АНТИПЛАГИАТ
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Отчет о проверке на заимствования №1

Автор: УНИВЕРИС univeris@susu.ru / ID: 640
Проверяющий: (univeris@susu.ru / ID: 640)
Организация: Южно-Уральский государственный университет

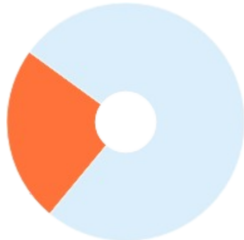
Отчет предоставлен сервисом «Антиплагиат» - http://susu.antiplagiat.ru

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 141911
Начало загрузки: 24.09.2019 20:41:36
Длительность загрузки: 00:00:05
Имя исходного файла: Makarovskikh!!!.pdf
Размер текста: 795 кБ
Символов в тексте: 19682
Слов в тексте: 2737
Число предложений: 95

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 24.09.2019 20:41:42
Длительность проверки: 00:00:45
Комментарии: не указано
Модули поиска: Модуль поиска ИПС "Адилет", Модуль выделения библиографических записей, Сводная коллекция ЭБС, Коллекция РГБ, Цитирование, Модуль поиска переводных заимствований, Коллекция eLIBRARY.RU, Коллекция ГАРАНТ, Модуль поиска Интернет, Коллекция Медицина, Модуль поиска перефразирований eLIBRARY.RU, Модуль поиска перефразирований Интернет, Коллекция Патенты, Модуль поиска общеупотребительных выражений, Модуль поиска "ЮУрГУ", Кольцо вузов



ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
23,91%	0%	76,09%

Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в отчете	Доля в тексте	Источник	Ссылка	Актуален на	Модуль поиска	Блоков в отчете	Блоков в тексте
[01]	7,3%	9,52%	Полный текст	http://istina.msu.ru	29 Янв 2017	Модуль поиска перефразирований Интернет	1437	5
[02]	5,52%	8,58%	Построение эйлеровых циклов с упоря.	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	1086	6
[03]	0,71%	6,25%	Скачать Скачать PDF	http://journal.ugatu.ac.ru	раньше 2011	Модуль поиска Интернет	140	12
[04]	0,36%	6,15%	ITIDS'2017. Труды V Всероссийской кон..	https://docplayer.ru	18 Июн 2019	Модуль поиска Интернет	71	14
[05]	0,67%	6,02%	МАТЕМАТИЧЕСКИЕ МОДЕЛИ И АЛГОР...	http://elibrary.ru	03 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	132	6
[06]	1,23%	4,78%	Скачать этот файл PDF	http://vestnik.susu.ru	30 Янв 2017	Модуль поиска перефразирований Интернет	243	4
[07]	0,36%	4%	АБСТРАГИРОВАНИЕ РАСКРОЙНОГО ПЛ.	http://elibrary.ru	05 Авг 2016	Коллекция eLIBRARY.RU	71	9
[08]	0%	3,89%	https://istina.msu.ru/download/2638908..	https://istina.msu.ru	15 Сен 2018	Модуль поиска Интернет	0	10
[09]	1,49%	3,68%	АБСТРАГИРОВАНИЕ РАСКРОЙНОГО ПЛ.	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	294	3
[10]	0,55%	3,24%	AOE-Trails Constructing for a Plane Conn..	http://ceur-ws.org	05 Янв 2018	Модуль поиска переводных заимствований	108	2
[11]	0,01%	3,15%	ПОСЛЕДОВАТЕЛЬНОСТИ ЦЕПЕЙ С УПО..	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	2	3
[12]	0%	2,99%	Построение эйлеровых циклов с упоря.	http://elibrary.ru	01 Янв 2017	Коллекция eLIBRARY.RU	0	5
[13]	0,05%	2,99%	Построение эйлеровых циклов с упоря.	http://docplayer.ru	03 Ноя 2018	Модуль поиска Интернет	10	5
[14]	0%	2,99%	ОЦЕНКА МОЩНОСТИ <i>OE</i>-ПОКРЫ.	http://elibrary.ru	31 Авг 2017	Коллекция eLIBRARY.RU	0	5
[15]	0,16%	2,78%	ОПТИМИЗАЦИЯ ИСПОЛЬЗОВАНИЯ РЕ...	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	32	2

[16]	0%	2,6%	Панюкова, Татьяна Анатольевна Свой...	http://dlib.rsl.ru	раньше 2011	Коллекция РГБ	0	5
[17]	0%	2,54%	Макаровских, Татьяна Анатольевна По..	http://dlib.rsl.ru	30 Апр 2015	Коллекция РГБ	0	9
[18]	0,23%	2,41%	О ЧИСЛЕ <i>OE</i>-ЦЕПЕЙ ДЛЯ ЗАДАНН..	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	46	2
[19]	2,21%	2,27%	Загрузить (3.3 MB) (8/12)	http://iitp.ru	05 Янв 2017	Модуль поиска перефразирований Интернет	435	1
[20]	0%	2,12%	260888	http://e.lanbook.com	10 Мар 2016	Сводная коллекция ЭБС	0	6
[21]	0,51%	2,03%	Информационно-телекоммуникацион..	http://elibrary.ru	02 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	101	2
[22]	0%	2%	Скачать этот файл PDF	https://vestnik.susu.ru	06 Авг 2017	Модуль поиска Интернет	0	4
[23]	0%	1,97%	ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПО..	http://elibrary.ru	17 Дек 2016	Коллекция eLIBRARY.RU	0	7
[24]	0%	1,97%	Скачать этот файл PDF (7/7)	http://vestnik.susu.ru	раньше 2011	Модуль поиска Интернет	0	7
[25]	1,94%	1,93%	Скачать статью	http://math.nsc.ru	29 Янв 2017	Модуль поиска перефразирований Интернет	381	1
[26]	0%	1,58%	О ЧИСЛЕ <i>OE</i>-ЦЕПЕЙ ДЛЯ ЗАДАНН..	http://elibrary.ru	04 Авг 2016	Коллекция eLIBRARY.RU	0	3
[27]	0,6%	1,47%	Задачи маршрутизации специального..	http://tekhnosfera.com	08 Янв 2017	Модуль поиска перефразирований Интернет	118	2
[28]	0%	1,4%	ОЦЕНКА МОЩНОСТИ <i>OE</i>-ПОКРЫ..	http://elibrary.ru	03 Янв 2018	Модуль поиска перефразирований eLIBRARY.RU	0	1
[29]	0%	1,25%	ПО СЛЕДАМ НАУЧНОЙ КОНФЕРЕНЦИИ..	http://elibrary.ru	05 Авг 2016	Коллекция eLIBRARY.RU	0	4
[30]	0%	0,6%	Магистерская работа // Задачи маршр..	http://bankrabot.com	01 Янв 2017	Модуль поиска перефразирований Интернет	0	1

Текст документа

УДК 512.5, 519.1(075.8)

ПОСТРОЕНИЕ САМОНЕПЕРЕСЕКАЮЩИХСЯ

ОЕ-МАРШРУТОВ В ПЛОСКОМ ЭЙЛЕРОВОМ ГРАФЕ

с 2019 г. 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 сентября 2019 г.

В статье предложен полиномиальный алгоритм построения самонепересекающегося маршрута с упо-

рядоченным 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 упорядо-

ченным охватыванием. Данная задача возникает при технологической подготовке процесса раскроя, когда

требуется определить маршрут движения режущего инструмента, при котором отсутствуют самопересече-

ния 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 реша-

ет 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 рас-

кроя

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Макаровских Т.А. Построение самонепересекающихся ОЕ-маршрутов в плоском эй-

леровом графе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика.

2019. Т. X, No Y. С. Z1–Z2. DOI: 10.14529/cmseXXXXX.

Введение

В работе [1] поставлена задача CPDP (Cutting Path Determination Problem), заключа-

ющаяся 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, 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, полученный в результате расщепления вершин исходного графа. В частном случае, когда раскройный план оказывается гомеоморфен плоскому связному 4-регулярному графу, в работе [14] 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 [15] «доказательство» $\mathscr{N} \mathscr{P}$-полноты задачи построения самонепересекающейся цепи в плоском эйлеровом графе. Рассуждения базируются на определении непересекающейся цепи, являющемся определением A-цепи [16] (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, 17, 18] и в траектории резки отсутствуют пересечения [19].

В разделе 1 приведены необходимые определения, описаны используемые обозначения для представления данных. В разделе 2 рассматривается класс самонепересекающихся OE-цепей (или NOE-цепей). 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 связом 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, 14, 17–19]. 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 $\mathrm{Int}(J)$ 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, 18]:

\bullet vk(e), k = 1, 2 — вершины, инцидентные ребру e;

\bullet lk(e), k = 1, 2 — ребра, полученные вращением ребра e против часовой стрелки вокруг вершины vk(e);

\bullet rk(e), k = 1, 2 — ребра, полученные вращением ребра e по часовой стрелке вокруг вершины vk(e);

\bullet fk(e) — грань , находящаяся справа при движении по ребру e от вершины vk(e) к вершине v3 - k(e), k = 1, 2.

Пример предста вления графа подробно рассмотрен в [2].

Представление графа фактически задает ориентацию его ребер. Далее предполагается, что движение по ребру для определенности осуществляется от вершины v1(e) к вершине v2(e). Поскольку при задании графа G неизвестно, какое из ребер в каком направлении будет пройдено, то при выполнении алгоритма производится перестановка значений полей vk(e), rk(e) и lk(e), fk(e), k = 1, 2 некоторых ребер. В алгоритме данную процедуру выпол- няет функц ия REPLACE , функциональным назначением которой является замена индексов функций vk(e), lk(e), rk(e) и fk(e) на 3 - k, k = 1, 2 [18].

Далее будем считать, что все рассматриваемые плоские графы представлены указан- ными функциями. Пространственная сложность такого представления будет O(| E(G) | \cdot log2 | V (G) |) [14]. В дальнейшем будем использовать ряд понятий, определения которых име- ются в работах [13, 16, 20]. Привед ем основные из них для удобства читателя.

Определение 1. Будем говорить, что цикл C = v1e1v2e2 . . . vk в эйлеровом графе G имеет упорядоченное охватывание (называется OE-цепью), если для любой его начальной части Ci = v1e1v2e2 . . . ei, i \leq | E(G) | выполнено условие Int (Ci) \cap G = \emptyset [13].

Определение 2. Эйлерову цепь T будем называть A-цепью [16], если она является AG- совместимой цепью. Таким образом, последовательные ребра в цепи T (инцидентные вер- шине v) являются соседями в циклическом порядке O \pm (v) [16].

Определение 3. Рангом ребра e \in E(G) будем называть значение функции rank(e) : E(G) \rightarrow N, определяемое рекурсивно:

\bullet пусть E1 = \{ e \in E : e \subset f0 \} – множество ребер, ограничивающих внешнюю грань f0 графа G(V,E), тогда (\forall e \in E1)(rank(e) = 1);

\bullet пусть Ek(G) – множество ребер ранга 1 графа

Gk \Biggl(V,E \setminus \Biggl(k l=\bigcup - 1 1

El \Biggr) \Biggr),

тогда (\forall e \in Ek)(rank(e) = k) [13].

Ранг ребра определяет его удаленность от внешней грани и показывает, какое мини- мальное число граней необходимо пересечь, чтобы добраться от внешней грани f0 до этого ребра.

Определение 4. Рангом грани f \in F(G) будем называть значение функции rank : F(G) \rightarrow Z \geq 0:

rank(f)= \Biggl\{ 0 , при f = f0,

mine \in E(f) rank(e), в противном случае,

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

Algorithm 1 NOE-CHAIN (G)

Require: плоский эйлеров граф G, заданный функциями vk(e), lk(e), rk(e), fk(e), k = 1, 2 и rank(e);

Ensure: NOE-цепь в графе G;

1: G \widehat = \sansN \sanso \sansn \sansl \sansn \sanst \sanse \sansr \sanss \sanse \sansc \sanst \sansi \sansn \sansg (G); \triangleleft Расщепить все вершины степени выше 4

2: G \widetilde = \sansC \sansu \sanst \sansP \sanso \sansi \sansn \sanst \sansS \sansp \sansl \sansi \sanst \sanst \sansi \sansn \sansg (G)\widehat ; \triangleleft Расщепить все точки сочленения всех рангов

3: C \ast =AOE_TRAIL(G)\widetilde ; \triangleleft Построить AOE-цепь в графе G \~

4: C=Absorb(C \ast); \triangleleft Стянуть все расщепленные вершины

Algorithm 2 Функция Non-intersecting (G)

Require: плоский эйлеров граф G, заданный функциями vk(e), lk(e), rk(e), fk(e), k = 1, 2 и rank(e);

Ensure: плоский связный 4-регулярный граф G \ast, определяемый аналогичным образом;

1: for all v \in V (G) do \triangleleft Инициализация функции Checked(v)

2: Checked(v) := \bff \bfa \bfl \bfs \bfe ;

3: end for

4: for all (e \in E(G)) do \triangleleft Поиск вершин степени больше 4 и их расщепление

5: k := 1; \triangleleft Просмотреть вершину с индексом 1, затем – 2

6: while (k \leq 2) do

7: if (! Checked(vk(e))) then \triangleleft Обработать только не обработанную ранее вершину

8: if (k = 2) then \triangleleft Скорректировать индексы

9: REPLACE(e); \triangleleft обрабатываются вершины v1(e)

10: end if

11: Handle (e); \triangleleft Вызвать функцию для обработки вершины v1(e)

12: Checked(v1(e)) := \bft \bfr \bfu \bfe ; \triangleleft Пометить вершину как просмотренную

13: end if

14: k := k + 1;

15: end while

16: end for

Конец Функции

2. Алгоритм построения NOE-цепи

Рассмотренный в [14] 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 графе G называется самонепересекающимся если он гомеоморфен плоской замкнутой жордановой кривой без самопересечений, полученной из графа G с помощью применения O(| E(G) |) 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.

Функция Non-intersecting (G) (алгоритм 2) строит 4-регулярный граф G \widehat 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. 1(а) и 1(б)). Для выполнения указанных преобразований необходимо просмотреть функции vk(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

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

V (G) определена булева функция

Checked(v) = \Biggl\{ true, если вершина просмотрена;

false, в противном случае.

При выполнении инициализации (строки 1-3 в описании алгоритма 2) все вершины объявляют непросмотренными, т. е. Checked(v) = false для всех v \in V (G). Просмотр вершины v = v1(e), такой что Checked(v) = false состоит в выполнении процедуры Handle (e) (алгоритм 3). которая производит обработку данной вершины, заключающуюся в ее расщеплении в соответствии с рис. 1(а) и 1(б).

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.

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

Для 4-регулярного графа G \widehat с определенными для него рангами фиктивных ребер и введенными в его представление фиктивными гранями можно применить последовательно алгоритм CUT-POINT-SPLITTINGG \widehat построения графа G, \widetilde с расщепленными точками сочленения, и алгоритм AOE-TRAIL(G) \widetilde [14] построения AOE-цепи C \ast в графе G \widetilde . При построении цепи C \ast алгоритм AOE-TRAIL(G) \widetilde при наличии двух смежных непройденных ребер одного

Algorithm 3 Процедура Handle (e)

```
1: procedure Handle(e)
2: v := v1(e); \triangleleft Расщепляемая вершина
3: efirst := e; \triangleleft Сохранить первое рассматриваемое ребро
4: d := 0; \triangleleft Инициализация счетчика для степени вершины d
5: F := FaceNum() + 1; \triangleleft Определить номер для новой грани
6: repeat \triangleleft Проход 1: Определение степени вершины v
7: le := l1(e);
8: if (v1(le) \not= v) then REPLACE(le);
9: end if \triangleleft При необходимости поменять индексацию функций
10: e := le; d := d + 1; \triangleleft Учесть ребро при подсчете степени и перейти к следующему
11: until (e = efirst); \triangleleft Повторять, пока не будут просмотрены все ребра, инцидентные v
12: if (d > 4) then \triangleleft Если степень текущей вершины больше 4
13: e := efirst; \triangleleft Начать с первого рассматриваемого ребра
14: le := lk (e); \triangleleft Определить номер его левого соседа
15: enext := lk (le); \triangleleft Сохранить ребро, для следующей итерации
16: fl := new EDGE; fle := fl; efirst := e; \triangleleft Ввести фиктивное ребро, смежное le
17: repeat \triangleleft Расставить указатели для ребер
18: e := enext; le := lk (e); fr := fl;
19: f1(fl) := F ; f2(fl) := f2(e); \triangleleft Определить грани, смежные фиктивному ребру
20: rank(fl) := facerank(f2(fl)); \triangleleft Определить «ранг» фиктивного ребра
21: \triangleleft Функция facerank() вычисляет ранг грани в соответствии с определением
22: fl := new EDGE; enext := lk (le);
23: until (lk(le) = efirst);
24: end if
25: end procedure
```

ранга для гарантированного выполнения условия упорядоченного охватывания в первую очередь выбирает фиктивное ребро.

Процедура Absorb(\ast) заменяет в \ast 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) |).

Функция Non-Intersec ting 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 [14] решают задачу построения AOE-цепи C за время O(| E(G \ast) | \cdot log2 | V (G \ast) |), то и задача построения NOE-цепи в графе G решается за полиномиальное время O(| E(G) | 2).

Сказанное выше является доказательством 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. 2(а). 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, постоение NOE-цепи можно начать из любой вершины, смежной внешней грани. Пусть это будет расщепляемая вершина v2.

После применения процедуры Handle() получим граф, представленный на рис. 2(б). В полученном графе все вершины имеют степень 4. В графе на рис. 2(б) помимо расщепленной вершины v3 имеются и точки сочленения v4 и v6 рангов 3 и 2 соответственно, а также точка сочленения ранга 2 в расщепленной вершине v3, инцидентная ребрам e13, e4 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. 2(в). С помощью алгоритма AOE-Trail в полученном графе определим AOE-маршрут, в котором символом «*» обозначен переход по фиктивным ребрам

T \ast = v2e5v1e4v3 \ast \ast \ast e19v4e17v5e18v4e20v3e16v5e15v3 \ast

e3v8e12v6e11v9e14v3e6v9e7v7e10v6e9v7e8v8e2v3 \ast e13v1e1v2,

которому после стягивания расщепленных вершин соответствует NOE-маршрут

T \ast = v2e5v1e4v3e19v4e17v5e18v4e20v3e16v5e15v3

e3v8e12v6e11v9e14v3e6v9e7v7e10v6e9v7e8v8e2v3e13v1e1v2,

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 [14] для построения AOE-покрытия. В цепях полученного покрытия удалим все искусственные ребра, стягивая все расщепленные вершины. В результате получим NOE-покрытие.

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 3 , (2) отсутствуют самопересечения траектории резки.

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

а) Граф G б) 4-регулярный граф G \widehat

в) 4-регулярный граф G \widetilde 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