УДК 512.5, 519.1(075.8) **DOI: 10.14529/cmse190103**

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОСТРОЕНИЯ A-ЦЕПЕЙ С УПОРЯДОЧЕННЫМ ОХВАТЫВАНИЕМ В ПЛОСКОМ СВЯЗНОМ 4-РЕГУЛЯРНОМ ГРАФЕ

© 2019 Т.А. Макаровских

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: Makarovskikh.T.A@susu.ru

Поступила в редакцию: 24.07.2018

В CAD/CAM-системах технологической подготовки процессов раскроя встает задача построения маршрута движения режущего инструмента, при котором отрезанная от листа часть не требует дополнительных разрезаний и запрещены пересечения траектории резки (касания допускаются). Формально такая задача может быть сформулирована как задача построения самонепересекающейся цепи в плоском эйлеровом графе, представляющим гомеоморфный образ раскройного плана. В конечном счете задачи построения маршрутов, удовлетворяющих технологическим ограничениям, сводятся к нахождению A-цепи с упорядоченным охватыванием в плоском связном 4-регулярном графе. В статье предложен алгоритм нахождения такой цепи. Выполнение алгоритма состоит из двух этапов. На первом этапе выявляются и расщепляются точки сочленения ранга k. На втором этапе построение цепи начинается из произвольной вершины, инцидентной внешней грани; первым ребром цепи выбирается инцидентное данной вершине ребро максимального ранга; далее организуется итерационный процесс, где в качестве следующего ребра выбирается непройденное ребро максимального ранга, являющееся левым либо правым соседом текущего ребра. Показано, что для плоского связного 4-регулярного графа алгоритм строит маршрут с указанными свойствами за линейное время. Представленные алгоритмы реализованы в виде компьютерной программы. Приведены примеры решения ряда тестовых задач.

Ключевые слова: плоский граф, маршрут, раскройный план, полиномиальный алгоритм, CAD/CAM.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Макаровских Т.А. Программное обеспечение для построения A-цепей с упорядоченным охватыванием в плоском связном 4-регулярном графе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 1. С. 36–53. DOI: 10.14529/cmse190103.

Введение

Лазерная резка является одной из основных современных технологий, используемых при обработке листового материала, что делает актуальной задачу определения траектории движения режущего инструмента. Задача определения траектории заключается в определении точной последовательности резов. Развитие автоматизации производства привело к появлению технологического оборудования с числовым программным управлением, используемого для резки листовых материалов. Новые технологии позволяют осуществлять вырезание по произвольной траектории с достаточной для практики точностью. Преимуществом при использовании лазерной резки является минимальность таких показателей как ширина реза и термические деформации. Целью задачи определения маршрута резки является поиск такого пути режущего инструмента, при котором выполняются условия предшествования, а время, затраченное на вырезание, минимально [1].

В терминах задачи лазерной резки под условием предшествования понимается требование к тому, чтобы отрезанная от листа часть не требовала дополнительных разрезаний (т.е. все элементы вложенного контура должны быть вырезаны прежде, чем внешний контур окажется полностью вырезанным).

Общее время, требуемое на выполнение резки, можно представить как суммарное время для осуществления всех вырезаний, времени на выполнение холостых переходов, времени на врезку и пр. Так как основным условием предшествования при выполнении лазерной резки является требование к отсутствию разрезаний для уже отрезанной части листа, то

- во-первых, все элементы внутренних контуров должны быть вырезаны прежде, чем будут полностью отрезаны от листа внешние контуры;
- во-вторых, необходимо учитывать термальные эффекты, например, стоит избегать пересечения имеющихся резов.

При технологической подготовке процессов раскроя с применением лазерной резки (для CAD/CAM систем) возникает задача нахождения маршрута для режущего инструмента, при котором отрезанная от листа часть не требует разрезаний, и не допускаются пересечения в траектории резки (касания допускаются) [1, 2]. В процессе лазерной резки происходит нагревание металлического листа. В связи с этим при определении траектории движения режущего инструмента необходимо учитывать как термические эффекты, так и возможные деформации. Это приводит к необходимости избегать пересечений резов.

В работах [1, 3] приводится классификация задач маршрутизации режущего инструмента и отмечается, что технологии ECP (Endpoint Cutting Problem) и ICP (Intermittent Cutting Problem) за счет возможности совмещения границ вырезаемых деталей позволяют сократить расход материала, длину резки и длину холостых проходов [3]. Проблемы уменьшения отходов материала и максимального совмещения фрагментов контуров вырезаемых деталей решаются на этапе составления раскройного плана.

Несмотря на очевидные преимущества технологий ECP и ICP, в настоящее время большинство отечественных [6–10] и зарубежных [1, 3–5] работ посвящено развитию технологии GTSP (General Travelling Salesman Problem), которая не предполагает совмещение контуров вырезаемых деталей. Таким образом, при использовани данной технологии длина траектории будет равна сумме периметров всех контуров, а количество точек врезки — количеству контуров. Однако при этом проблема выполнения отмеченных выше условий предшествования оказывается тривиальной.

Построению эффективных алгоритмов для раскройных планов, в которых допускаются совмещения границ контуров, посвящен ряд работ [11–13]. Алгоритм из работы [11] находит траекторию движения режущего инструмента и минимизирует число точек врезки. Предложенный в работе алгоритм применим только для раскройных планов, допускающих плоскую достройку до эйлерова графа. В статье [12] рассмотрена та же задача, что и в [11], задача формализована достаточно подробно, введено понятие маршрута с ограничениями в плоском графе (OE-маршрутов) и приведен не только алгоритм решения задачи для эйлерова графа, но и решена задача китайского почтальона (таким образом, приведен алгоритм решения задачи для плоского связного графа). Один из подходов к построению покрытия упорядоченной последовательностью реберно-непересекающейся последовательностью цепей рассмотрен и в работе [13]. Он требует решения задачи сельских почтальонов. Напомним, что в маршруте почтальона отсутствует требование, чтобы маршрут являлся реберно-непересекающимся.

Все отмеченные выше работы [11–13] не исключают построение траектории с самопересечениями.

О построении маршрута, не допускающего пересечения траекторий, впервые упоминается в работе [14]. В данной работе речь идет о самонепересекающейся цепи, тем не менее, в данной статье понятие самонепересекающейся цепи ошибочно сужено до понятия A-цепи [15]. В действительности, класс самонепересекающихся цепей шире класса A-цепей [17]. Под самонепересекающейся цепью понимают эйлеров цикл C в плоском графе G, гомеоморфный плоскому циклическому графу $\tilde{G}$ (представляющему жорданову кривую на плоскости), который может быть получен из графа G с помощью применения $O(|E(G)|)$ операций расщепления вершин. В работах [17, 18] показано, что задача построения самонепересекающейся цепи в графе G сводится к задаче нахождения A-цепи с упорядоченным охватыванием для плоского связного 4-регулярного графа $\tilde{G}$, полученного в результате введения в гомеоморфный образ раскройного плана фиктивных вершин и ребер. После стягивания всех фиктивных вершин и ребер в графе $\tilde{G}$ будет получена самонепересекающаяся цепь в графе G.

Таким образом, разработка эффективных алгоритмов для построения A-цепи с упорядоченным охватыванием для плоского связного 4-регулярного графа является актуальной задачей.

Статья организована следующим образом. В разделе 1 приводится представление гомеоморфного образа раскройного плана. В разделе 2 введены необходимые определения и предварительные результаты, используемые в дальнейших рассуждениях. Приведен полиномиальный алгоритм построения AOE-цепи в плоском связном 4-регулярном графе. Доказана его результативность. В разделе 3 приведены примеры тестовых задач, решенных с помощью разработанного алгоритма. В заключении перечислены полученные в работе результаты, отмечены направления дальнейших исследований.

1. Абстрактное представление раскройного плана в виде плоского графа

Для определения последовательности резки фрагментов раскройного плана не используется информация о форме детали, поэтому все кривые без самопересечений и соприкосновений на плоскости, представляющие форму деталей, интерпретируются в виде ребер плоского графа [19], а все точки пересечений и соприкосновений представляются в виде вершин графа. Для изложенных в работе алгоритмов необходимо представление графа в виде системы инцидентности. Чтобы система инцидентности с точностью до гомеоморфизма была эквивалентна раскройному плану, необходимо и достаточно в каждой вершине указать циклический порядок[1] на множестве инцидентных этой вершине ребер.

Рассмотрим подробнее представление плоского графа в виде его гомеоморфного образа, т.е. в виде системы инцидентности. Гомеоморфным образом раскройного плана является плоский граф G с внешней гранью f_0 на плоскости S. Для любой части J графа G (т.е. $J \subseteq G$) обозначим через $\mathsf{Int}\,(J)$ теоретико-множественное объединение его внутренних граней (объединение всех связных компонент $S \setminus J$, не содержащих внешней грани). Тогда $\mathrm{Int}(J)$ можно интерпретировать как отрезанную от листа часть. Множества вершин, ребер и граней графа J будем обозначать через $V(J)$, $E(J)$ и $F(J)$ соответственно.

[1]Циклическим порядком называется способ расположения объектов на окружности.

Топологическое представление плоского графа G на плоскости S с точностью до гомеоморфизма определяется заданием для каждого ребра $e \in E(G)$ следующих функций [2, 20]:

- $v_k(e)$, $k = 1, 2$ — вершины, инцидентные ребру e;
- $l_k(e)$, $k = 1, 2$ — ребра, полученные вращением ребра e против часовой стрелки вокруг вершины $v_k(e)$;
- $r_k(e)$, $k = 1, 2$ — ребра, полученные вращением ребра e по часовой стрелке вокруг вершины $v_k(e)$;
- $f_k(e)$ — грань, находящаяся справа при движении по ребру e от вершины $v_k(e)$ к вершине $v_{3-k}(e)$, $k = 1, 2$.

Иллюстрация введенных функций дана на рис. 1.

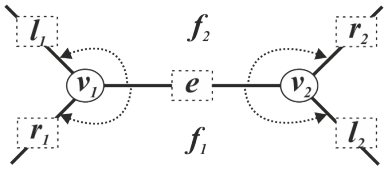


Рис. 1. Функции для представления плоского графа

Таким образом, используя известные координаты прообразов вершин графа G и размещения фрагментов раскройного плана, являющихся прообразами ребер графа G, любой маршрут в графе G можно интерпретировать как траекторию режущего инструмента.

Представление графа фактически задает ориентацию его ребер. Далее предполагается, что движение по ребру для определенности осуществляется от вершины $v_1(e)$ к вершине $v_2(e)$. Поскольку при задании графа G неизвестно, какое из ребер в каком направлении будет пройдено, то при выполнении алгоритма производится перестановка значений полей $v_k(e)$, $r_k(e)$ и $l_k(e)$, $f_k(e)$, $k = 1, 2$ некоторых ребер. В алгоритме данную процедуру выполняет функция `REPLACE`, функциональным назначением которой является перестановка индексов функций $v_k(e)$, $l_k(e)$ и $r_k(e)$ на $3 - k$, $k = 1, 2$ [20].

Далее будем считать, что все рассматриваемые плоские графы представлены указанными функциями. Пространственная сложность такого представления будет $O(|E(G)| \cdot \log_2 |V(G)|)$.

2. Построение AOE-цепи в плоском графе

2.1. Определения и обозначения

В дальнейшем будем использовать ряд понятий, определения которых имеются в работах [15, 21, 22]. Приведем их для полноты изложения.

Определение 1. *Графом разрешенных переходов* $T_G(v)$ вершины $v \in V(G)$ [21] будем называть граф, вершинами которого являются ребра, инцидентные вершине v, т.е. $V(T_G(v)) = E_G(v)$, а множество ребер — разрешенные переходы между ребрами.

Определение 2. *Системой разрешенных переходов* T_G [21] будем называть множество $\{T_G(v) \mid v \in V(G)\}$, где $T_G(v)$ — граф переходов в вершине v.

Определение 3. Путь $P = v_0, e_1, v_1, ..., e_k, v_k$ в графе G является T_G-*совместимым* [21], если $e_i,\ e_{i+1} \in E(T_G(v_i))$ для каждого i $(1 \le i \le k-1)$.

Определение 4. Пусть для цепи $T = v_0, k_1, v_1, \ldots, k_n, v_n,\ v_n = v_0$ в графе $G = (V, E)$ в каждой вершине $v \in V$ задан циклический порядок $O^{\pm}(v)$, определяющий систему переходов $A_G(v) \subset O^{\pm}(v)$. В случае, когда $\forall v \in V(G)\ A_G(v) = O^{\pm}(v)$, систему переходов $A_G(v)$ будем называть *полной системой переходов.*

В частности, маршруты, в которых цикл из пройденных ребер не охватывает еще непройденных (т.е. заданы условия предшествования: отрезанная от листа часть не требует дополнительных разрезаний) представляют класс OE-маршрутов.

Определение 5. Будем говорить, что цикл $C = v_1e_1v_2e_2\ldots v_k$ в плоском эйлеровом графе G имеет *упорядоченное охватывание* (называется OE-цепью), если для любой его начальной части $C_i = v_1e_1v_2e_2\ldots e_i$, $i \le |E(G)|$ выполнено условие $\mathrm{Int}\,(C_i) \cap G = \emptyset$.

Определение 6. Упорядоченная последовательность реберно-непересекающихся OE-цепей

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

покрывающая граф G и такая, что

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

называется *маршрутом с упорядоченным охватыванием* (OE-маршрутом).

Определение 7. Эйлерову цепь T будем называть A-*цепью* [15], если она является A_G-совместимой цепью. Таким образом, последовательные ребра в цепи T (инцидентные вершине v) являются соседями в циклическом порядке $O^{\pm}(v)$.

Определение 8. Будем говорить, что цепь является AOE-*цепью*, если она одновременно является OE-цепью и A-цепью.

Определение 9. *Рангом ребра* $e \in E(G)$ будем называть значение функции $\mathrm{rank}(e) : E(G) \to \mathbb{N}$, определяемое рекурсивно:

- пусть $E_1 = \{e \in E :\ e \subset f_0\}$ — множество ребер, ограничивающих внешнюю грань f_0 графа $G(V, E)$, тогда $(\forall e \in E_1)\,(\mathrm{rank}(e) = 1)$;
- пусть $E_k(G)$ — множество ребер ранга 1 графа

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

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

Ранг ребра определяет его удалённость от внешней грани и показывает, какое минимальное число граней необходимо пересечь, чтобы добраться от внешней грани f_0 до этого ребра. Это позволяет для определения ранга использовать граф G', топологически двойственный исходному графу G: множеством вершин графа G' является множество $F(G)$,

а ребрам графа G' соответствует наличие между двумя гранями границы ненулевой длины, т.е. ребра $e \in E(G)$.

Для всех $f \in F(G)$ расстояние в графе G' между f и внешней гранью f_0 можно определить, построив в графе G' дерево $T^{f_0}_{G'}$ кратчайших путей до вершины $f_0 \in F$. Наличие в представлении графа G функций $l_k : E(G) \to E(G)$, $k = 1, 2$ позволяет найти функции ранга за время, не превосходящее величины $O(|E(G)| \cdot \log_2 |V(G)|)$.

Различные способы вычисления значений функции $\mathrm{rank}(e)$ приведены в работах [22, 23].

Теорема 1. Если в плоском графе G графе существует A-цепь, то существует и AOE-цепь.

Доказательство. A-цепь в плоском эйлеровом графе представляет замкнутую жорданову кривую без самопересечений. Данную кривую можно получить следующим образом: рассмотрим граф G, в каждой вершине которого задана система переходов, соответствующая A-цепи, и граф G', полученный из графа G расщеплением вершин в соответствии с системой разрешенных переходов (рис. 2). Построенный таким образом граф G' представляет собой жорданову кривую без пересечений и в соответствии с теоремой Жордана [16] разбивает плоскость на внешнюю и внутреннюю компоненты связности.

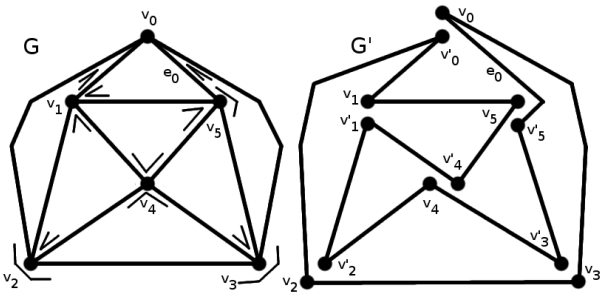


Рис. 2. Плоский граф G с заданной системой переходов A-цепи и соответствующий ему граф G', являющийся жордановой кривой на плоскости

Очевидно, что на полученной кривой найдется вершина v_0, инцидентная внешней грани графа G и ребру e_n: $\mathrm{rank}(e_n) = 1$. Если принять вершину v_0 за начало AOE-цепи, а направление обхода выбрать таким образом, чтобы ребро (e_n) было концом цепи C_n, то в силу отсутствия пересечений для внутренности любой начальной части цепи $C_i = v_0 e_1 v_1 e_2 \dots e_i$, $i \le |E(G)|$ будет выполнено условие

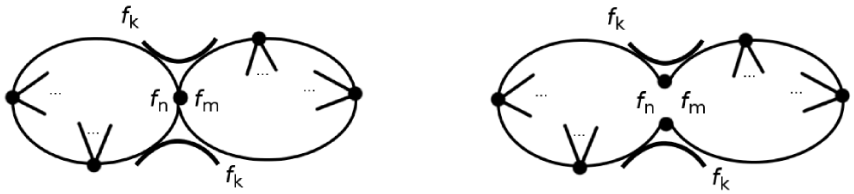
$$\mathrm{Int}\,(C_i) \cap G = \emptyset,$$

т.е. такая A-цепь C_n будет являться и OE-цепью. Действительно, предположение существования ребра $e \in \mathrm{Int}\,(C_i)$ сразу приводит к противоречию с тем, что в системе переходов цепи отсутствуют пересечения. *Теорема доказана.*

Теорема 2. В плоском связном 4-регулярном графе G существует AOE-цепь.

Доказательство. Доказательство существования A-цепи в связном 4-регулярном графе приведено в [15, Следствие VI.6]. Так как в этом графе существует A-цепь, то в силу теоремы 1 в нем существует и AOE-цепь. *Теорема доказана.*

Определение 10. Суграф G_k графа G, для которого $E(G_k) = \{e \in E(G) : \mathrm{rank(e)} \ge \mathrm{k}\}$ назовем *сугрaфом ранга* k.



а) Верная система переходов для расщепления точки сочленения суграфа G_k

б) Расщепление в соответствии с системой переходов в точке сочленения суграфа G_k

Рис. 3. Расщепление точек сочленения ранга k

Далее будет рассмотрен алгоритм `AOE-TRAIL`, позволяющий найти AOE-цепь в плоском связном 4-регулярном графе, в котором любой суграф G ранга k не имеет точек сочленения.

При поиске точек сочленения суграфа G_k используются следующие свойства 4-регулярных графов.

Предложение 1. Вершина, инцидентная четырем ребрам, смежным внешней грани, является точкой сочленения.

Предложение 2. Внешняя грань суграфа G_k является объединением всех граней ранга k в графе G.

Справедливость этих предложений очевидна в силу 4-регулярности графа.

Определение 11. *Рангом грани* $f \in F(G)$ будем называть значение функции rank : $F(G) \to \mathrm{Z}^{\geq 0}$:

$$\mathrm{rank}(f) = \begin{cases} 0, & \text{при } f = f_0, \\ \min_{e \in E(f)} \mathrm{rank}(e), & \text{в противном случае,} \end{cases}$$

где $E(f)$ — множество ребер инцидентных грани $f \in F$.

Определение 12. *Точкой сочленения ранга* k назовем вершину, инцидентную четырем ребрам, смежным внешней грани суграфа G_k.

2.2. Расщепление точек сочленения ранга k

Для корректного построения AOE-цепи в плоском связном 4-регулярном графе необходимо предварительно «правильно» расщепить все точки сочленения суграфов G_k. В результате такого расщепления получим граф, у которого любой суграф G_k не содержит точек сочленения. Последовательность расщепления вершин не имеет значения, т.к. это локальная операция.

Под «правильным» будем понимать переход между дугами, соответствующими циклическому порядку и инцидентными различным парам граней (см. рис. 3.а)).

Результат расщепления приведен на рис. 3.б).

Рассмотрим алг. 1, используемый для выполнения операции расщепления точек сочленения всех рангов в плоском связном 4-регулярном графе.

Алгоритм 1 `CUT-POINT-SPLITTING`

Require: плоский связный 4-регулярный граф $G = (V, E)$, представленный для всех $e \in E(G)$ функциями v_s, l_s, r_s , $s = 1, 2$.

Ensure: гомеоморфный образ графа $G = (V, E)$, представленный для всех $e \in E(G)$ функциями v_s, l_s, r_s , $s = 1, 2$, в котором любой суграф G_k не имеет точек сочленения.

Промежуточные данные: $\forall v \in V(G)$: point(v) — массив указателей на одно из ребер, инцидентных вершине v, rank(v) — массив рангов вершин, count(v) — счетчик инцидентных ребер одного ранга для каждой вершины;
$\forall f \in F(G)$: массив rank(f).

```
1: Initiate():
2: for all v ∈ V(G) do                    ▷ Обнулить счетчик инцидентных вершине ребер одного ранга
3:     point(v) := 0; count(v) := 0
4: end for
5: Ranking(G)                             ▷ Определение ранга всех вершин, ребер и граней графа
6: Finding():                             ▷ Нахождение точек сочленения
7: for all e ∈ E(G) do
8:     point(v_1(e)) := e
9:     point(v_2(e)) := e
10: end for
11: for all v ∈ V(G) do                   ▷ Просмотреть последовательно все вершины
12:     e := point(v)
13:     k := rank(v)                      ▷ Сохранить значение ранга k инцидентного ребра e
14:     if v = v_1(e) then                ▷ Определить ориентацию ребра e
15:         s := 1
16:     else
17:         s := 2
18:     end if
19:     e := l_s(e)                       ▷ Подсчет количества инцидентных вершине v ребер ранга k
20:     for i = 1 up to 4 do
21:         if rank(e) = k then
22:             count(v) := count(v) + 1
23:             if i < 4 then
24:                 e := l_s(e)
25:             end if
26:         end if
27:     end for
28:     if count(v) = 4 then              ▷ Если найдена точка сочленения, расщепить вершину
29:         if (f_s(e) = f_s(l_s(e)) and f_{3−s}(e) = f_{3−s}(l_s(e))) or (f_s(e) = f_{3−s}(l_s(e)) and f_{3−s}(e) =
    f_s(l_s(e))) then
30:             e* := l_s(e), l_s(e) := r_s(e), r_s(r_s(e)) := e,
31:             r_s(e*) := l_s(e*), l_s(l_s(e*)) := e*
32:         else
33:             e* := r_s(e), r_s(e) := l_s(e), l_s(l_s(e) := e,
34:             l_s(e*) := r_s(e*), r_s(r_s(e*)) := e*
35:         end if
36:     end if
37: end for
```

Справедлива следующая теорема.

Теорема 3. Алгоритм `CUT-POINT-SPLITTING` определяет точки сочленения всех рангов плоского связного 4-регулярного графа и расщепляет их без нарушения связности за время не превосходящее $O(|E|)$.

Доказательство. На этапе инициализации (функция `Initiate()`, строки 1–4) для каждой вершины $v \in V(G)$ обнуляется счетчик $count(v)$ инцидентных ей ребер, ранг которых совпадает с рангом v. Затем вычисляются ранги всех вершин, ребер и граней графа (функция `Ranking()`, строка 5).

Далее осуществляется поиск и расщепление точек сочленения каждого ранга (строки 6–37). В первом цикле (строки 7–10) осуществляется инициализация массива $point(v)$: для каждой вершины определяется инцидентное ей ребро. В следующем цикле (строки 11–37) определяется ранг k текущей вершины (строка 13), уточняется ориентация текущего ребра (строки 14–18). В цикле (строки 19–27) подсчитывается количество инцидентных вершине v ребер, ранг которых совпадает с рангом k вершины. Если текущая вершина инцидентна четырем ребрам ранга k, то в соответствии с предложением 1, то она является точкой сочленения ранга k. В этом случае выполняется расщепление точки сочленения ранга k за счет соответствующей модификации указателей на смежные ребра (см. строки 28–37 алг. 1) в соответствии с рис. 3.

Итак, алг. 1 за время не превосходящее $O(|E|)$ осуществляет последовательный просмотр всех вершин графа G и в случае обнаружения точки сочленения ранга k выполняет расщепление найденной вершины на две вершины степени 2 без нарушения связности графа. Повторный просмотр вершин не требуется: алгоритм не изменяет ранги ребер, а только модифицирует их смежность. В результате каждой операции расщепления будет получено две вершины степени 2. Следовательно, в результате единственного последовательного просмотра всех вершин графа для каждого ранга k будут расщеплены все имеющиеся точки сочленения. Появление новых точек сочленения ранга k невозможно, т.к. значение функции rank(e) не изменяется. *Теорема доказана.*

2.3. Алгоритм построения AOE-цепи в плоском связном 4-регулярном графе без точек сочленения ранга k

Рассмотрим алг. 2, выполняющий построение AOE-цепи в плоском связном 4-регулярном графе без точек сочленения.

Алгоритм 2 AOE-`TRAIL`

Require: плоский связный 4-регулярный граф G без точек сочленения, представленный функциями v_k, l_k, r_k (рис. 1), $k = 1, 2$;

1: начальная вершина $v \in V(f_0)$.

Ensure: $ATrail$ — выходной поток, содержащий построенную алгоритмом AOE-цепь.

2: Initiate(G, v_0)

3: Ranking(G) ▷ Ранжирование

4: Constructing() ▷ Построение

Процедура инициализации (`Initiate()`) заключается в инциализации значением 0 счетчика `counter` количества ребер, включенных в результирующую цепь, а также в присваивании всем ребрам пометки `true`, соответствующей непройденному ребру.

Функциональное назначение процедуры `Ranking()` — определить для каждого ребра $e \in E(G)$ графа его ранг $\mathrm{rank}(e)$. Как отмечалось ранее, различные алгоритмы вычисления значений функции $\mathrm{rank}(e)$ приведены в работах [22, 23], их вычислительная сложность не превосходит величины $O(|E(G)| \cdot \log_2 |V(G)|)$.

Описание процедуры `Constructing()`, выполняющей построение AOE-цепи представлено ниже (см. алг. 3).

Алгоритм 3 `Constructing (v)`

1: $e = \arg\max_{e \in E(v)} \mathrm{rank}(e)$ ▷ Выбрать ребро максимального ранга, инцидентное вершине v
2: **repeat**
3: **if** $v \neq v_1(e)$ **then**
4: REPLACE(e)
5: **end if** ▷ При необходимости скорректировать нумерацию функций для ребра e
6: **Print**(v, e) ▷ Вывести ребро e в результирующую последовательность
7: mark(e) := **false**; counter:=counter+1; $v := v_2(e)$ ▷ Пометить ребро как пройденное
8: **if** ($\mathrm{rank}(r_2(e)) \geq \mathrm{rank}(l_2(e))$) **then** ▷ Выбрать следующим ребро максимального ранга
9: **if** mark($r_2(e)$) **then** ▷ Проверить, пройдено ли добавляемое в цепь ребро
10: $e := r_2(e)$ ▷ Для пройденных ребер значение в массиве $mark$ — ложь
11: **else**
12: $e := l_2(e)$
13: **end if**
14: **else**
15: **if** (mark($l_2(e)$)) **then** ▷ Выбрать непройденное ребро
16: $e := l_2(e)$
17: **else**
18: $e := r_2(e)$
19: **end if**
20: **end if**
21: **until** (counter $> |E(G)|$) ▷ Завершить цикл, когда просмотрены все ребра

В строке 1 предписано для заданной инцидентной внешней грани вершины v выбрать инцидентное ей ребро e максимального ранга.

В следующем далее цикле **repeat** — **until** (строки 2–21) осуществляется построение AOE-цепи:

- при необходимости корректируется нумерация функций для ребра e (строки 3–5), т.е. производится взаимообмен номеров инцидентных ребру e вершин, таким образом, чтобы вершина $v_1(e)$ посещалась при обходе раньше вершины $v_2(e)$;
- текущее ребро выводится в результирующую последовательность (строка 6);
- помечается как пройденное (mark(e)=**false**), в качестве текущей вершины v выбирается $v_2(e)$, увеличивается значение счётчика пройденных ребер (строка 7);
- в качестве следующего текущего ребра e выбирается инцидентное вершине v непройденное ребро (mark(e)=**true**), являющееся по возможности левым или правым соседом предыдущего ребра.

Теорема 4. Алгоритм `AOE-TRAIL` и процедура `Constructing(v)` строят AOE-цепь в плоском связном 4-регулярном графе G за время $O(|E(G)| \cdot \log_2 |V(G)|)$.

Доказательство. Результативность процедуры `Initiate()` очевидна. Результативность процедуры `Ranking()` доказана в [24]. Их совокупная вычислительная сложность не превосходит $O(|E(G)| \cdot \log_2 |V(G)|)$.

Основной цикл процедуры `Constructing()` состоит в выборе последующего непройденного ребра максимального ранга, смежного ребру e. Процедура выполняется до тех пор, пока все ребра не будут включены в результирующую цепь (их пометка будет изменена на `false`).

Докажем результативность процедуры `Constructing()`. Если для текущего ребра e вершина $v = v_2(e)$ посещается впервые, то выполнение тела цикла можно интерпретировать как расщепление вершины $v^* = v_2(e)$ в соответствии с системой переходов A-цепи. После расщепления гомеоморфный образ полученного графа уже не будет содержать вершины v^*. При повторном попадании алгоритма в вершину $v^* \in V(G)$ алгоритм продолжает формирование цепи в соответствии с гомеоморфным образом полученного ранее ребра.

Гомеоморфный образ полученного графа после выполнения тела цикла является плоским связным графом, так как ни один суграф ранга k не содержит точек сочленения.

После выполнения $|V(G)|$ расщеплений в соответствии с системой переходов A-цепи получим гомеоморфный образ графа, являющийся окружностью. В этом случае поток $ATrail$ будет содержать полученную AOE-цепь.

При выполнении расщеплений гомеоморфный образ остается связным графом, поэтому все ребра будут обработаны алгоритмом (получат пометку `false`). Процедура `Constructing()` имеет единственный цикл. Вычислительная сложность тела цикла не превосходит величины $O(\log_2 |V(G)|)$, а число итераций этого цикла равно $|E(G)|$. Следовательно, вычислительная сложность алгоритма не превосходит величины $O(|E(G)| \cdot \log_2 |V(G)|)$. *Теорема доказана.*

3. Примеры работы алгоритма

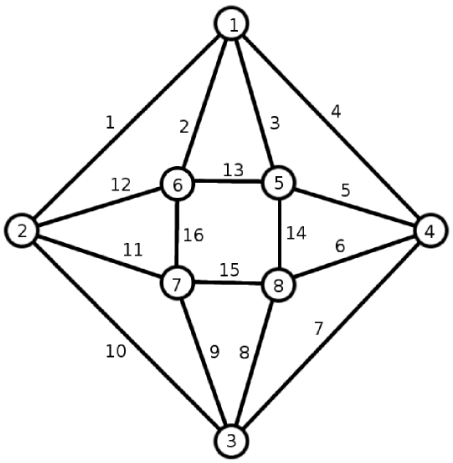
Представленные выше алгоритмы реализованы в виде компьютерной программы. Приведем примеры тестовых задач, решенных с помощью этой программы.

Во входном файле указывается общее число ребер, каждая последующая строка файла соответствует одному ребру. В ней приведены значения указанных в разделе 1 функций в следующем порядке: инцидентные вершины (функции $v_k(e)$, $k = 1, 2$); левые (функции $l_k(e)$, $k = 1, 2$) и правые (функции $r_k(e)$, $k = 1, 2$) соседние ребра; грани $f_k(e)$, $k = 1, 2$, инцидентные ребрам $l_k(e)$; последним числом в строке указывается ранг $\mathrm{rank}(e)$ соответствующего ребра. Плоский связный 4-регулярный граф с примером его кодирования приведены на рис. 4.

Разработанная программа в зависимости от номера начальной вершины (задаваемого пользователем) определяет AOE-цепь в графе, либо выводит сообщение, что задача не имеет решения.

Для тестирования разработанной программы достаточно рассмотреть следующие возможные случаи:

- граф, не имеющий точек сочленения (в этом случае не осуществляется вызов функции `CutPointSplitting`, пример такого графа представлен на рис. 4); найденные программой последовательности ребер, соответствующие AOE-цепям в указанном графе, приведены в табл. 1;




```
16
1 2 4 12 2 10 1 1
1 6 1 13 3 12 0 2
1 5 2 5 4 13 0 2
1 4 3 7 1 5 1 1
4 5 4 14 6 3 0 2
4 8 5 8 7 14 0 2
4 3 6 10 4 8 1 1
3 8 7 15 9 6 0 2
3 7 8 11 10 15 0 2
3 2 9 1 7 11 1 1
2 7 10 16 12 9 0 2
2 6 11 2 1 16 0 2
6 5 16 3 2 14 0 3
5 8 13 6 5 15 0 3
8 7 14 9 8 16 0 3
7 6 15 12 11 13 0 3
```

Рис. 4. Пример плоского связного 4-регулярного графа с помеченными вершинами и ребрами и данные входного файла для этого графа

Таблица 1

Найденные программой *AOE*-цепи для графа, приведенного на рис. 4

Вершина	*AOE*-цепь (ребра)
1	2–13–14–15–16–12–11–9–8–7–6–5–3–4–7–10–1
2	11–16–13–14–15–9–8–6–5–3–2–12–1–4–7–10
3	8–15–16–13–14–6–5–3–2–12–11–9–10–1–4–7
4	5–14–15–16–13–3–2–12–11–9–8–6–7–10–1–4

- граф, имеющий одну точку сочленения (см. рис. 5а); найденные программой последовательности ребер, соответствующие *AOE*-цепям в указанном графе, приведены в табл. 2;

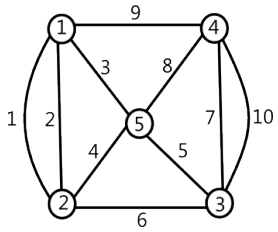
Таблица 2

Найденные программой *AOE*-цепи для графа, приведенного на рис. 5а

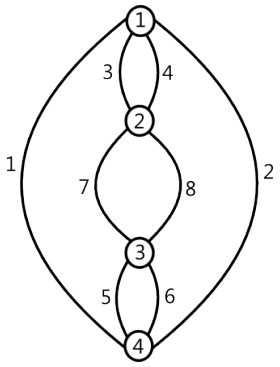
Вершина	*AOE*-цепь (ребра)
1	2–4–3–9–8–5–7–10–6–1
2	2–3–4–6–5–8–7–10–9–1
3	5–8–7–10–9–3–4–2–1–6
4	7–5–8–9–3–4–2–1–6–10

- граф, имеющий более одной точки сочленения, но все точки сочленения имеют один и тот же ранг (см. рис. 5б); найденные программой последовательности ребер, соответствующие *AOE*-цепям в указанном графе, приведены в табл. 3;
- граф, имеющий точки сочленения разных рангов (см. рис. 6); найденные программой последовательности ребер, соответствующие *AOE*-цепям в указанном графе, приведены в табл. 4.

Разработанное программное обеспечение в рассмотренных случаях корректно находит решение задачи для корректно заданных исходных данных. Так как других случаев



а) Пример плоского связного 4-регулярного графа с помеченными вершинами и ребрами, имеющего одну точку сочленения ранга 2 в вершине 5



б) Пример плоского связного 4-регулярного графа с помеченными вершинами и ребрами, имеющего две точки сочленения ранга 2 в вершинах 2 и 3

Рис. 5. Примеры графов, имеющих точки сочленения

Таблица 3

Найденные программой *AOE*-цепи для графа, приведенного на рис. 5б

Вершина	*AOE*-цепь (ребра)
1	3–4–2–6–8–7–5–1
4	5–7–8–6–2–4–3–1

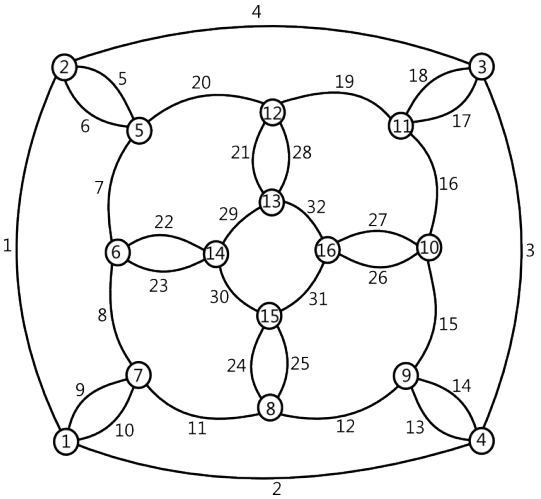


Рис. 6. Пример плоского связного 4-регулярного графа с помеченными вершинами и ребрами, имеющего точки сочленения ранга 2 в вершинах 5, 7, 9, 11 и ранга 3 в вершинах 13, 14, 15 и 16

Таблица 4

Найденные программой *AOE*-цепи для графа, приведенного на рис. 6

Вершина	*AOE*-цепь (ребра)
1	9–8–23–30–24–25–31–26–27–32–29–22–7–20–21–28–19–18–17–16–15–14–13–12–11 10–2–3–4–5–6–1
2	5–6–1–9–8–23–30–24–25–31–26–27–32–29–22–7–20–21–28–19–18–17–16–15–14–13–12–11 10–2–3–4
3	17–16–27–32–29–22–23–30–24–25–31–26–15–14–13–12–11–10–9–8–7–20–21–28–19–18–4–5–6–1–2–3
4	13–12–25–31–26–27–32–29–22–23–30–24–11–10–9–8–7–20–21–28–19–18–17–16–15–14–3–4–5–6–1–2

нахождения точек сочленения не существует, можно заключить, что для корректно заданных исходных данных программа корректно решает поставленную задачу поиска *AOE*-цепи в плоском связном 4-регулярном графе.

Заключение

Таким образом, предложенный в статье подход позволяет решить проблему маршрутизации при вырезании деталей, когда на маршрут движения режущего инструмента одновременно наложены такие технологические ограничения, как (1) отрезанная от листа часть не требует дополнительных разрезаний, (2) осуществляется переход по смежному контуру.

В статье предложен подход для построения *A*-цепи с упорядоченным охватыванием (*AOE*-цепи) в плоском связном 4-регулярном графе. Разработанный алгоритм позволяет решить задачу за полиномиальное время.

Изложенный алгоритм решения задачи маршрутизации в плоских графах решает задачу определения последовательности резки в соответствии с наложенными ограничениями по раскройному плану при использовании технологий ECP и ICP, допускающих возможность совмещения границ вырезаемых деталей. Применение этих технологий позволяет сократить расход материала, длину резки и длину холостых проходов.

В качестве направлений дальнейших исследований можно выделить разработку графического интерфейса для более удобного ввода исходных данных, создание библиотеки классов для решения задачи маршрутизации в плоских графах. Кроме того, открытой остается задача построения оптимального по длине холостых переходов покрытия несвязного графа цепями из классов *OE* и *AOE*.

Литература

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

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

3. Dewil R., Vansteenwegen P., Cattrysse D., Laguna M., Vossen T. An Improvement Heuristic Framework for the Laser Cutting Tool Path Problem // International Journal of Production

Research. 2015. Vol. 53, Issue 6. P. 1761–1776. DOI: 10.1080/00207543.2014.959268

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

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

6. Петунин А.А., Ченцов А.Г., Ченцов П.А. К вопросу о маршрутизации перемещений при листовой резке деталей // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2017. Т. 10, № 3. С. 25–39. DOI: 10.14529/mmp170303

7. Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2018. Т. 11, № 1. С. 60–74. DOI: 10.14529/mmp180106

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

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

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

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

12. Panyukova T.A. Constructing of OE-postman Path for a Planar Graph // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2014. Т. 7, № 4. С. 90–101. DOI: 10.14529/mmp140407

13. Garfinkel R.S., Webb I.R. On Crossings, the Crossing Postman Problem, and the Rural Postman Problem. // Networks. 1999. Vol. 34(3). P. 173–180.

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

15. Fleischner H. Eulerian Graphs and Related Topics. Part 1, Vol. 1.: Ann. Discrete Mathematics, 1990. № 45.

16. Филиппов А.Ф. Элементарное доказательство теоремы Жордана // Успехи математических наук. 1950. Т. 5, Вып. 5(39). С. 173–176.

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

18. Makarovskikh T., Panyukov A. Development of Routing Methods for Cutting out Details // CEUR Workshop Proceedings. 2018. Vol. 2098. P. 249–263. URL: htpp://ceur-ws.org/Vol-2098 (дата обращения: 20.07.2018).

19. Зыков А.А. Основы теории графов. М.: Вузовская книга, 2004. 664 с.

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

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

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

23. Савицкий Е.А. Использование алгоритма поиска в ширину для определения уровней вложенности ребер плоского графа // Информационные технологии и системы: Труды Третьей междунар. науч. конф. Челябинск: Изд-во ЧелГУ, 2014. С. 43–45.

24. Панюкова Т.А. Цепи с упорядоченным охватыванием в плоских графах // Дискретный анализ и исследование операций. Часть 2. 2006. Т. 13, № 2. С. 31–43.

Макаровских Татьяна Анатольевна, к.ф.-м.н., доцент, кафедра математического и компьютерного моделирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse190103

SOFTWARE FOR CONSTRUCTING OF *A*-CHAINS WITH ORDERED ENCLOSING FOR A PLANE CONNECTED 4-REGULAR GRAPH

© 2019 T.A. Makarovskikh

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: Makarovskikh.T.A@susu.ru

Received: 24.07.2018

The task of constructing the cutting path of the tool when the part cut off from the sheet does not require additional cuts and the trajectories of cuts intersections are forbidden (the touches are allowed) may arise in CAD/CAM-systems of technological preparation of cutting processes. Formally, such a task can be formulated as the problem of constructing a self-non-intersecting chain in a plane Euler graph representing a homeomorphic image of a cutting plan. Finally, the tasks of constructing routes satisfying the technological constraints are reduced to defining an A-chain with ordered enclosing for a plane connected 4-regular graph. This article is devoted to the algorithm for finding such a chain. The execution of this algorithm consists of two stages. At the first stage, cut-vertices of rank k are identified and splitted. At the second stage, the construction of the chain starts from an arbitrary vertex incident to the outer face; the first edge of the chain is chosen to be an edge of maximal rank incident to a given vertex; next, an iterative process is organized where, as the next edge, an unpassed edge of maximum rank is chosen, which is the left or right neighbour of the current one. It is shown that the algorithm constructs a route with the indicated properties in linear time for a plane connected 4-regular graph. The considered algorithms are realized as a computer program. The examples of some test problems are given in this paper.

Keywords: plane graph, path, cutting plan, polynomial-time algorithm, CAD/CAM.

FOR CITATION

Makarovskikh T.A. Software for Constructing of *A*-chains with Ordered Enclosing for a Plane Connected 4-regular Graph. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 1. pp. 36–53. (in Russian) DOI: 10.14529/cmse190103.

References

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

2. Makarovskikh T.A., Panyukov A.V., Savitsky E.A. Mathematical Models and Routing Algorithms for CAD Technological Preparation of Cutting Processes. Automation and Remote Control. 2017. vol. 78, no. 4. pp. 868–882.

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

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

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

6. Petunin A.A., Chentsov A.G., Chentsov P.A. On the Issue of Routing Movements in Sheet Cutting of Parts. *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Matematicheskoe modelirovanie i programmirovanie* [Bulletin of the South Ural State University, Series: Mathematical Modelling and Programming]. 2017. vol. 10, no. 3. pp. 25–39. DOI: 10.14529/mmp170303 (in Russian)

7. Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme. *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Matematicheskoe modelirovanie i programmirovanie* [Bulletin of the South Ural State University, Series: Mathematical Modelling and Programming]. 2018. vol. 11, no. 1. pp. 60–74. DOI: 10.14529/mmp180106 (in Russian)

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

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

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

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

12. Panyukova T.A. Constructing of OE-postman Path for a Planar Graph. Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software. 2014. vol. 7, no. 4. pp. 90–101. DOI: 10.14529/mmp140407

13. Garfinkel R.S., Webb I.R. On Crossings, the Crossing Postman Problem, and the Rural Postman Problem. Networks. 1999. vol. 34(3). pp. 173–180.

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

15. Fleischner H. *Eulerian Graphs and Related Topics. Part 1, Vol.1.* Ann. Discrete Mathematics. 1990. no. 45.

16. Filippov A.F. The Elementary Proof of Jordan Theorem *Uspekhi matematicheskikh nauk* [Successes of Mathematical Sciences]. 1950. vol. 5, no. 5(39). pp. 173–176. (in Russian)

17. Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts. IFAC-PapersOnLine. 2017. vol. 50, issue 1. pp. 2284-2289.

18. Makarovskikh T., Panyukov A. Development of Routing Methods for Cutting out Details. CEUR Workshop Proceedings. 2018. vol. 2098. pp. 249–263. Available at: `htpp://ceur-ws.org/Vol-2098` (accessed: 20.07.2018).

19. Zykov A.A. *Osnovy teorii grafov* [The Basics of Graph Theory]. Moscow, High School Book, 2004. 664 p. (in Russian)

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

21. 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

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

23. Savitskiy E.A. Using of Wide-with Search Algorithm for Ranking of Edges of a Plane Graph. *Informacionnye tekhnologii i sistemy: tr. Tret'ej mezhdunar. nauch. konf.* [Information technologies and systems: Proceedings of the Third International conference]. Chelyabinsk, Publishing of Chelyabinsk State University. 2014. pp. 43–45. (in Russian)

24. Panyukova T.A. Chains with Ordered Enclosing for Plane Graphs. *Diskretnyj analiz i issledovanie operacij. Chast' 2.* [Discrete analysis and operation research. Part 2]. 2006. vol. 13, no. 2. pp. 31–43. (in Russian)