**УДК** 519.8.А

А. А. ПЕТУНИН, Е. Г. ПОЛИЩУК, С. С. УКОЛОВ

# НОВЫЙ АЛГОРИТМ ПОСТРОЕНИЯ КРАТЧАЙШЕГО ПУТИ ОБХОДА КОНЕЧНОГО МНОЖЕСТВА НЕПЕРЕСЕКАЮЩИХСЯ КОНТУРОВ НА ПЛОСКОСТИ

*А. А. Петунин, Е. Г. Полищук, С. С. Уколов.* **Новый алгоритм построения кратчайшего пути обхода конечного множества непересекающихся контуров на плоскости.**

**Аннотация.** Рассматривается проблема маршрутизации режущего инструмента машин термической резки с ЧПУ когда точки врезки расположены на границах деталей, ограниченных отрезками прямых и дугами окружностей с использованием техники непрерывной резки (CCP), то есть каждый контур вырезается целиком. Общая задача минимизации длины маршрута в этом случае сводится к минимизации длины холостого хода. Показано, что она эквивалентна поиску кратчайшей ломаной с вершинами, расположенными на контурах. Представлен алгоритм построения такой ломаной для заданного порядка обхода контуров, доставляющий локальный минимум а также предложены достаточные условия глобального минимума. Предложен эвристический алгоритм маршрутизации на основе метода переменных окрестностей (VNS). Рассмотрены результаты численных экспериментов в сравнении с точным решением задачи GTSP.

**Ключевые слова:** задача резки, непрерывная резка, оптимизация, достаточные условия, эвристика, GTSP, метод переменных окрестностей

**Введение.** В процессе разработки управляющих программ для машин термической резки листового материала с ЧПУ возникает несколько оптимизационных задач. Одна из них – минимизация длины холостого хода инструмента. Она в некоторых случаях может быть сведена к задаче поиска кратчайшей ломаной, вершины которой расположены на заданных плоских контурах, являющихся границами деталей. Расположение этих контуров на плоскости в свою очередь получено решением другой оптимизационной задачи – «раскроя». Обе упомянутые задачи являются NP-полными.

Задача минимизации длины холостого хода инструмента сама по себе является частным случаем общей оптимизационной проблемы маршрутизации инструмента. Её полное решение в общем случае не может быть получено в разумное время для задач типичных для современного промышленного производства (сотни / тысячи контуров), поэтому на практике применяются разнообразные эвристики, дающие решения приемлемого качества.

Для полного решения задачи маршрутизации режущего инструмента для машин листовой резки с ЧПУ для минимизации времени или стоимости резки требуется решить целый ряд задач. Их описание и классификация приведены подробно в [[1](#_bookmark17)—[3](#_bookmark18)], и схематически изображены на рис. [1](#_bookmark0).

* **Задача непрерывной резки** (Continuous Cutting Problem, CCP): каждый контур (ограничивающий одну из деталей) вырезается за один раз, одним движением инструмента, но резка может начаться в любой точке контура (и заканчивается в ней же)
* **Обобщённая задача коммивояжера** (Generalized Traveling Salesman Problem, GTSP): резка может начаться в одной из заранее заданных точек на контуре (количество таких точек конечно), после этого контур вырезается целиком
* **Задача резки с остановками** (Endpoint Cutting Problem, ECP): резка контура может начинаться только в заранее заданных точках на нём, но контур может вырезаться за несколько раз, частями
* **Сегментная задача непрерывной резки** (Segment Continuous Cutting Problem, SCCP): вводится понятие сегмента как обобщение понятия контура; сегмент может быть частью контура или объединением нескольких контуров и / или их частей. Каждый сегмент вырезается целиком, от начала до конца, таким образом
* **Обобщённая сегментная задача непрерывной резки** (Generalized Segment Continuous Cutting Problem, GSCCP): подобна сегментной задаче непрерывной резки (SCCP), но разбивка на сегменты не задана заранее и сама подлежит оптимизации
* **Задача прерывистой резки** (Intermittent Cutting Problem, ICP): наиболее общая формулировка задачи резки, встречающаяся в научной литературе, контуры могут вырезаться частями, в несколько подходов, начиная с произвольной точки.



Фиксированные точки врезки

TSP

Конечное множество точек врезки

ECP

GTSP

Континуум точек врезки

ICP

GSCCP

SCCP

CCP

Рис. 1. Классификация задач резки

На практике задача оптимизации маршрута режущего инструмента зачастую сводится к дискретной оптимизации за счёт выбора конечного множества возможных точек врезки на контурах деталей с некоторым заранее заданным шагом *ε*, то есть сводится к задаче ECP [[4](#_bookmark19)—[6](#_bookmark20)] и её частному случаю — GTSP [[7](#_bookmark21)—[10](#_bookmark22)]. Задача непрерывной резки CCP тоже может сводиться к GTSP, в этом случае суммарная ошибка длины холостого хода оценивается как , где *N* – количество контуров. Для достижения точности результата *δ*, таким образом необходимо выбирать малое , так что полное количество допустимых точек врезки растёт (как ) и полный перебор требует экспоненциального времени. Тем не менее, этот класс задач может успешно решаться, например, средствами динамического программирования (DP) а для небольших – даже точно (см. в частности [[11](#_bookmark23)]).

*·*

*≈*

*≈*

В данной работе рассматриваются вопросы поиска оптимального маршрута без применения дискретизации (то есть задача CCP), которые слабо освещены в открытых источниках, см. например [[12](#_bookmark24); [13](#_bookmark25)], где описаны некоторые эвристики.

**Технологические ограничения.** Тот факт, что полученный маршрут должен быть физически выполнен на конкретной машине термической резки с ЧПУ, накладывает на последний определённые технологические ограничения.

Так называемое «ограничение предшествования» (наиболее подробно описанное в литературе), вызвано тем, что после вырезания замкнутого контура, его содержимое более ничем не удерживается и может свободно вращаться, сдвигаться и даже падать. Поэтому внутренние контура деталей должны вырезаться до того, как будет завершена резка содержащих их внешних контуров. Аналогично и детали, размещённые в отверстиях других деталей, также должны вырезаться до завершения резки содержащих их контуров.

Наконец, технология резки в большинстве случаев диктует, что резак не может двигаться прямо по контуру детали, но с некоторым сдвигом (несколько миллиметров). Этот сдвиг может вычисляться как в ходе решения задачи маршрутизации, так и после – при генерации управляющей программы на основе полученного маршрута или даже непосредственно на станке в процессе резки. Более того, точки врезки (в которых начинается резка) как правило должны находиться на ещё большем расстоянии от контуров деталей во избежание повреждения последних. В данной работе эти вопросы, тем не менее, не рассматриваются, то есть строится маршрут, проходящий в точности по контурам деталей, и точки врезки (а равно и точки окончания резки и выключения инструмента) также ищутся прямо на контурах деталей.

* 1. **Задача непрерывной резки.** Рассмотрим Эвклидову плоскость и на ней фигуру (в большинстве практических случаев – прямоугольник), ограниченную замкнутым контуром. Это – модель листового материала, подлежащего резке. Пусть *N* попарно непересекающихся плоских контуров расположены внутри *B*, ограничивая *n* деталей {*A*1*, A*2*...An*}. Деталь может быть ограничена одним или несколькими контурами (одним внешним и несколькими отверстиями), так что в общем случае .

Контуры *Ci* могут быть произвольной формы, но мы будем рассматривать только состоящие из (конечного числа) отрезков прямых линий и дуг окружностей, так как именно такие геометрические примитивы поддерживаются программным обеспечением современных машин термической резки с ЧПУ. Частный случай, когда контура состоят только из отрезков прямых, сводится к одному из вариантов задачи обхода прямоугольников (Touring Polygon Problem, TPP), см. [[14](#_bookmark26)].

Далее, внутри *B* (как правило, на границе) выберем две точки и обозначим их *M*0, *MN*+1 (почти всегда *M*0=*MN*+1), которые будут использоваться как начало и конец маршрута резки.

Задача непрерывной резки (Continuous Cutting Problem, CCP) состоит в поиске:

1. *N* штук точек врезки
2. Последовательности обхода контуров *Ci*, то есть перестановки *N* элементов *I* = (*i*1*, i*2*, ...iN*)

Результатом решения задачи будет являться маршрут

*{M*0*, Mi*1*, Mi*2*, . . . MiN , MN*+1*}* (1)

Целевая функция в данном случае сильно упрощается по сравнению с общей задачей маршрутизации резки и сводится фактически к минимизации длины холостого хода:

(2)

где, для простоты записи мы полагаем .

Кроме того, мы потребуем, чтобы искомое решение задачи удовлетворяло описанному выше ограничению предшествования.

Хотя контуры *Ci* по условию не пересекаются, они могут быть вложены друг в друга:, где *C*˜*a* обозначает 2-мерную фигуру, ограниченную контуром *Ca* (в более традиционных обозначениях *Ca* = *∂C*˜*a*).

В общей задаче маршрутизации режущего инструмента это соответствует двум разным случаям (наличие отверстий в деталях с одной стороны и размещение меньших деталей в отверстиях больших), но в нашем случае оба этих варианта обрабатываются одинаково.

Если один контур расположен внутри другого, то внутренний должен быть вырезан (посещён) ранее, чем внешний: , в перестановке *I* = (*i*1*, i*2*, ...iN*). Таким образом, множество допустимых перестановок ограничено.

* 1. **Алгоритм CCP-Relax решения задачи непрерывной резки.** Предлагаемый алгоритм решения задачи непрерывной резки (см. [[15](#_bookmark27)]) состоит из нескольких шагов, что хорошо соответствует самой природе решаемой задачи.
     1. **Удаление «внешних» контуров.** Для автоматического соблюдения ограничения предшествования мы начинаем с удаления всех контуров, внутри которых есть вложенные контура, так, чтобы остались только:

В общем случае это приводит к уменьшению (в некоторых случаях– существенному) сложности задачи (с до некоторого ), что в свою очередь сокращает время счёта на втором и в особенности третьем шагах алгоритма.

* + 1. **Непрерывная оптимизация.** На этом этапе мы полагаем, что последовательность обхода контуров *I* = (*i*1*, i*2*, ...iN* ) задана (фиксирована) и ищем координаты точек врезки *Mi Ci* во все контура, минимизируя полную длину холостого хода [(2](#_bookmark1)). Для этого начальные позиции точек

*∈*

врезки выбираются произвольным образом (например, случайно) и затем положение одной (каждой) из точек *Mi* изменяется, а все остальные остаются неподвижны: . Большинство слагаемых в целевой функции [(2](#_bookmark1)) при этом постоянны, так что сама функция упрощается до

(a) На пересечении звена ломаной (b) С использованием *принципа Ферма*

Рис. 2. Оптимальное положение точки врезки

Несложный геометрический анализ показывает, что если точки *Mi-1* и *Mi*+1 расположены по разные стороны сегмента контура *Ci*, то оптимальное положение точки врезки *Mi* оказывается на пересечении с этим сегментом: *Mi* = *Mi* 1*Mi*+1 *Ci* (если, конечно, такое пересечение существует; в противном случае решением будет один из концов сегмента), см. рис. [2a](#_bookmark3).

*−*

Если же точки располагаются с одной стороны сегмента, решение легко находится при помощи *принципа Ферма*, или другими словами правила «угол падения равен углу отражения» (или опять на одном из концов сегмента), см. рис. [2b](#_bookmark4).

Общая схема этого шага оптимизации может быть записана таким образом:

1. Выбираем произвольные начальные позиции точек врезки *Mi ∈*

*Ci, ∀i*.

1. *i* 1*, N* находим оптимальное положение *Mi* как описано выше за константное время
2. Повторяем предыдущий шаг, до тех пор, пока длина холостого пути [(2](#_bookmark1)) не сойдётся (с некоторой наперёд заданной точностью *δ*)

На практике весь процесс хорошо сходится за время *O*(*N*) и поэтому многократно применяется в качестве подпрограммы на следующем шаге.

* + 1. **Дискретная оптимизация.** Наиболее вычислительно сложная задача заключается в поиске перестановки *I* = (*i*1*, i*2*, ...iN* ), минимизирующей полную длину холостого хода min. Фактически, это решение Задачи коммивояжёра (Traveling Salesman Problem, TSP), только длина пути вычисляется не аддитивно, а при помощи процесса непрерывной оптимизации, описанного на предыдущем шаге.

В данной работе для поиска такой перестановки применяется метод переменных окрестностей (Variable Neighborhood Search, VNS, см. [[16](#_bookmark28)]) по такой схеме:

1. Начальная перестановка *I* = (*i*1*, i*2*, ...iN* ) выбирается произвольным образом (например, случайно)

2. *k* = 1

1. while *k < kmax*:
   1. Выбираем перестановку *II ∈ Nk*(*I*) из окрестности *Nk*(*I*), доставляющую минимум *L*(*II*)
   2. if *L*(*II*) *< L*(*I*):
      1. *I ← II*

3.2.2. *k ←* 1

* 1. else

3.3.1. *k ← k* + 1

1. Завершение работы.

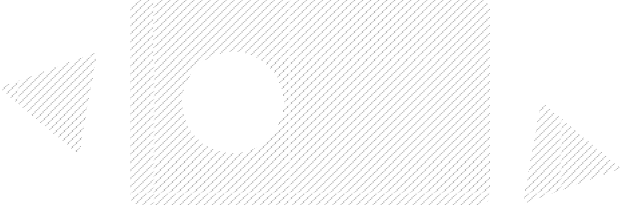
На шаге 3.1 многократно применяется непрерывная оптимизация из предыдущего этапа:

Окрестности различного размера конструируются разнообразными способами, например:

* Все возможные парные перестановки (то есть, окрестности размера 1 в смысле транспозиционной метрики)
* Циклические перестановки 3 контуров. Поскольку всего таких перестановок получается *O*(*N* 3), выбираются только те из них, в которых задействованные контуры расположены в исходной перестановке *I* = (*i*1*, i*2*, ...iN* ) не далее, чем на предопределённом расстоянии друг от друга; это предопределённое расстояние является параметром алгоритма
* Подобным же образом, выбираются циклические перестановки 4 контуров, лежащих не далее заданного расстояния друг от друга в исходной перестановке *I* = (*i*1*, i*2*, ...iN* )
* Выбирается последовательный блок контуров произвольной длины и к нему применяется циклический сдвиг
* Все контуры в последовательном блоке контуров (произвольной длины) переставляются в обратном порядке
* Перестановка двух последовательных (но не смежных) блоков контуров
* Циклическая перестановка нескольких последовательно расположенных последовательных блоков контуров (произвольной, но одинаковой длины)
* И ещё порядка десяти других способов генерации «близких» к исходной перестановок

Если размер некоторой окрестности , получаемой одним из способов, оказывается слишком большим (что приводит к увеличению времени счёта), он легко может быть ограничен при помощи введения дополнительного параметра алгоритма, подобно тому, как это сделано для тройных и четверных циклических перестановок.

Кроме того, сам метод переменных окрестностей допускает несколько вариантов применения, например замена полного перебора (на шаге 3.1) на «первый подходящий» или метод Монте-Карло, но их влияние на скорость и качество получаемого решения задачи непрерывной резки требует дальнейшего исследования.



*M*5

*M*3

*M*+ *M*

4

*M*2

*M*1

Рис. 3. Добавление точек врезки во «внешние» контура M+

* + 1. **Восстановление удалённых контуров.** На предыдущем шаге мы получили последовательность обхода контуров и точки врезки для каждого из них, но только для тех, которые не содержат других контуров внутри себя. Теперь необходимо дополнить эту последовательность оставшимися контурами (удалёнными на первом шаге) и точками врезки для них, причём таким образом, чтобы ограничение предшествования оказалось соблюдённым.

Заметим, что маршрут, полученный к этому моменту

*Route* = *{M*0*, M*1*, M*2*, . . . MN1 , MN1* +1*}* (3)

обязательно пересекает все исходные контуры *Ci*, потому что для контуров, сохранённых на первом шаге, он их явно посещает по построению шагов 2 и 3, а для остальных контуров – ввиду того, что каждый из них включает в себя один из посещённых контуров, а начальная и конечная точка *M*0 и *MN*+1 всегда выбираются снаружи всех контуров *Ci*.

Таким образом, для каждого «внешнего» контура *Ci*, который пока не включён в маршрут резки, мы находим все точки пересечения с полученным маршрутом [(3](#_bookmark6)), и если таких точек несколько (что как правило и бывает), выбираем из них самую последнюю, то есть посещаемую маршрутом позже всех других, см. рис. [3](#_bookmark5). Сам контур *Ci* вставляется в перестановку в месте, соответствующем выбранной точке врезки.

После добавления таким образом всех «внешних» контуров и соответствующих им точек врезки, мы получаем уже полный маршрут, который посещает все исходные контуры, причём внутренние контуры посещаются строго раньше содержащих их внешних. Полная длина маршрута при этой операции, очевидно, не меняется.

Легко понять, что получаемый таким образом полный маршрут является оптимальным решением исходной задачи непрерывной резки. Действительно, если бы существовал более короткий маршрут, посещающий все контура, из него можно было бы просто удалить точки врезки, лежашие на «внешних» контурах, получив тем самым маршрут, обходящий «внутренние» контура и имеющий ту же, то есть меньшую длину. Таким образом, существует лучшее решение для задачи обхода части контуров без учёта ограничений предшествования, но это по предположению невозможно.

Таким образом, мы строго выполняем ограничение предшествования, тратя на это линейное время .

На этом выполнение предложенного эвристического алгоритма решения задачи непрерывной резки завершается.

1. **Условия оптимальности решения непрерывной задачи.** С практической точки зрения, вышеописанный алгоритм оказывается вполне работоспособным и даёт хорошие результаты, как в смысле времени работы, так и качества получаемых решений. Однако, это чисто эмпирический результат, так что было бы интересно получить теоретические оценки качества работы данного алгоритма.

Наиболее важным и одновременно наиболее сложным является, конечно, третий этап – дискретная оптимизация (фактически решение задачи коммивояжёра), как с теоретической, так и с практической точки зрения. В данной же работе исследуется второй шаг алгоритма – непрерывная оптимизация. Оказывается возможным сформулировать некоторые утверждения относительно получаемых в её ходе решений.

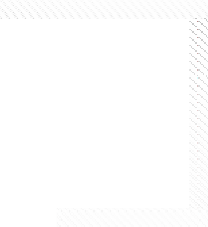
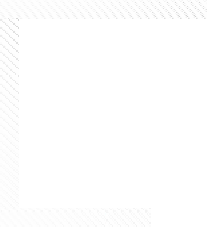
На рис. [4](#_bookmark7) показан пример маршрута резки, который не являясь кратчайшим, тем не менее неё может быть приведён к таковому никакими индивидуальными сдвигами точек врезки. Таким образом, возникает вопрос, при каких условиях предлагаемый алгоритм гарантирует получение действительно оптимального маршрута, то есть, другими словами, глобального минимума оптимизационной задачи.

Итак, рассмотрим следующую задачу: пусть контуры *Ci* на плоскости состоят только из отрезков прямых линий, они не пересекаются и не вложены друг в друга. Порядок обхода контуров заранее задан. Требуется найти кратчайшую ломаную линию, чьи вершины лежат на заданных контурах (в указанном порядке).

Мы начинаем с (произвольной) ломаной линии *L*1. Далее для каждого контура *Ci* мы повторяем следующую операцию: сдвигаем вершину ломаной *Mi Ci* в такое положение, которое минимизирует полную длину ломаной, при этом остальные вершины *Mj* () неподвижны. Это сводится к несложной геометрической задаче нахождения точки, для которой сумма расстояний до двух других фиксированных точек минимальна.

*/*

*∈*



Локальный минимум

*C*1

*C*2

Глобальный минимум

Рис. 4. Два маршрута резки, доставляющие локальный и глобальный минимум

В процессе таких пошаговых сдвигов мы получаем последовательность ломаных линий *Lk*, причём последовательность длин этих ломаных линий по построению монотонно убывает. Обозначим (точная нижняя граница длин ломаных линий) и пусть *L\** – ломаная линия длины , то есть предельная точка последовательности ломаных линий *Lk*. В качестве метрики на пространстве ломаных линий можно использовать сумму расстояний между вершинами с одинаковыми номерами.

В реальных примерах стабилизация последовательности ломаных происходит буквально в течение нескольких итераций (не более 10) и ломаная *L\** действительно получается во всех численных экспериментах По построению, длина ломаной *L\** не может быть уменьшена никаким сдвигом одной из её вершин (в рамках содержащего её контура), также как и сдвигом любого количества её вершин, не являющихся соседними.

*∗*

# Локальный минимум.

*П р е д л о ж е н и е 1. Сдвиг нескольких соседних вершин ломаной L таким образом, что сдвигаемые вершины остаются на тех же самых сегментах контуров, не приводит к уменьшению полной длины ломаной*

*∗*

*Д о к а з а т е л ь с т в о.* Рассмотрим сдвиг *двух* соседних вершин. Обозначим четыре последовательных вершины ломаной *L* за *Mi* 1*, Mi, Mi*+1*, Mi*+2.

*−*

*∗*

Пусть точки *Mi Si, Mi*+1 *Si*+1 лежат на прямолинейных

*∈ ∈*

сегментах соответствующих контуров *Si Ci, Si*+1 *Ci*+1.

*⊂ ⊂*

Докажем, что:

*∀MiI ∈ Si, MiI*+1 *∈ Si*+1 : *|Mi−*1*MiIMiI*+1*Mi*+2*|* � *|Mi−*1*MiMi*+1*Mi*+2*|*

Для произвольной вершины *MiI ∈ Si*: *|Mi−*1*MiIMi*+1*Mi*+2*|* минимально, когда *MiI* = *Mi*, и аналогично для произвольной вершины

*MiI*+1 *Si*+1: *Mi* 1*MiMiI*+1*Mi*+2 минимально, когда *MiI*+1 = *Mi*+1.

*∈ | |−*

Предположим, что

*∃MiI ∈ Si, ∃MiI*+1 *∈ Si*+1 : *|Mi−*1*MiIMiI*+1*Mi*+2*| < |Mi−*1*MiMi*+1*Mi*+2*|*

Очевидно, что *MiI /*= *Mi, MiI*+1 =*/ Mi*+1.

Введём обозначение *Mi*(*s*) = *Mi* + *s · −M−i−M→iI*, *Mi*+1(*t*) = *Mi*+1 +

*−−−−−****−*** *→*

*t Mi*+1*MiI*+1 (*s, t* [0*,* 1]), *f* (*s, t*) = *Mi* 1*Mi*(*s*)*Mi*+1(*t*)*Mi*+2 .

* *∈ | |−*

Но положения вершин *Mi* and *Mi*+1 выбраны так, что:

(4)

Если, например, , то это может быть только если точки *Mi-1, Mi, Mi*+1 лежат на одной прямой (когда *Mi-1* и *Mi+1* по одну сторону от *Si*; в противном случае заменяем *Mi-1* на её отражение относительно *Si*). Таким образом, если одновременно

*− −*

*−*

то значит все четыре точки *Mi-1, Mi, Mi*+1*, Mi*+2 лежат на одной прямой и проходящая через них ломаная фактически является отрезком прямой и заведомо кратчайшая, не может быть сделана короче.

*−*

Рассмотрим теперь случай, когда хотя бы одна из производных в (4) не равна нулю. Обозначим .

Это значит, что *τ∗* [0*,* 1]: *ϕ*(*τ∗*) *> ϕ*(0). Но по нашему предположению *ϕ*(1) *< ϕ*(0), то есть *ϕ*(0) *< ϕ*(*τ∗*) *> ϕ*(1).

*∃ ∈*

Теперь заметим, что *ϕ*(*t*) представляет собой сумму четырёх слага-

* *· ·*

емых вида (*a* + *b t*)2 + (*c* + *d t*)2, и каждое из этих слагаемых имеет положительную вторую производную, так что и *d*2*ϕ*(*t*)*/dt*2 � 0, а значит функция *ϕ*(*t*) является выпуклой на [0*,* 1] и не может принимать во

внутренней точке интервала значения большие, чем на его концах.

Значит, наше предположение невозможно.

Мы рассмотрели сдвиг *двух* соседних вершин ломаной. Для большего числа соседних вершин доказательство аналогично, только более громоздко.

Окончательно, мы доказали, что ломаная линия *L* представляет собой **локальный** минимум.

* 1. **Глобальный минимум.** Перейдём к *достаточному* условию того, что ломаная *L* является глобальным минимумом.

*∗*

Пусть *Mi Ci* – вершина ломаной *L* . Соседние вершины *Mi* 1 и *Mi*+1 находятся вне *Ci*, поскольку по условию задачи мы исключили вло- женные контуры По построению *L* : *MiI Ci* : *Mi* 1*Mi* + *MiMi*+1 �

*∈ ∗ −*

*∀ ∈ | | | |∗ −*

*Mi* 1*MI* + *MIMi*+1 .

*| | | |− i i*

**Условие 1.** Пусть выполняется **одно** из следующих условий:

* 1. Сегмент *Mi−*1*Mi*+1 пересекает контур *Ci*, то есть *Mi ∈ Mi−*1*Mi*+1
  2. Касательная в точке *Mi* к эллипсу с фокусами *Mi* 1 и *Mi*+1 и проходящему через *Mi*, разделяет эллипс и контур *Ci*.

*−*

*П р е д л о ж е н и е 2. Если* ***Условие 1*** *выполняется для (всех кон- туров) L , то при сдвиге нескольких соседних вершин ломаной L в рамках содержащих их контуров, полная длина ломаной не уменьшается, то есть ломаная L∗ представляет собой глобальный минимум.*

*∗ ∗*

*Д о к а з а т е л ь с т в о.* Используя те же обозначения, рассмотрим

четыре соседних вершины *Mi−*1*, Mi, Mi*+1*, Mi*+2 *∈ L∗*. *Mi ∈ Ci, Mi*+1 *∈*

*C* .

*i*+1

Предположим, что

*∃MiI ∈ Ci, ∃MiI*+1 *∈ Ci*+1 : *|Mi−*1*MiIMiI*+1*Mi*+2*| < |Mi−*1*MiMi*+1*Mi*+2*|*

Снова, *MiI /*= *Mi, MiI*+1 */*= *Mi*+1.

Обозначим *Mi*(*s*) = *Mi*+*s·−M−i−M→iI*, *Mi*+1(*t*) = *Mi*+1+*t·M****−*** *i−*+*−*1*−M****−****iI*+*→*1

(*s, t ∈* [0*,* 1]), *f* (*s, t*) = *|Mi−*1*Mi*(*s*)*Mi*+1(*t*)*Mi*+2*|*.

Но теперь **Условие 1** гарантирует, что *f* (*s,* 0) � *f* (0*,* 0) для *s ∈*

[0*,* 1], то есть снова

*∂f* (*s, t*)

*∂s* 1

(0*,*0)

*∂f* (*s, t*)

*∂t* 1

� 0*,*

(0*,*0)

� 0 (5)

Поэтому остальная часть предыдущего доказательства повторяется отсюда без изменений.

Предположим, что кроме *L* существует другая ломаная, также являющаяся глобальным минимумом. Тогда из доказательства следует, что они представляют собой одну и ту же линию (как множество точек) и отличаются только положением вершин, то есть тем, какие именно пересечения с контурами выбраны в качестве вершин ломаных линий.

*∗*

**Условие 1** легко проверяется программно, однако его можно ещё упростить с тем, чтобы в большинстве практических случаев его можно было проверить просто визуально, буквально «на глаз».

**Условие 2.** Выполняется **любое** из условий:

1. Сегмент *Mi−*1*Mi*+1 пересекает контур *Ci*: *Mi ∈ Mi−*1*Mi*+1
2. Если вершина *Mi* является внутренней точкой одного из отрезков контура *Ci* и при этом весь контур расположен по одну сторону линии, проходящей через этот отрезок (это и есть касательная,

которую использует **Условие 1**; иначе существовала бы лучшая вершина *MiI ∈ Ci*).

1. Если вершина *Mi* является также вершиной контура *Ci* (принадлежит сразу двум его отрезкам) и при этом весь контур находится внутри угла, образованного лучами, идущими из вершины *Mi* вдоль этих двух отрезков
2. Если контур *Ci* ограничивает собой выпуклый многоугольник *C*˜*i*

**4. Новый подход к задаче прерывистой резки (ICP).** Задача прерывистой резки – самая общая и сложная из всех классов задач резки. Её изучение и решение представляется полезным как с теоретической точки зрения, так и для решения практических задач современного производства.

Кроме стандартной техники резки (которая фактически приводит к задаче непрерывной резки), используются и другие техники резки, например, «мульти-сегментная» и «мульти-контурная» резка. В первом случае один контур может вырезаться в несколько подходов, по частям. Во втором наоборот, несколько контуров вырезаются одним движением резака, см. рис. [5](#_bookmark9).

Для того, чтобы можно было описать эти техники резки в используемой математической модели, мы вводим новые понятия:

*Сегмент резки S* = *−M−M−→∗* – это траектория инструмента от точки

врезки *M* до соответствующей точки выключения инструмента *M∗*.

В стандартной технике сегмент резки содержит ровно один контур

детали, но в общем случае это уже не так. При мульт-сегментной резки сегмент содержит только часть контура, а при мульти-контурной – несколько контуров (или их частей). Рис. [5](#_bookmark9), таким образом, может представлять не только пример мульти-контурной резки, но также и один единственный сегмент в составе некоторой большей задачи резки.



Рис. 5. Пример составного сегмента резка для шести деталей / контуров

Поскольку сегмент резки по определению содержит информацию о направлении резки, нам потребуется ещё одно понятие:

*Базовый сегмент BS* – это часть сегмента резки *S* = *−M−M−→∗*, без начальной и конечной части (вспомогательных движений резака, когда он движется от точки врезки к собственно контуру и отходит от вырезанного контура в конце). Базовый сегмент не содержит информации о направлении резки и является чисто геометрическим объектом.

При помощи понятия базового сегмента оказывается возможным сформулировать обобщение задачи непрерывной резки:

*Сегментная задача непрерывной резки* (Segment Continuous Cutting Problem, SCCP) – это задача резки фиксированного набора базовых сегментов резки *SCCP* = *BSi* .

{ }

Описанный в разделе [2](#_bookmark2) алгоритм CCP-Relax естественным образом обобщается на решение задачи SCCP.

Заметим теперь, что если задана раскройная карта, то есть расположение всех деталей и их граничных контуров, из неё может быть сгенерирован целый *ансамбль* наборов базовых сегментов путём разбие- ния контуров на сегменты и объединения нескольких сегментов в один. См., например, рис. [6](#_bookmark10), где мульти-контурные сегменты выделены чёрным цветом. Это позволяет ещё больше обобщить задачу и сформулировать новый класс:

*Обобщённая задача сегментной резки* (Generalized Segment Continuous Cutting Problem, GSCCP) – *ансамбль* нескольких задач сегментной резки (SCCP) для одного раскройного плана: *GSCCP* = *{SCCPi}*.

* 1. Стандартная резка, 45 сегментов (b) Мульти-контурная резка, 39 сегментов

Рис. 6. *Ансамбль* задач сегментной резки

Введение нового класса задач резки GSCCP существенно расширяет существующую классификацию задач маршрутизации инструмента машин термической резки с ЧПУ. Фактически, задачи SCCP и GSCCP являются частными случаями задачи ICP, содержащими конечное множество базовых сегментов резки: *CCP SCCP GSCCP ICP* .

*⊂ ⊂ ⊂*

**Общая схема решения задачи GSCCP.** Предполагая ансамбль

*{ }* { } *∈ ∈*

*SCCPi* наборов базовых сегментов *SCCPi* = *BSj* , *i* 1*, T, j*

1*, Ki* фиксированным, используем следующую общую схему решения задачи GSCCP:

* Каждая из задач *SCCPi* решается независимо любым существующим алгоритмом, например:
  + 1. *CCP-Relax*, описанная в разделе [2](#_bookmark2) эвристика.
    2. *DP-GTSP*, точный алгоритм динамического программирования для задач небольшого размера, см. [[11](#_bookmark23)].
    3. *Greedy-GTSP*, итеративный жадный эвристический алгоритм, см. [[17](#_bookmark29)].

Для алгоритмов дискретной оптимизации контуры деталей предварительно преобразуются в конечные множества допустимых точек врезки, как показано на рис. [7](#_bookmark11).

* Лучшее решение выбирается на основе значения целевой функции [(2](#_bookmark1)).



Рис. 7. Задача GTSP, полученная по задаче (S)CCP на рис. [6](#_bookmark10), 425 точек врезки

Рис. [8](#_bookmark12) показывает решения задач SCCP с рис. [6](#_bookmark10), полученные при помощи алгоритма *CCP-Relax*. Легко видеть, что маршруты действительно различаются. Более того, с практической точки зрения, различие может быть ещё более значительным, так как маршруты отличаются не только геометрически, но и количеством точек врезки, а эта операция сравнительно дорогая для современных машин резки с ЧПУ, как в смысле стоимости, так и времени работы.

1. **Численные эксперименты.** Оценка качества решений, получаемых алгоритмом CCP-Relax проводилась на нескольких раскройных планах, содержащих реальные детали. В качестве базы сравнения был взят точный алгоритм решения задачи GTSP на основе динамического программирования (см. [[11](#_bookmark23)]), гарантирующий нахождение наилучшего решения для задач с небольшим количеством контуров, а также специализированная версия эвристики GNLS [[18](#_bookmark30)].

На рис. [9](#_bookmark13) представлено точное решение, при этом видны допустимые точки врезки, полученные дискретизацией исходных контуров деталей. На рис. [10](#_bookmark14) представлено решение задачи CCP, полученное алгоритмом CCP-Relax.

Можно отметить, что полученные маршруты практически идентичны. Небольшие отклонения вызваны процессом предварительной дискретизации (при переводе задачи CCP в GTSP). Вследствие этого алгоритм CCP-Relax формирует прямые сегменты холостого хода инструмента, а в случае задачи GTSP получаются слегка ломаные линии, что слегка увеличивает результирующую длину маршрута. Более точно это отражено в табл. [1](#_bookmark15) для нескольких раскройных планов.

* 1. Стандартная резка (b) Мульти-контурная резка

Рис. 8. Решение задачи GSCCP с рис. [6](#_bookmark10)



Рис. 9. Точное решение задачи GTSP, задание № 464



Рис. 10. Решение задачи CCP, задание № 464

Таблица 1. Сравнение решений задач CCP и GTSP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Задание | #229 | #464 | #3211 | #20205 |
| Деталей | 11 | 14 | 17 | 115 |
| Контуров | 12 | 21 | 22 | 198 |
| Точек GTSP | 491 | 429 | 493 | 3917 |
| *LGTSP* , m | 7.729 | 4.743 | 4.557 | 26.098 |
| *LCCP* , m | 7.727 | 4.706 | 4.536 | 25.987 |

На рис. [11](#_bookmark16) показано решение задачи CCP (алгоритмом CCP-Relax) для раскройного плана на 198 контуров, когда точное решение не известно. Вообще, для задач такого и большего размера (а именно они представляют практический интерес), оценка оптимальности решения представляет собой довольно большие сложности. Тем не менее, сравнение с решениями хорошо исследованной задачи GTSP вполне применимо для оценки качества решения. Как известно, GTSP является NP-сложной даже на Эвклидовой плоскости [[19](#_bookmark31)].

Хотя из общих соображений понятно, что чем больше ограничений (имеются в виду ограничения предшествования) наложено на задачу GTSP, тем проще её решить, вопрос влияния уровня вложенности деталей на границы сложности решений ещё не достаточно исследовался и заслуживает внимания. По этому поводу хочется отметить работы [[20](#_bookmark32); [21](#_bookmark33)]. Для двух видов ограничений предшествования теоретически доказана полиномиальная временная сложность задачи GTSP. Один тип описан Е. Баласом в [[22](#_bookmark34)] для классической задачи TSP. Эффективный точный алгоритм для задачи GTSP с такими ограничениями предшествования предложен в работах [[23](#_bookmark35); [24](#_bookmark36)]. Маршруты, содержащие ограничения второго вида, называются квази- и псевдо-пирамидальными. Эффективные параметрические алгоритмы решения задачи GTSP с ограничениями такого вида предложены в [[25](#_bookmark37); [26](#_bookmark38)].

Ввиду вышеизложенного, можно заключить, что алгоритмический анализ даже задачи GTSP в приложении к маршрутизации резки всё ещё является малоисследованной областью. В частности, отсутствие эффективных моделей MILP (Mixed Integer Linear Program) для задачи GTSP делает невозможным использование современных решателей, таких как Gurobi [[27](#_bookmark39)] для построения верхних и нижних границ и оценки эвристических решений. Это также представляется перспективным направлением исследований.

# Заключение.

1. Задача минимизации длины холостого хода инструмента машин термической резки с ЧПУ для задачи непрерывной резки сведена к аналогичной задаче, не содержащей ограничений предшествования, сокращая количество контуров и время работы алгоритма
2. Предложен эвристический алгоритм CCP-Relax поиска положений точек врезки в задаче непрерывной резки
3. Доказано, что алгоритм CCP-Relax при заданном порядке обхода контуров доставляет локальный минимум длины холостого хода
4. Сформулировано несколько легко проверяемых условий, при которых решение, полученное алгоритмом CCP-Relax, является глобаль- ным минимумом.
5. Алгоритм CCP-Relax может применяться также для решения более широкого класса задач резки – SCCP (Сегментная резка) и GSCCP (Обобщённая сегментная резка), что в свою очередь является перспективным подходом к решению общей задачи ICP.

Ближайшим направлением дальнейших исследований является обобщение алгоритма CCP-Relax на более практический случай, когда точки врезки лежат вне контуров деталей в соответствии с технологическими требованиями современного оборудования с ЧПУ.

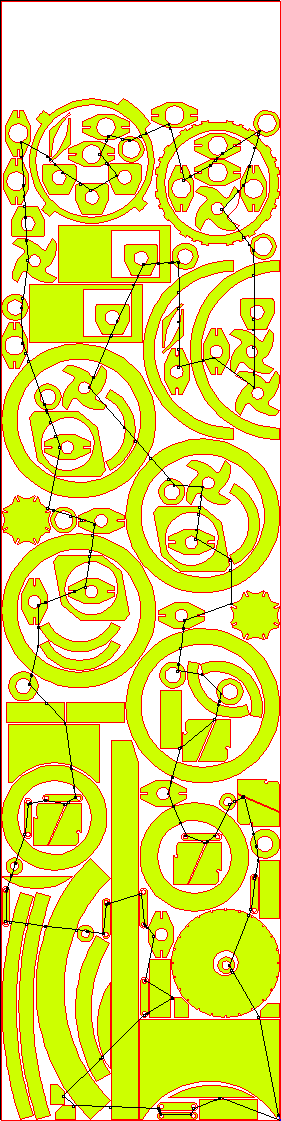


Рис. 11. Пример решения задачи CCP большого размера, задание #20205

# Литература.

1. *Hoeft J.*, *Palekar U. S.* Heuristics for the plate-cutting traveling salesman problem // IIE Transactions. — 1997. — т. 29, № 9. — с. 719—731. — ISSN 1573-9724. — DOI: 10.1023/A:1018582320737.
2. *Dewil R.*, *Vansteenwegen P.*, *Cattrysse D.* A review of cutting path algorithms for laser cutters // International Journal of Advanced Manufacturing Technology. — 2016. — т. 87, № 5. — с. 1865— 1884. — ISSN 1433-3015. — DOI: [10.1007/s00170-016-8609-1](https://doi.org/10.1007/s00170-016-8609-1).
3. *Petunin A. A.*, *Stylios C.* Optimization Models of Tool Path Problem for CNC Sheet Metal Cutting Machines // IFAC-PapersOnLine. — 2016. — т. 49, № 12. — с. 23—28.
4. *Dewil R.*, *Vansteenwegen P.*, *Cattrysse D.* Construction heuristics for generating tool paths for laser cutters // International Journal of Production Research. — 2014. — т. 52, № 20. — с. 5965—5984.
5. *Sherif S. U.*, *Jawahar N.*, *Balamurali M.* Sequential optimization approach for nesting and cutting sequence in laser cutting // Journal of Manufacturing Systems. — 2014. — т. 33, № 4. — с. 624—638.
6. Generation of cutter paths for hard material in wire EDM / S. Imahori [и др.] // Journal of Materials Processing Technology. — 2008. — т. 206,

№ 1. — с. 453—461. — ISSN 0924-0136. — DOI: [10.1016/j.jmatprotec.](https://doi.org/10.1016/j.jmatprotec.2007.12.039) [2007.12.039](https://doi.org/10.1016/j.jmatprotec.2007.12.039).

1. *Chentsov A. G.*, *Chentsov A. A.* A Discrete–Continuous Routing Problem with Precedence Constraints // Proceedings of the Steklov Institute of Mathematics. — 2018. — т. 300, № 1. — с. 56—71. — ISSN 1531- 8605. — DOI: [10.1134/S0081543818020074](https://doi.org/10.1134/S0081543818020074).
2. Elements of dynamic programming in local improvement constructions for heuristic solutions of routing problems with constraints / A. A. Petunin [и др.] // Automation and Remote Control. — 2017. — т. 78, № 4. — с. 666—681. — ISSN 1608-3032. — DOI: [10 . 1134 / S0005117917040087](https://doi.org/10.1134/S0005117917040087).
3. *Ye J.*, *Chen Z. G.* An Optimized Algorithm of Numerical Cutting-Path Control in Garment Manufacturing // Advanced Materials Research. — 2013. — т. 796. — с. 454—457. — ISSN 1662-8985. — DOI: [10.4028/ www.scientific.net/AMR.796.454](https://doi.org/10.4028/www.scientific.net/AMR.796.454).
4. *Yu W.*, *Lu L.* A route planning strategy for the automatic garment cutter based on genetic algorithm // 2014 IEEE Congress on Evolutionary Computation (CEC). — 2014. — с. 379—386. — ISSN 1941-0026. — DOI: [10.1109/CEC.2014.6900425](https://doi.org/10.1109/CEC.2014.6900425).
5. Model of megalopolises in the tool path optimisation for CNC plate cutting machines / A. G. Chentsov [и др.] // International Journal of Production Research. — 2018. — т. 56, № 14. — с. 4819—4830. — ISSN 0020-7543. — DOI: [10.1080/00207543.2017.1421784](https://doi.org/10.1080/00207543.2017.1421784).
6. *Arkin E. M.*, *Hassin R.* Approximation algorithms for the geometric covering salesman problem // Discrete Applied Mathematics. — 1994. — т. 55, № 3. — с. 197—218. — ISSN 0166-218X. — DOI: [10.1016/0166- 218X(94)90008-6](https://doi.org/10.1016/0166-218X(94)90008-6).
7. *Vicencio K.*, *Davis B.*, *Gentilini I.* Multi-goal path planning based on the generalized Traveling Salesman Problem with neighborhoods // 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2014. — с. 2985—2990. — ISSN 2153-0866. — DOI: [10.1109/IROS.2014.6942974](https://doi.org/10.1109/IROS.2014.6942974).
8. Touring a sequence of polygons / M. Dror [и др.] // Proceedings of the thirty-fifth annual ACM symposium on Theory of computing. — ACM. 2003. — с. 473—482.
9. *Petunin A. A.*, *Polishchuk E. G.*, *Ukolov S. S.* On the new Algorithm for Solving Continuous Cutting Problem // IFAC-PapersOnLine. — 2019. — т. 52, № 13. — с. 2320—2325. — ISSN 2405-8963. — DOI: [10.1016/j.ifacol.2019.11.552](https://doi.org/10.1016/j.ifacol.2019.11.552).
10. *Hansen P.*, *Mladenovic´ N.*, *Moreno Pe´rez J. A.* Variable neighbourhood search: methods and applications // Annals of Operations Research. — 2010. — т. 175, № 1. — с. 367—407. — ISSN 1572-9338. — DOI: [10.1007/s10479-009-0657-6](https://doi.org/10.1007/s10479-009-0657-6).
11. *Petunin A. A.*, *Chentsov A. G.*, *Chentsov P. A.* About routing in the sheet cutting // Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software. — 2017. — т. 10,

№ 3. — с. 25—39. — ISSN 2071-0216. — DOI: [10.14529/mmp170303](https://doi.org/10.14529/mmp170303).

1. *Smith S. L.*, *Imeson F.* GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem // Computers & Operations Research. — 2017. — т. 87. — с. 1—19. — ISSN 0305-0548. — DOI: [10.1016/j.cor.2017.05.010](https://doi.org/10.1016/j.cor.2017.05.010).
2. *Papadimitriou C. H.* Euclidean TSP is NP-complete // Theoretical Computer Science. — 1977. — т. 4. — с. 237—244.
3. *Chentsov A. G.*, *Grigoryev A. M.* A Scheme of Independent Calculations in a Precedence Constrained Routing Problem // SpringerLink. — 2016. — с. 121—135. — DOI: [10.1007/978-3-319-44914-2\\_10](https://doi.org/10.1007/978-3-319-44914-2\_10).
4. *Saliy Y. V.* Influence of predestination conditions on the computational complexity of solution of route problems by the dynamic programming method // Bulletin of the Udmurt University. Maths. Mechanics. Computer Science. — 2014. — № 1. — с. 76—86. — ISSN 1994-9197.
5. *Balas E.* New classes of efficiently solvable generalized Traveling Salesman Problems // Annals of Operations Research. — 1999. — т. 86. — с. 529—558. — ISSN 1572-9338. — DOI: 10.1023/A:1018939709890.
6. *Chentsov A. G.*, *Khachai M. Y.*, *Khachai D. M.* An exact algorithm with linear complexity for a problem of visiting megalopolises // Proceedings of the Steklov Institute of Mathematics. — 2016. — т. 295, № 1. — с. 38—46. — ISSN 1531-8605. — DOI: [10.1134/S0081543816090054](https://doi.org/10.1134/S0081543816090054).
7. *Chentsov A.*, *Khachay M.*, *Khachay D.* Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem // IFAC-PapersOnLine. — 2016. — т. 49, № 12. — с. 651— 655. — ISSN 2405-8963. — DOI: [10.1016/j.ifacol.2016.07.767](https://doi.org/10.1016/j.ifacol.2016.07.767).
8. *Khachai M. Y.*, *Neznakhina E. D.* Approximation Schemes for the Generalized Traveling Salesman Problem // Proceedings of the Steklov Institute of Mathematics. — 2017. — т. 299, № 1. — с. 97—105. — ISSN 1531-8605. — DOI: [10.1134/S0081543817090127](https://doi.org/10.1134/S0081543817090127).
9. *Khachay M.*, *Neznakhina K.* Complexity and approximability of the Euclidean generalized traveling salesman problem in grid clusters // Annals of Mathematics and Artificial Intelligence. — 2020. — т. 88,

№ 1. — с. 53—69. — ISSN 1573-7470. — DOI: [10.1007/s10472-019-](https://doi.org/10.1007/s10472-019-09626-w) [09626-w](https://doi.org/10.1007/s10472-019-09626-w).

1. *Gurobi Optimization*. Gurobi optimizer reference manual. — 2020. — URL: [http://www.gurobi.com](http://www.gurobi.com/).

**Петунин Александр Александрович** — д-р техн. наук., доцент; заведующий лабораторией оптимальных методов раскроя промышленных материалов и оптимальных маршрутных технологий УрФУ. Число научных публикаций — . [a.a.petunin@urfu.ru,](mailto:a.a.petunin@urfu.ru) https://urfu.ru; Уральский Федеральный университет, 620002, Екатеринбург, Мира, 19.

**Полищук Ефим Григорьевич** — канд. физ-мат. наук, доцент; старший научный сотрудник лаборатории оптимальных методов раскроя промышленных материалов и оптимальных маршрутных технологий УрФУ. Число научных публикаций — . [e.g.polishchuk@urfu.ru,](mailto:e.g.polishchuk@urfu.ru) https://urfu.ru; Уральский Федеральный университет, 620002, Екатеринбург, Мира, 19.

**Уколов Станислав Сергеевич** — младший научный сотрудник лаборатории оптимальных методов раскроя промышленных материалов и оптимальных маршрутных технологий УрФУ. Область научных интересов: маршрутизация инструмента машин термической резки с ЧПУ. Число научных публикаций — 5. [s.s.ukolov@urfu.ru,](mailto:s.s.ukolov@urfu.ru) https://urfu.ru; Уральский Федеральный университет, 620002, Екатеринбург, Мира, 19.

**Поддержка исследований.** Работа выполнена при финансовой поддержке РФФИ (грант 20-08-00873).