graph $T_G(v) \in M$, and set of its edges

$$E(T_G(v)) = \{e, f \in E_G(v) : (\forall p \in P_G(v))(e,f) \not\subset p\}.$$

Graph of forbidden transitions $T_G(v)$ in this case will consist of $|P_G(v)|$ cliques, this fact can be used for recognition if $T(v) \in M$ using algorithm [9].

As it was considered earlier, algorithm of S. Szeider in common does not allow constructing of allowed trails having maximal length. The most interesting are allowed Eulerian trails. The necessary and sufficient condition for P_G-compatible trails existence is proved by the following theorem [8].

Theorem 2 [A. Kotzig]. Connected Eulerian graph *G* has P_G-compatible Eulerian trail if and only if

$$(\forall v \in V)(\forall p \in P_G(v))\left(|p| \le \frac{1}{2} d_G(v)\right).$$

Obviously, complexity of checking the condition of existence of P_G-compatible Eulerian trail is not more than $O(|E(G)|)$.

Let's list the algorithm for construction of compatible trail.

Algorithm P_G-Compatible Eulerian Trail

Input data:

- Eulerian graph $G=(V,E)$,
- Transitions system $P_G(v)$ $\forall v \in V(G)$.

Output data:

- Allowed Eulerian cycle G_{k+1}.

Step 1. Let $k=0$, $G_k = G$.

Step 2. Find a vertex *v* for which $d_{G_k}(v) > 2$.

Step 3. Find element of partition system containing maximal number of edges. It's enough to look through the adjacency list of current vertex *v* and count how many times each element of partition meets at this list. Choosing this element we get a class

$$C_1 \in P_{G_k}(v) : |C_1| = \{\max |C| \,|\, C \in P_{G_k}(v)\}.$$

Step 4. Find any edges $e_1(v) \in C_1$ and $e_2(v) \in E_{G_k}(v) - C_1$. If it's possible choose edges e_1 and e_2 incident to vertices of degree more than 2. If set $E_{G_k}(v) - C_1 = \varnothing$ then stop: there is no P_G-compatible Eulerian trail. Otherwise go to step 5.

Step 5. Construct graph G_{k+1} by detaching vertex $\hat{v}$, to which only edges e_1 and e_2 are incident, from vertex *v*. The other edges are kept incident to vertex *v*.

Step 6. Let class $C_2 \in P_{G_k}(v)$ contains the edge $e_2(v)$. Exclude vertices v_1 and v_2 from partition system. Define $P^-_{G_k}(v) := P_{G_k}(v) - \{C_1, C_2\}$.

For further modification of partition system define the following.

Step 6.1. All partition systems not containing vertex *v* are taken to the modified system without any changes.

Step 6.2. If systems C_1 and C_2 had been consisted of one edge $|C_1| = |C_2| = 1$ then $P'_{G_{k+1}}(v) := P^-_{G_k}(v)$.

Step 6.3. If $|C_1| > |C_2| = 1$ then

$$P'_{G_{k+1}}(v) := P^-_{G_k}(v) \cup \{C_1 - \{e_1(v)\}\}.$$

Step 6.4. If $|C_2| > 1$ then

$$P'_{G_{k+1}}(v) := P^-_{G_k}(v) \cup \left\{C_1 - \left\{e_1(v)\right\}, C_2 - \left\{e_2(v)\right\}\right\}.$$

Step 6.5. Construct

$$P_{G_{k+1}} = \bigcup_{x \in V(G_{1,2})} P'_{G_{k+1}}(x).$$

Step 7. Define the value

$$\sigma(G_{k+1}) = 2\left(\left|E(G_{k+1})\right| - \left|V(G_{k+1})\right|\right).$$

Let's admit that the number of edges is a constant value and the number of vertices is increased by 1.

Step 8. If $\sigma(G_{k+1}) > 0$, let $k = k+1$ and go to step 2 for graph G_{k+1}. Otherwise go to step 9.

Step 9. Choose any vertex v and mark all achievable vertices. If there are unmarked vertices go to step 10 otherwise stop the received graph G_{k+1} is Eulerian trail without forbidden transitions.

Step 10. Get vertices v_1 and v_2 from a list of marked and unmarked vertices of graph G_{k+1}. These vertices are split from vertex v of graph G_0. Unite them to one vertex $v_{1,2}$. There we get a modified graph $\hat{G}_{k+1}$. Let $k = k+1$.

Step 11. Choose edges $e_1(v_{1,2}) \in C_1$ and $e_2(v_{1,2}) \in E_{G_k}(v_{1,2}) - C_1$ so that $\{e_1, e_2\} \neq E(v_1)$ and $\{e_1, e_2\} \neq E(v_2)$. If set $E_{G_k}(v_{1,2}) - C_1 = \varnothing$ then stop: there is no P_G-compatible Eulerian trail. Otherwise construct graph G_{k+1} by splitting vertex $\hat{v}_{1,2}$ from vertex $v_{1,2}$ Only edges e_1 and e_2 are incident to vertex $\hat{v}_{1,2}$. All other edges are incident to vertex $v_{1,2}$ and go to step 9.

The following theorem is proved in [9].

Theorem 3. Algorithm P_G-compatible Eulerian trail correctly solves the problem of constructing the P_G-compatible Eulerian trail.

That paper also shows that computing complexity of this algorithm is

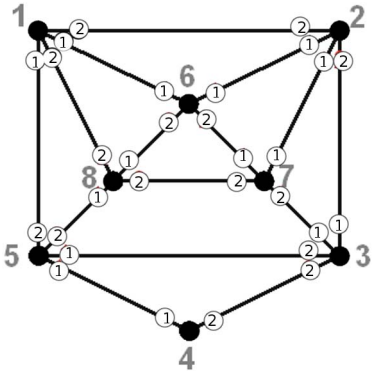
$$O\left(\sum_{k=0,1,\cdots,\sigma(G)} d_{G_k}(v_k)\right) = O\left(\left|E(G)\right| \cdot \left|V(G)\right|\right).$$

Thus, the constructed algorithms are resolved by the polynomial time and can be simply realized using standard computing facilities.

Let's consider the allowed trail construction for a graph on **Figure 5**. Let its transitions system is as shown in a table below. The numbers in white circles show the number of partition system the edge belongs for each vertex.

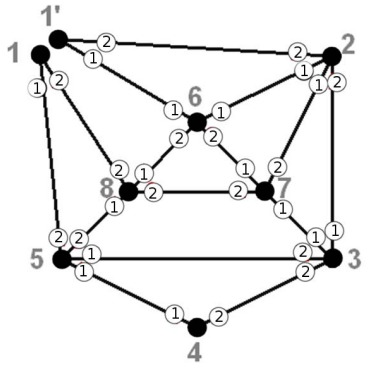
Let's construct allowed Eulerian cycle beginning and ending at vertex 1. At the first iteration the first vertex is split. To simplify the illustration let's consider the "short version" of vertex splitting (the example of "full version" is presented on **Figure 3**). **Figure 6** and table show graph with split vertex 1 and a list of connectivity where new or modified elements are colored by gray.

Figures 7-12 and tables under them represent all other iterations of algorithm.



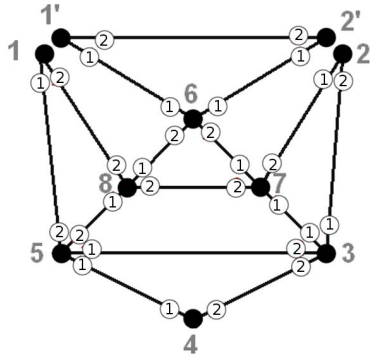
1	(2,2)	(6,1)	(8,2)	(5,1)
2	(1,2)	(6,1)	(7,1)	(3,2)
3	(2,1)	(7,1)	(5,2)	(4,2)
4	(3,2)	(5,1)		
5	(4,1)	(3,1)	(8,2)	(1,2)
6	(1,1)	(2,1)	(7,2)	(8,2)
7	(2,2)	(3,1)	(6,1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5,2)

Figure 5. Example.



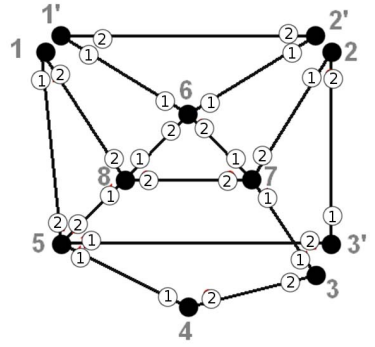
1	(8,2)	(5,1)		
2	(1',2)	(6,1)	(7,1)	(3,2)
3	(2,1)	(7,1)	(5,2)	(4,2)
4	(3,2)	(5,1)		
5	(4,1)	(3,1)	(8,2)	(1,2)
6	(1',1)	(2,1)	(7,2)	(8,2)
7	(2,2)	(3,1)	(6,1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5,2)
1'	(2,2)	(6,1)		

Figure 6. The first iteration of algorithm.



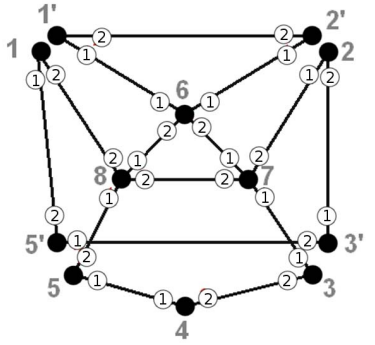
1	(8,2)	(5,1)		
2	(7,1)	(3,2)		
3	(2,1)	(7,1)	(5,2)	(4,2)
4	(3,2)	(5,1)		
5	(4,1)	(3,1)	(8,2)	(1,2)
6	(1',1)	(2',1)	(7,2)	(8,2)
7	(2,2)	(3,1)	(6,1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5,2)
1'	(2',2)	(6,1)		
2'	(1',2)	(6,1)		

Figure 7. The second iteration of algorithm.



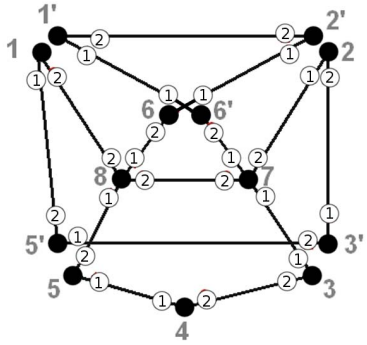
1	(8,2)	(5,1)		
2	(7,1)	(3',2)		
3	(7,1)	(4,2)		
4	(3,2)	(5,1)		
5	(4,1)	(3',1)	(8,2)	(1,2)
6	(1',1)	(2',1)	(7,2)	(8,2)
7	(2,2)	(3,1)	(6,1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5,2)
1'	(2',2)	(6,1)		
2'	(1',2)	(6,1)		
3'	(2,1)	(5,2)		

Figure 8. The third iteration of algorithm.



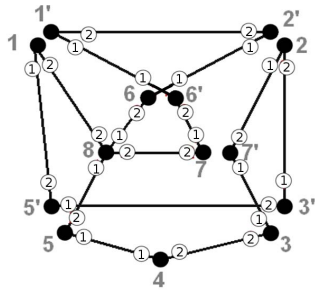
1	(8,2)	(5,1)		
2	(7,1)	(3',2)		
3	(7,1)	(4,2)		
4	(3,2)	(5',1)		
5	(3',1)	(1,2)		
6	(1',1)	(2',1)	(7,2)	(8,2)
7	(2,2)	(3,1)	(6,1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5',2)
1'	(2',2)	(6,1)		
2'	(1',2)	(6,1)		
3'	(2,1)	(5,2)		
5'	(4,1)	(8,2)		

Figure 9. The forth iteration of algorithm.



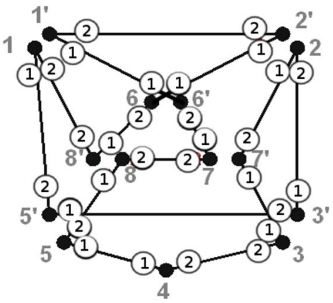
1	(8,2)	(5,1)		
2	(7,1)	(3',2)		
3	(7,1)	(4,2)		
4	(3,2)	(5',1)		
5	(3',1)	(1,2)		
6	(2',1)	(8,2)		
7	(2,2)	(3,1)	(6',1)	(8,2)
8	(1,2)	(6,1)	(7,2)	(5',2)
1'	(2',2)	(6',1)		
2'	(1',2)	(6,1)		
3'	(2,1)	(5,2)		
5'	(4,1)	(8,2)		
6'	(1',1)	(7,2)		

Figure 10. The fifth iteration of algorithm.



1	(8,2)	(5,1)		
2	(7',1)	(3',2)		
3	(7',1)	(4,2)		
4	(3,2)	(5',1)		
5	(3',1)	(1,2)		
6	(2',1)	(8,2)		
7	(6',1)	(8,2)		
8	(1,2)	(6,1)	(7,2)	(5',2)
1'	(2',2)	(6',1)		
2'	(1',2)	(6,1)		
3'	(2,1)	(5,2)		
5'	(4,1)	(8,2)		
6'	(1',1)	(7,2)		
7'	(2,2)	(3,1)		

Figure 11. The sixth iteration of algorithm.



1	(8',2)	(5,1)
2	(7',1)	(3',2)
3	(7',1)	(4,2)
4	(3,2)	(5',1)
5	(3',1)	(1,2)
6	(2',1)	(8',2)
7	(6',1)	(8,2)
8	(7,2)	(5',2)
1'	(2',2)	(6',1)
2'	(1',2)	(6,1)
3'	(2,1)	(5,2)
5'	(4,1)	(8,2)
6'	(1',1)	(7,2)
7'	(2,2)	(3,1)
8'	(1,2)	(6,1)

Figure 12. The last iteration of algorithm.

In result graph is split into a simple cycle. It's possible to construct allowed Eulerian cycle beginning at any its vertex. For example, the following cycle beginning at vertex 1 can be constructed:

$$1 \to 2 \to 6 \to 8 \to 1 \to 5 \to 3 \to 2 \to 7$$
$$\to 3 \to 4 \to 5 \to 8 \to 7 \to 6 \to 1.$$

It's possible to recognize the system of transitions and to solve the problem of constructing the allowable path by linear time. It's also possible to find P_G-allowable Eulerian cycle for Eulerian graph G or to proclaim that such a cycle does not exist. This problem can be solved by the time $O\left(\left|V(G)\right| \cdot \left|E(G)\right|\right)$ using algorithm P_G-compatible Eulerian trail.

4. Acknowledgements

The research is supported by the Ministry of Education and Science of Russian Federation, contract 14.B37.21. 0395.

REFERENCES

[1] T. Panyukova, "Cover with Ordered Enclosing for Flat Graphs," *Electronic Notes in Discrete Mathematics*, Vol. 28, 2007, pp. 17-24. doi:10.1016/j.endm.2007.01.004

[2] T. Pisanski, T. W. Tucker and A. Zitnik, "Straight-Ahead walks in Eulerian Graphs," *Discrete Mathematics*, Vol. 281, No. 1-3, 2004, pp. 237-246. doi:10.1016/j.disc.2003.09.011

[3] H. Fleischner, "Eulerian Graphs and Related Topics," Part 1, Vol. 1, Elsevier, Amsterdam, 1990.

[4] H. Fleischner, "Eulerian Graphs and Related Topics," Part 1, Vol. 2, Elsevier, Amsterdam, 1991.

[5] H. Fleischner, L. W. Beineke and R. J. Wilson, "Eulerian Graphs, Selected Topics in Graph Theory 2," Academic Press, London, New York, 1983, pp. 17-53.

[6] D. Chebikin, "On k-Edge-Ordered Graphs," *Discrete Mathematics*, Vol. 281, No. 1-3, 2004, pp. 115-128. doi:10.1016/j.disc.2003.09.004

[7] S. Szeider, "Finding Paths in Graphs Avoiding Forbidden Transitions," *Discrete Applied Mathematics*, Vol. 126, No. 2-3, 2003, pp. 261-273. doi:10.1016/S0166-218X(02)00251-2

[8] A. Kotzig, "Moves without Forbidden Transitions in a Graph," *Matematický Časopis*, Vol. 18, No. 1, 1968, pp. 76-80.

[9] T. A. Panyukova, "The Paths with Local Restrictions," *Reports of South Ural State University. Section*: *Mathematical Modelling and Programming*, Vol. 5, No. 16, 2010, pp. 58-67 (in Russian).

