УДК 512.5, 519.1(075.8) DOI: 10.14529/cmse190403

ПОСТРОЕНИЕ САМОНЕПЕРЕСЕКАЮЩИХСЯ *OE*-МАРШРУТОВ В ПЛОСКОМ ЭЙЛЕРОВОМ 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: 24.09.2019

В статье предложен полиномиальный алгоритм построения самонепересекающегося маршрута с упорядоченным охватыванием в плоском эйлеровом графе. Предложенный подход состоит в расщеплении всех вершин исходного графа степени выше 4 и введении фиктивных вершин и ребер, сводя, таким образом, исходную задачу к решенной ранее автором задаче построения A-цепи с упорядоченным охватыванием в плоском связном 4-регулярном графе. Приведенный алгоритм сведения решает поставленную задачу за полиномиальное время. Рассмотрен тестовый пример построения самонепересекающейся цепи с упорядоченным охватыванием. Данная задача возникает при технологической подготовке процесса раскроя, когда требуется определить маршрут движения режущего инструмента, при котором отсутствуют самопересечения траектории резки и отрезанная от листа часть не требует разрезаний. Раскройный план представлен в виде плоского графа, являющегося его гомеоморфным образом. Предложенный в статье алгоритм решает проблему маршрутизации при вырезании деталей, когда на маршрут движения режущего инструмента одновременно наложены такие технологические ограничения.

Ключевые слова: плоский граф, маршрут, раскройный план, полиномиальный алгоритм, процесс раскроя.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Макаровских Т.А. Построение самонепересекающихся *OE*-маршрутов в плоском эйлеровом графе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 4. С. 30–42. DOI: 10.14529/cmse190403.

Введение

В работе [1] поставлена задача CPDP (Cutting Path Determination Problem), заключающаяся в определении оптимального маршрута вырезания деталей по заданному раскройному плану одним или несколькими режущими инструментами. При этом предполагается наличие двух очевидных ограничений: 1) все детали должны быть вырезаны; 2) ни один из вырезанных фрагментов не должен требовать дальнейших разрезаний, т.е. выполнено OE (Odered Eclosing) ограничение [2]. Для решения проблемы CPDP известны более детальные постановки: GTSP (General Travelling Salesman Problem) [3–11], CCP (Continuous Cutting Problem), ECP (Endpoint Cutting Problem) [12] и ICP (Intermittent Cutting Problem), [2]. Отметим, что ECP и ICP допускают совмещение границ вырезаемых деталей, что позволяет сократить расход материала, длину резки и длину холостых проходов [4]. Проблемы уменьшения отходов материала и максимального совмещения фрагментов контуров вырезаемых деталей решаются на этапе составления раскройного плана.

Несмотря на отмеченные преимущества компьютерных технологий ECP и ICP, в настоящее время большинство публикаций посвящено развитию технологий GTSP и CCP, которые используют очевидные алгоритмы маршрутизации режущего инструмента, состоящие в поконтурном вырезании деталей.

Развитию компьютерных технологий ECP и ICP посвящены работы [2, 12] и [13]. В них даны полиномиальные алгоритмы OE-маршрутизации, когда отрезанная от листа часть не требует дальнейших разрезаний.

Большую как теоретическую, так и практическую ценность представляет построение самонепересекающихся OE-маршрутов (NOE-маршрутов) в плоских эйлеровых графах. Под самонепересекающимся маршрутом имеется в виду циклический граф, представляющий плоскую жорданову кривую без самопересечений [16], полученный в результате расщепления вершин исходного графа. Под расщеплением понимается определенная в [18] операция, которая заключается в следующем. Пусть G — связный граф, $v \in V(G)$ — вершина степени $\deg(v) \geq 3$. Если $x = \{v, v_1\}$ и $y = \{v, v_2\}$ — пара ребер, инцидентных v, тогда под отщеплением пары ребер $\{x, y\}$ от вершины v понимается получение нового графа $G_{x,y}$, полученного из G удалением ребер x и y, введением новой вершины v_{xy} и пары ребер $\{v_{xy}v_1\}$ и $\{v_{xy}v_2\}$ (см. [14, рис. 1]).

В частном случае, когда раскройный план оказывается гомеоморфен плоскому связному 4-регулярному графу, в работе [15] предложен полиномиальный алгоритм построения AOE-цепи, являющейся самонепересекающейся цепью. Данный алгоритм позволяет решать задачи маршрутизации в плоских связных графах со степенями вершин, не превосходящими 4.

Отметим приведенное в [17] «доказательство» $\mathcal{NP}$-полноты задачи построения самонепересекающейся цепи в плоском эйлеровом графе. Рассуждения базируются на определении непересекающейся цепи, являющемся определением A-цепи [18] (т. е.цепи, в которой очередным в маршруте ребром является следующее ребро в определенном для текущей вершины циклическом порядке). Очевидно, что A-цепь представляет лишь частный случай самонепересекающейся цепи.

В данной статье предложен полиномиальный алгоритм для построения самонепересекающейся цепи в плоском связном эйлеровом графе. Предложенный в статье алгоритм решает задачу маршрутизации при вырезании деталей, когда выполняются два ограничения: отрезанная от листа часть не требует дополнительных разрезаний [2, 19, 20] и в траектории резки отсутствуют пересечения [21].

Статья организована следующим образом. В разделе 1 приведены необходимые определения, описаны используемые обозначения для представления данных. В разделе 2 рассматривается класс самонепересекающихся OE-цепей (называемых далее NOE-цепями). Цепи указанного класса соответствуют траектории движения режущего инструмента, избегающей пересечения траектории резания. Показано, что задача построения NOE-цепи в плоском связном эйлеровом графе может быть за полиномиальное время сведена к задаче построения AOE-цепи в плоском связном 4-регулярном графе. Приведен алгоритм такого сведения. В разделе 3 рассмотрен тестовый пример построения NOE-цепи. В заключении перечислены полученные в работе результаты, отмечены направления дальнейших исследований.

1. Определения и обозначения, используемые для представления данных

В данной работе будем использовать представление графа, используемое автором в предыдущих работах [2, 15, 19–21]. В работах автора предложено вместо раскройного плана использовать его гомеоморфный образ, представляющий 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 если J — начальная часть маршрута, то $\mathrm{Int}(J)$ можно интерпретировать как отрезанную от листа часть. 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 $v_{3-k}(e)$, $k = 1, 2$.

Пример представления графа подробно рассмотрен в [2].

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 замена индексов функций $v_k(e)$, $l_k(e)$, $r_k(e)$ и $f_k(e)$ на $3 - k$, $k = 1, 2$ [20].

Далее будем считать, что все рассматриваемые плоские графы представлены указанными функциями. Пространственная сложность такого представления будет $O(|E(G)| \cdot \log_2 |V(G)|)$ [15]. В дальнейшем будем использовать ряд понятий, определения которых имеются в работах [13, 18, 22]. Приведем основные из них для удобства читателя.

Определение 1. Будем говорить, что цикл $C = v_1e_1v_2e_2 \dots v_k$ в эйлеровом графе G имеет **упорядоченное охватывание** (называется OE-цепью), если для любой его начальной части $C_i = v_1e_1v_2e_2 \dots e_i$, $i \leq |E(G)|$ выполнено условие $\mathrm{Int}\,(C_i) \cap G = \emptyset$ [13].

Определение 2. Эйлерову цепь T будем называть **A-цепью** [18], если она является A_G-совместимой цепью. Таким образом, последовательные ребра в цепи T (инцидентные вершине v) являются соседями в циклическом порядке $O^{\pm}(v)$ [18].

Определение 3. Рангом ребра $e \in E$ графа $G = (V, E)$ будем называть значение функции $\mathrm{rank}(e) : E \to \mathbb{N}$, 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 — множество ребер ранга 1 графа

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

тогда $(\forall e \in E_k)\,(\mathrm{rank}(e) = k)$ [13].

Алгоритм 1 `NOE-CHAIN (G)`

Require: плоский эйлеров граф G, заданный функциями $v_k(e)$, $l_k(e)$, $r_k(e)$, $f_k(e)$, $k = 1, 2$ и $\mathrm{rank}(e)$;
Ensure: NOE-цепь в графе G;
1: $\widehat{G} := \mathsf{NonIntersecting}(G)$; ▷ Расщепить все вершины степени выше 4
2: $\widetilde{G} := \mathsf{CutPointSplitting}(\widehat{G})$; ▷ Расщепить все точки сочленения всех рангов
3: $C^*{:=}\mathsf{AOE\_TRAIL}(\widetilde{G})$; ▷ Построить AOE-цепь в графе $\tilde{G}$
4: $C{:=}\mathsf{Absorb}(C^*)$; ▷ Стянуть все расщепленные вершины

Ранг ребра определяет его удаленность от внешней грани и показывает, какое минимальное число граней необходимо пересечь, чтобы добраться от внешней грани f_0 до этого ребра.

Определение 4. Рангом грани $f \in F(G)$ будем называть значение функции $\mathrm{rank} : F(G) \to \mathrm{Z}^{\geq 0}$:

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

где $E(f)$ — множество ребер инцидентных грани $f \in F$ [2].

2. Алгоритм построения NOE-цепи

Рассмотренный в [15] класс AOE-цепей достаточно узок. К тому же в общем случае не известны эффективные алгоритмы построения таких цепей. Для практических задач оказывается достаточным построение не AOE-цепи, а самонепересекающейся OE-цепи.

Определение 5. Эйлеров цикл в C плоском эйлеровом графе G называется самонепересекающимся если он гомеоморфен плоской замкнутой жордановой кривой [16] без самопересечений, который может быть получен из графа $G = (V, E)$ с помощью применения $|E|$ операций расщепления вершин.

Для построения самонепересекающейся эйлеровой OE-цепи (или цикла) в плоском эйлеровом графе (в дальнейшем эту цепь будем называть NOE-цепью (non-intersecting OE-trail)) можно воспользоваться алгоритмом 1.

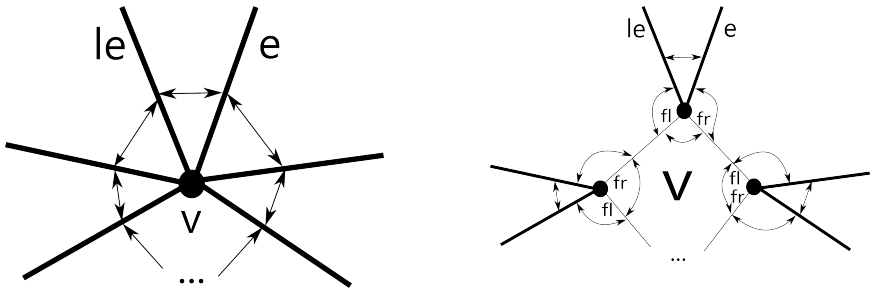
Процедура `Non-intersecting (G)` (алгоритм 2) строит 4-регулярный граф $\widehat{G}$ (с точностью до гомеоморфизма) расщепляя в графе G все вершины $v \in V(G)$ степени $2l$ ($l \geq 3$) на l фиктивных вершин степени 4 и вводит l фиктивных ребер, инцидентных полученным после расщепления вершинам и образующим цикл (см. рис. 1а и 1б). Для выполнения указанных преобразований необходимо просмотреть функции $v_k(e)$, $k = 1, 2$ для всех ребер $e \in E(G)$, и внести требуемые модификации в систему кодирования графа. С этой целью на множестве вершин графа $V(G)$ определена булева функция

$$\mathrm{Checked}(v) = \begin{cases} \texttt{true}, & \text{если вершина просмотрена;} \\ \texttt{false}, & \text{в противном случае.} \end{cases}$$

При выполнении инициализации (строки 1–3 в описании алгоритма 2) все вершины объявляют непросмотренными, т. е. $\mathrm{Checked}(v) =$ `false` для всех $v \in V(G)$. Просмотр вершины $v = v_1(e)$, такой что $\mathrm{Checked}(v) =$ `false` состоит в выполнении процедуры `Handle (e)` (алгоритм 3). которая производит обработку данной вершины, заключающуюся в ее расщеплении в соответствии с рис. 1а и 1б.

Алгоритм 2 Процедура `Non-intersecting (`G`)`

```
 1: procedure NON-INTERSECTING(G)
Require: плоский эйлеров граф G, заданный функциями v_k(e), l_k(e), r_k(e), f_k(e), k = 1,2 и rank(e);
Ensure: плоский связный 4-регулярный граф G*, определяемый аналогичным образом;
 2:     for all v ∈ V(G) do                          ▷ Инициализация функции Checked(v)
 3:         Checked(v) := false;
 4:     end for
 5:     for all (e ∈ E(G) ) do                       ▷ Поиск вершин степени больше 4 и их расщепление
 6:         k := 1;                                  ▷ Просмотреть вершину с индексом 1, затем – 2
 7:         while (k ≤ 2) do
 8:             if (not Checked(v_k(e))) then        ▷ Обработать только не обработанную ранее вершину
 9:                 if (k = 2) then                  ▷ Скорректировать индексы
10:                     REPLACE(e);                  ▷  обрабатываются вершины v_1(e)
11:                 end if
12:                  Handle ( e);                    ▷ Вызвать функцию для обработки вершины v_1(e)
13:                 Checked(v_1(e)) := true;         ▷ Пометить вершину как просмотренную
14:             end if
15:             k := k + 1;
16:         end while
17:     end for
18: end procedure
```



а) Исходные указатели на соседние ребра в расщепляемой вершине

б) Расщепление вершины (жирными линиями показаны ребра графа G, тонкими линиями – дополнительные (фиктивные) ребра) и модификация указателей в соответствии с расщеплением

Рис. 1. Расщепление вершины степени выше 4 и модификация указателей на ребра

Алгоритм 3 в результате цикла `repeat -until` (строки 6–11) подсчитывает степень d текущей вершины v. Если $d > 4$, выполняется второй цикл `repeat -until` (строки 12–23), в котором обрабатываемая вершина расщепляется на $d/2$ фиктивных вершин, вводятся d фиктивных ребер, инцидентных этим вершинам и образующим цикл.

Отметим, что строки 18–23 затрагивают не только изменение указателей на ребра, но и вводят новую (фиктивную) грань F, инцидентную всем фиктивным вершинам и ребрам, а также определяют ранги фиктивных ребер.

Алгоритм 3 Процедура Handle (e)

```
1: procedure HANDLE(e)
2:     v := v_1(e);                                          ▷ Расщепляемая вершина
3:     e_first := e;                                         ▷ Сохранить первое рассматриваемое ребро
4:     d := 0;                                               ▷ Инициализация счетчика для степени вершины d
5:     F := FaceNum() + 1;                                   ▷ Определить номер для новой грани
6:     repeat                                                ▷ Проход 1: Определение степени вершины v
7:         le := l_1(e);
8:         if (v_1(le) ≠ v) then REPLACE(le);
9:         end if                                            ▷ При необходимости поменять индексацию функций
10:        e := le; d := d + 1;                              ▷ Учесть ребро при подсчете степени и перейти к следующему
11:    until (e = e_first);                                  ▷ Повторять, пока не будут просмотрены все ребра, инцидентные v
12:    if (d > 4) then                                       ▷ Если степень текущей вершины больше 4
13:        e := e_first;                                     ▷ Начать с первого рассматриваемого ребра
14:        le := l_k (e);                                    ▷ Определить номер его левого соседа
15:        e_next := l_k (le);                               ▷ Сохранить ребро, для следующей итерации
16:        fl := new EDGE; fle := fl; e_first := e;          ▷ Ввести фиктивное ребро, смежное le
17:        repeat                                            ▷ Расставить указатели для ребер
18:            e := e_next; le := l_k (e); fr := fl;
19:            f_1(fl) := F; f_2(fl) := f_2(e);              ▷ Определить грани, смежные фиктивному ребру
20:            rank(fl) := facerank(f_2(fl));                ▷ Определить «ранг» фиктивного ребра
21:                    ▷ Функция facerank() вычисляет ранг грани в соответствии с определением
22:            fl := new EDGE; e_next := l_k (le);
23:        until (l_k(le) = e_first);
24:    end if
25: end procedure
```

Определение 6. Ранг фиктивного ребра (строка 20) равен рангу инцидентной фиктивному ребру грани исходного графа.

Для 4-регулярного графа $\widehat{G}$ с определенными для него рангами фиктивных ребер и введенными в его представление фиктивными гранями можно применить последовательно алгоритм `CUT-POINT-SPLITTING`($\widehat{G}$) построения графа $\widetilde{G}$ с расщепленными точками сочленения (понятие точки сочленения ранга k см. в [15, Определение 12]), и алгоритм `AOE-TRAIL`($\widetilde{G}$) [15] построения AOE-цепи C^* в графе $\widetilde{G}$. При построении цепи C^* алгоритм `AOE-TRAIL`($\widetilde{G}$) при наличии двух смежных непройденных ребер одного ранга для гарантированного выполнения условия упорядоченного охватывания в первую очередь выбирает фиктивное ребро.

Процедура `Absorb`(*) заменяет в * все фиктивные ребра и инцидентные им вершины, полученные при расщеплении вершины v (выполняет операцию стягивания фиктивных вершин). В результате выполнения процедуры получим NOE-цепь C в исходном графе G. Цепь C, полученная после удаления фиктивных ребер за счет стягивания вершин, будет принадлежать классу OE, т.к. процедура удаления ребер не нарушает порядка следования оставшихся ребер в цепи, что исключает появление цикла, охватывающего еще непройденные ребра.

Так как процедура `Handle` состоит из двух последовательных просмотров ребер, инцидентных текущей вершине v, то вычислительная сложность процедуры равна $O(|E(G)|)$. Процедура `Non-Intersecting` заключается в однократном просмотре всех

ребер, то есть ее вычислительная сложность также составляет величину $O(|E(G)|)$. Следовательно, алгоритм сведения плоского связного эйлерова графа к плоскому связному 4-регулярному графу решает поставленную задачу за время $O(|E(G)|^2)$. Поскольку алгоритм `AOE-TRAIL` и предшествующая его вызову функция `CUT-POINT-SPLITTING` [15] решают задачу построения AOE-цепи C за время $O(|E(G^*)| \cdot \log_2 |V(G^*)|)$, то и задача построения NOE-цепи в графе G решается за полиномиальное время $O(|E(G)|^2)$.

Из сказанного выше в данном разделе следует справедливость следующей теоремы.

Теорема 1. Алгоритм `NOE-CHAIN` решает задачу построения NOE-цепи в плоском эйлеровом графе за время $O(|E(G)|^2)$.

3. Иллюстрация работы алгоритма

Рассмотрим работу алгоритма на примере графа, приведенного на рис. 2а. Пример затрагивает общий случай: в графе имеется вершина степени выше 6, не смежная внешней грани, а также присутствуют (и возникают в процессе расщепления) точки сочленения разных рангов. Рассматриваемый граф является эйлеровым, следовательно, построение NOE-цепи можно начать из любой вершины, смежной внешней грани. Пусть это будет вершина v_2.

После применения процедуры `Handle()` получим граф, представленный на рис. 2б. В полученном графе все вершины имеют степень 4. В графе на рис. 2б помимо расщепленной вершины v_3 имеются и точки сочленения v_4 и v_6 рангов 3 и 2 соответственно, а также точка сочленения ранга 2 в расщепленной вершине v_3, инцидентная ребрам e_{13}, e_4 и двум фиктивным ребрам того же ранга. Эти вершины необходимо расщепить с помощью алгоритма `CUT-POINT-SPLITTING`. В результате выполнения указанного алгоритма получим граф, представленный на рис. 2в. С помощью алгоритма `AOE-Trail` в полученном графе определим AOE-маршрут, в котором символом «*» обозначен переход по фиктивным ребрам

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

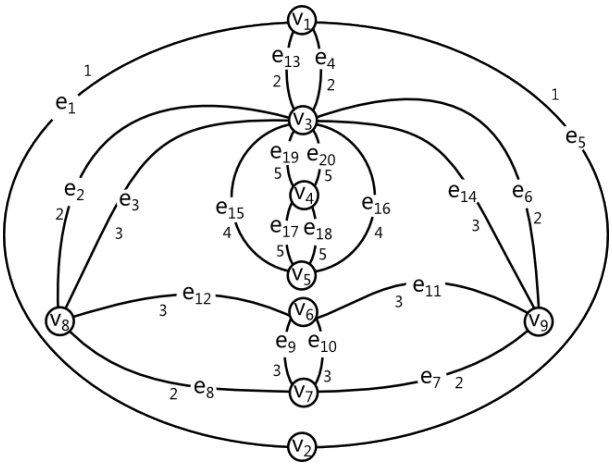
которому после стягивания расщепленных вершин соответствует NOE-маршрут

$$\begin{aligned} T^* \;=\; & v_2e_5v_1e_4v_3e_{19}v_4e_{17}v_5e_{18}v_4e_{20}v_3e_{16}v_5e_{15}v_3 \\ & e_3v_8e_{12}v_6e_9v_7e_{10}v_6e_{11}v_9e_{14}v_3e_6v_9e_7v_7e_8v_8e_2v_3e_{13}v_1e_1v_2, \end{aligned}$$

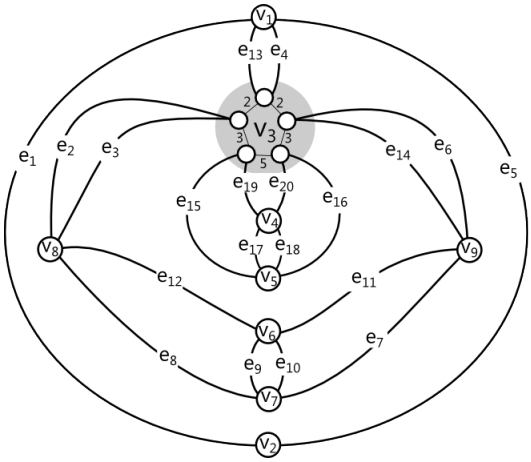
в исходном графе.

Заключение

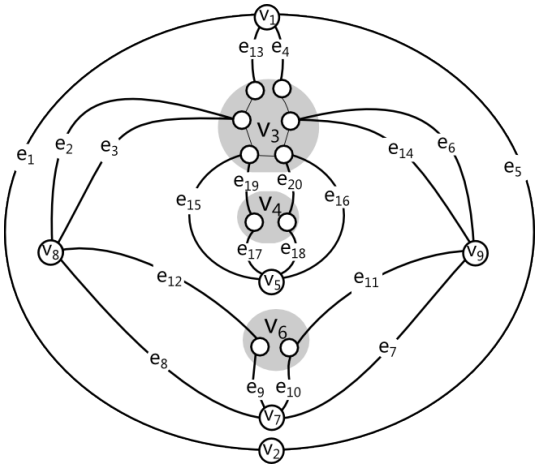
Предложенный в работе алгоритм строит NOE-цепь в плоском эйлеровом графе. В случае плоского неэйлерова (в общем случае несвязного) графа G без висячих вершин необходимо расщепить все вершины степени выше 4 в соответствии с алгоритмом 3. В результате получим граф, степени вершин которого равны 3 или 4 (не уменьшая общности рассуждений, вершины степени 2 не рассматриваются). Для этого графа применима та же последовательность действий, что описана в [15] для построения AOE-покрытия. В цепях полученного покрытия удалим все искусственные ребра, стягивая все расщепленные вершины. В результате получим NOE-покрытие.



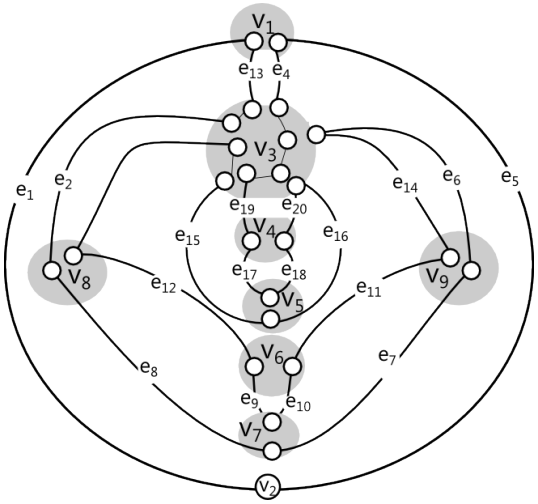
а) Граф G



б) Менее, чем 4-регулярный граф $\widehat{G}$



в) граф $\widetilde{G}$ без точек сочленения



г) результирующий самонепересекающийся OE-маршрут C

Рис. 2. Пример построения NOE-цепи в плоском эйлеровом графе, имеющем вершину степени выше 6, не смежную внешней грани

Предложенный в статье алгоритм решает 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) отсутствуют самопересечения траектории резки.

В качестве направлений дальнейших исследований можно выделить создание библиотеки классов для решения задачи маршрутизации в плоских графах.

Литература

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

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. DOI: 10.1016/10.1134/S0005117917050095.

3. 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.

4. 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, no. 6. P. 1761–1776. DOI: 10.1080/00207543.2014.959268.

5. Hoeft J., Palekar U. Heuristics for the Plate-cutting Travelling Salesman Problem // IIE Transactions. 1997. Vol. 29(9). P. 719–731. DOI: 10.1023/A:1018582320737.

6. 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. DOI: 10.1080/00207543.2014.895064.

7. 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.

8. 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.

9. 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. DOI: 10.1016/j.ifacol.2016.07.544.

10. 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. DOI: 10.1016/j.ifacol.2016.07.767.

11. 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. DOI: 10.1007/978-3-319-93800-4_6.

12. 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.

13. 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.

14. Borse Y.M. Splitting Lemma for 2-Connected Graphs // International Scholarly Research Network, ISRN Discrete Mathematics. 2012. Vol. 2012, art. 850538. DOI: 10.5402/2012/850538.

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

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

17. 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.

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

19. 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.

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. Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts // IFAC-PapersOnLine. 2017. Vol. 50, no. 1. P. 2284–2289. DOI: 10.1016/j.ifacol.2017.08.226.

22. 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.

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/cmse190403

CONSTRUCTING SELF-NON-INTERSECTING OE-CHAINS IN A PLANE EULERIAN GRAPH

© 2019 T.A. Makarovskikh

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

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

Received: 24.09.2019

The paper is devoted to a polynomial-time algorithm for constructing a self-non-intersecting ordered enclosing chain for a plane Eulerian graph. The proposed approach consists in splitting all the vertices of the original graph of degree higher than 4 and introducing fictive vertices and edges and, thus, reducing the considered earlier problem to the problem of finding an A-chain with ordered enclosing in a plane connected 4-regular graph. The presented reduction algorithm solves the problem in polynomial time. A test example of constructing a self-non-intersecting chain with ordered enclosing is considered. This problem arises during the technological preparation of the cutting process, when it is necessary to determine the path of the cutter when there are no self-intersections of the cutting path and the part cut off from the sheet does not require any cuts. The cutting plan may be presented as a planar graph which is the homeomorphic image of the cutting plan. The algorithm proposed in the article solves the problem of routing when cutting parts when such technological constraints are simultaneously imposed on the path of the cutter.

Keywords: plane graph, path, cutting plan, polynomial-time algorithm, cutting process.

FOR CITATION

Makarovskikh T.A. Constructing Self-non-intersecting OE-chains in a Plane Eulerian Graph. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 4. pp. 30–42. (in Russian) DOI: 10.14529/cmse190403.

References

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

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. DOI: 10.1016/10.1134/S0005117917050095.

3. 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.

4. 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, no. 6. pp. 1761–1776. DOI: 10.1080/00207543.2014.959268.

5. Hoeft J., Palekar U. Heuristics for the Plate-cutting Travelling Salesman Problem. IIE Transactions. 1997. vol. 29(9). pp. 719–731. DOI: 10.1023/A:1018582320737.

6. 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. DOI: 10.1080/00207543.2014.895064.

7. Petunin A.A., Chentsov A.G., Chentsov P.A. On the Issue of Routing Movements in Sheet Cutting of Parts. Bulletin of the South Ural State University. Series: Mathematical Modelling and Programming. 2017. vol. 10, no. 3. pp. 25–39. (in Russian) DOI: 10.14529/mmp170303.

8. Chentsov A.G., Grigoryev A.M., Chentsov A.A. Solving a Routing Problem with the Aid of an Independent Computations Scheme. Bulletin of the South Ural State University. Series: Mathematical Modelling and Programming. 2018. vol. 11, no. 1. pp. 60–74. (in Russian) DOI: 10.14529/mmp180106.

9. 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. DOI: 10.1016/j.ifacol.2016.07.544.

10. 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. DOI: 10.1016/j.ifacol.2016.07.767.

11. 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. DOI: 10.1007/978-3-319-93800-4_6.

12. 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.

13. 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.

14. Borse Y.M. Splitting Lemma for 2-Connected Graphs. International Scholarly Research Network, ISRN Discrete Mathematics. 2012. vol. 2012. Art. 850538. DOI: 10.5402/2012/850538.

15. 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.

16. Filippov A.F. The Elementary Proof of Jordan Theorem. Advances in Mathematical Sciences. 1950. vol. 5:5(39). pp. 173–176. (in Russian)

17. 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.

18. Fleischner H. Eulerian Graphs and Related Topics. Part 1, vol. 1. Ann. Discrete Mathematics. 1990. no. 45.

19. 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.

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. Makarovskikh T., Panyukov A. The Cutter Trajectory Avoiding Intersections of Cuts. IFAC-PapersOnLine. 2017. vol. 50, no. 1. pp. 2284–2289. DOI: 10.1016/j.ifacol.2017.08.226.

22. 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.