# Один способ нахождения кратчайшей ломаной с вершинами на контурах в плоскости

*Рассматривается эвристический алгоритм решения задачи непрерывной резки…*

Ключевые слова: маршрутизация, непрерывная резка, непрерывно-дискретная оптимизация,

# Введение

При разработке управляющих программ для машин фигурной листовой резки с ЧПУ возникает ряд оптимизационных задач, одной из которых является задача минимизации холостого хода инструмента, которая может быть сведена к задаче нахождения кратчайшей ломаной с вершинами на плоских контурах. Контуры интерпретируются как границы плоских деталей. Размещение контуров на плоскости определяется решением задачи фигурного раскроя (задача «нестинга»). Обе задачи, в общем случае, является NP-трудными. В свою очередь, задача нахождения холостого хода инструмента является подзадачей другой оптимизационной задачи – задачи оптимизации маршрута резки плоских деталей. В силу NPтрудности, её точное решение не может быть получено для реально возникающих в производстве задач (для сотен деталей) за разумное время, поэтому для её решения (так же, как и для задачи раскроя) применяются разнообразные эвристики, дающие решения приемлемого качества. В то же время вопрос оценки качества полученных решений в сравнении с оптимальным решением как правило остаётся нерешённым и представляет значительный научный интерес.

## Классификация задач резки

Общая задача маршрутизации резки 2-мерных плоских деталей как правило даже не ставится ввиду её большой сложности. В литературе (см. [1]) можно найти много вариантов её частных случаев, среди которых наиболее популярны (см. рис. 1):

* Задачанепрерывнойрезки (CCP, ContinuousCuttingProblem): каждый замкнутый контур (ограничивающий деталь) вырезается целиком одним движением резака, но резка может начаться с любой точки (и в ней же завершиться), см [5]
* Обобщённаязадачакоммивояжёра (GTSP, GeneralizedTravelingSalesmanProblem): резкаможетначаться только в одной из заранее заданных точек на контуре, контур должен быть вырезан целиком
* Задача резки с конечными точками (ECP, Endpoint Cutting Problem): резка может начаться только в одной из заранее заданных точек на контуре, причём контур может быть вырезан в несколько подходов, по частям.
* Задача непрерывной сегментной резки (SCCP, Segment Continuous Cutting Problem): вводится понятие сегмента резки (представляющее собой обобщение контура; может быть как частью контура, так и объединением нескольких контуров или их частей), сегмент вырезается целиком, таким образом .
* Задачаобобщённойсегментнойрезки (GSCCP, GeneralizedSegmentContinuousCuttingProblem): сегментная резка (SCCP), но выбор сегментов не фиксирован заранее, а сам подлежит оптимизации
* Задачапрерывистойрезки (ICP, IntermittentCuttingProblem) – наиболееобщая описанная в литературе формулировка задачи резки, когда контура могут резаться по частям, в несколько подходов и резка может начаться в любой точке контура.

GSCCP

GSCCP

TSP

GTSP

CCP

ECP

SCCP

ICP

GSCCP

Конечное множество точек врезки

Фиксированные точки врезки

Непрерывное множество точек врезки

Рис 1. Классификация задач маршрутизации резки

Задача резки в общем случае представляет собой сложную задачу непрерывно-дискретной оптимизации с довольно сложной целевой функцией. На практике она очень часто сводится к задаче дискретной оптимизации за счёт дискретизации контуров, подлежащих резке, с некоторым шагом , то есть сводится к задаче ECPили её частному случаю – GTSP.

При этом, однако, общая погрешность длины пути резки или холостого хода достигает , где –количество контуров. Чтобы гарантировать точность результата , требуется выбирать малое , таким образом общее количество точек быстро растёт (как )и полный перебор становится экспоненциальным (). Такие задачи тем не менее могут успешно решаться, например, методом динамического программирования, для небольших даже точно.

В данной статье рассматривается задача маршрутизации резки без использования дискретизации (CCP).

## Технологические ограничения

Необходимость исполнения полученного маршрута на машине листовой резки с ЧПУ накладывает на решение математической задачи резки (например, GTSP) целый ряд ограничений.

Наиболее хорошо описано в литературе так называемое «ограничение предшествования» (precedenceconstraint), вызванное тем, что после вырезания замкнутого контура, его внутренность как правило ничем не удерживается и может сдвигаться, поворачиваться и даже падать. По этой причине внутренние контура деталей должны вырезаться прежде, чем содержащие их внешние контура, а детали, расположенные в отверстиях больших деталей – ещё раньше. В случае задач ECP и ICP эти формулировки ещё усложняются ввиду того, что контуры деталей могут вырезаться не за один раз.

Другим интересным ограничением (не рассматриваемым в данной статье), вытекающим из той же технологической особенности, является необходимость учёта возможности столкновения режущей головки с ранее вырезанными деталями.

Наконец, в большинстве технологий резки требуется, чтобы резка осуществлялась не строго по контуру, а с некоторым отступом. Этот сдвиг может выполняться как в процессе решения задачи маршрутизации, так и после – на этапе генерации управляющей программы для машины резки с ЧПУ или даже самим станком непосредственно в процессе резки. Кроме того, точка врезки (точка включения инструмента) как правило должна быть расположена на ещё большем расстоянии от контура во избежание повреждения детали. Однако, в данной статье это требование не рассматривается. Таким образом, далее везде считается, что инструмент движется точно по контуру детали и точка врезки расположена прямо на контуре (как и точка выключения инструмента).

# Задача непрерывной резки

Рассмотрим эвклидову плоскость и на ней – область , ограниченную замкнутым контуром (как правило – прямоугольник), которая представляет собой модель листового материала, подлежащего резке.Пусть внутри задано попарно непересекающихся плоских контуров, ограничивающих деталей. Деталь может быть ограничена как одним контуром, так и несколькими (внешним и внутренними отверстиями), так что в общем случае .

Контурымогут иметь произвольную форму, но мы будем рассматривать только случай, когда они состоят из (конечного числа) отрезков прямых и дуг окружностей, что определяется существующим технологическим оборудованием. В частном случае, когда контуры состоят только из отрезков прямых, задача непрерывной резки сводится к одному из вариантов задачи обхода многоугольников (TouringPolygonProblem, TPP, см. [2]).

Далее, в области (как правило, на её границе) задаются две точки, обозначим их (на практике как правило ), представляющие собой начало и конец маршрута резки.

Задача непрерывной резки заключается в поиске:

1. точек врезки
2. Порядка обхода контуров, то есть перестановки элементов

Результатом решения задачи будет маршрут . Целевая функция в данном случае сильно упрощается по сравнению с общей задачей резки и сводится к минимизации длины холостого хода

Где для простоты мы вводим обозначения.

Кроме того, мы будем решать задачу оптимизации с дополнительным ограничением, так называемым «ограничением предшествования» (precedenceconstraint). Хотя контуры не пересекаются, они могут быть вложены друг в друга, то есть , где обозначает 2-мерную фигуру, ограниченную контуром (в более привычных обозначениях ). В общей задаче резки это может быть вызвано двумя разными обстоятельствами (отверстия в деталях и размещение меньших деталей в отверстиях больших для экономии материала), но в данной задаче эти варианты обрабатываются одинаково.

Если один контур расположен внутри другого, то вложенный контур должен быть вырезан (посещён) ранее внешнего: в перестановке.Таким образом, не все перестановки контуров оказываются допустимы.

# Общий алгоритм решения

Предлагаемый алгоритм решения состоит из нескольких этапов, естественно связанных с природой решаемой задачи.

## Удаление внешних контуров

Для автоматического соблюдения ограничений предшествования на первом этапе удаляются все контуры, содержащие вложенные, то есть остаются только контура, удовлетворяющие условию:

Это в общем случае приводит к уменьшению размера задачи (в некоторых случаях – значительному), и тем самым сокращает время расчётов на втором и особенно третьем этапе.

## Непрерывная оптимизация

Mi-1

Mi+1

Ci-1

Ci+1

Ci

Mi

M’i

Mi-1

Mi+1

Ci-1

Ci+1

Ci

Mi

M’i

M\*i-1

Рис. 2. Выбор оптимальной точки врезки

На этом этапе мы предполагаем последовательность обхода заданной и ищем координаты точек врезки в каждый контур , минимизирующих суммарную длину холостого хода. Для этого выбираются начальные положения точек (например, случайным образом) и положение каждой точки изменяется в предположении, что все остальные неподвижны: .Большая часть слагаемых в целевой функции константны, поэтому она упрощается до

Простейший геометрический анализ показывает, что неформально говоря, если точки расположены по разные стороны сегмента контура , то оптимальная позиция точки врезки – на пересечении с отрезком: (если такое пересечение существует; в противном случае решением будет один из концов сегмента).В случае же, если точки расположены по одну сторону сегмента, то решение также легко находится с применением принципа Ферма, порождающем знаменитое правило «угол падения равен углу отражения» (либо снова один из концов сегмента), см. рис. 2.

Общая схема оптимизации на данном этапе, таким образом выглядит так:

* Выбираем произвольные начальные положения точек врезки .
* находим оптимальную позицию как описано выше за константное время.
* Повторяем предыдущий шаг, пока позиции всех точек не сойдутся (с некоторой заранее заданной точностью).

На практике весь процесс хорошо сходится за время и поэтому многократно используется как подпрограмма на следующем шаге.

## Дискретная оптимизация

Самый вычислительно сложный этап заключается в поиске перестановки , минимизирующей длину холостого хода, то есть фактически решение задачи коммивояжера (TravellingSalesmanProblem, TSP) с функцией расстояния, вычисляемой при помощи непрерывной оптимизации, как описано на предыдущем шаге.

Для поиска решения используетсяэвристический метод переменных окрестностей (Variable Neighborhood Search, VNS, см. [4]). Схема его применения:

1. Начальная перестановка выбирается случайным образом
2. Пока
   1. Из окрестности выбирается перестановка , минимизирующая
   2. Если
   3. Иначе
3. Конец

На шаге 3.1 многократно применяется шаг непрерывной оптимизации:

Для построения окрестностей разного размера применяются различные приёмы, например:

* Все возможные парные перестановки (фактически это окрестность размера 1 в смысле транспозиционной метрики)
* Циклические перестановки 3 контуров. Все такие перестановки потребовали бы времени , поэтому выбираются только такие варианты, в которых переставляемые контуры удалены в исходной перестановке не более, чем на заранее заданное расстояние (параметр алгоритма).
* Аналогичным образом применяются циклические перестановки 4 контуров, находящихся в пределах некоторого расстояния друг от друга в исходной перестановке .
* Выбирается последовательный блок контуров произвольной длины и осуществляется их циклическая перестановка.
* Перестановка контуров в последовательном блоке произвольной длины «задом наперёд»
* Обмен двух последовательных (но не соседних) блоков контуров
* Циклический сдвиг контуров между несколькими последовательными блоками одинаковой длины
* И ещё порядка десяти различных рецептов построения «близких» перестановок

Если размер окрестности, генерируемой некоторым приёмом, оказывается слишком велик, он может быть легко ограничен введением дополнительного параметра, аналогично тому, как это сделано для тройных и четверных перестановок.

Кроме того, метод переменных окрестностей имеет некоторые вариации, уменьшающие перебор на шаге 3.1, такие как «Первый подходящий» или метод Монте-Карло, однако их влияние на качество и скорость решения задачи непрерывной резки нуждается в дальнейшем исследовании.

## Восстановление удалённых контуров

После того как маршрут инструмента для обхода контуров, не содержащих внутри себя других, построен, как описано выше, достроим его до полного маршрута, посещающего все исходные контуры, причём так, чтобы соблюдалось ограничение предшествования.

Обратим внимание, что полученный на предыдущем шаге маршрут

пересекает все исходные контуры , потому что оставленные на первом этапе контуры он посещает по построению на шагах 2 и 3, а все внешние контура он пересекает, потому что начальная и конечная точки и лежат снаружи всех контуров.

Таким образом для каждого (внешнего) контура , ещё не включённого в маршрут, мы находим все точки пересечения с ним маршрута, построенного на предыдущем шаге и если таких точек оказывается несколько (как правило), то выбираем из них самую последнюю (посещаемую маршрутом позже всех остальных), см. Рис. 3. После добавления полученных таким образом точек врезки мы получаем уже полный маршрут, который, с одной стороны, посещает все исходные контуры, и при этом внешние контуры всегда посещаются позже, чем содержащиеся в них внутренние. Длина маршрута, очевидно, при этом не изменяется.

Внешний контур

Mi

Mi-1

Mi-2

Mi+1

Mi+2

Добавочная точка врезки

Рис 3. Построение дополнительных точек врезки

Легко понять, что построенный таким образом маршрут для полной задачи также является оптимальным. Действительно, если бы существовал более короткий маршрут, обходящий все контуры, мы могли бы простым удалением из него точек врезки, лежащих на внешних контурах получить маршрут обхода только внутренних контуров, имеющий ту же, то есть меньшую длину. Тем самым мы бы построили более короткий маршрут для подзадачи без ограничений предшествования, что по предположению невозможно.

Таким образом, мы выполняем ограничение предшествования почти автоматически за линейное время.

# Оптимальность решения задачи непрерывной оптимизации

С практической точки зрения описанный алгоритм оказывается вполне работоспособным – он генерирует качественные маршруты резки за приемлемое время, однако это эмпирический результат, полученный его применением к раскройным картам. Интересным является теоретическое обоснование свойств получаемых маршрутов. Наибольшую сложность представляет, конечно, третий шаг алгоритма – дискретная оптимизация, причём как с теоретический, так и с практической точки зрения. В данной статье рассматривается второй шаг алгоритма – непрерывная оптимизация. Для неё получается сформулировать некоторые утверждения о качестве получаемого решения.

Итак, рассмотрим следующую задачу, которую приходится решать многократно. Порядок контуров задан. Надо найти ломаную с вершинами на контурах с наименьшей длиной. Дальше предполагается, что контуры состоят только из отрезков прямых.

Берем произвольную начальную ломаную L. Выполняя итерации, получим последовательность ломаных Lk. Итерация: последовательно перебирая контуры i=1,…,n решается следующая задача. Сдвинуть вершину M текущей ломаной, при неподвижных остальных вершинах, в такую точку контура С, чтобы длина ломаной стала наименьшей. А это перебором звеньев контура сводится к решению элементарных задач: на отрезке найти точку, для которой сумма расстояний до двух заданных точек минимальна. Получаем последовательность ломаных { Lk} . Последовательность длин этих ломаных монотонно убывает. Пусть m – минимум этих длин, а L\* – ломаная с длиной m, т.е. предельная точка последовательности { Lk}. ( За метрику в пространстве ломаных можно взять сумму расстояний между вершинами с одинаковыми индексами. )

**Замечание** 2. При проверке на реальных задачах за небольшое число итераций ( не больше 10 ) наступала стабилизация, т.е. получалась ломаная L\*.

В силу построения, длину ломаной L\* нельзя уменьшить, сдвигая ее вершины по одиночке или сдвигая несколько вершин, среди которых нет соседних.

**Утверждение 1.** Если сдвинуть несколько соседних вершин ломаной L\* так, что они остались на тех же звеньях контуров, то длина полученной ломаной не уменьшится.

**Доказательство**. Рассмотрим сдвиг двух соседних вершин. Пусть четыре последовательные вершины ломаной L\* находятся в точках P, ,S. Пусть

точка  принадлежит отрезку  и точка  принадлежит отрезку , соответствующих контуров. Покажем, что для любой точки Q и любой точки R длина ломаной P,Q,R,S не меньше длины ломаной P,,,S.

Для любой точки Q длина ломаной P,Q,,S наименьшая, когда Q= и для любой точки R длина ломаной P,,R,S наименьшая, когда R=. Нужно доказать, что для любых Q и R длина ломаной P,Q,R,S будет наименьшей, когда Q= и R=. Предположим противное, есть точки  и  такие, что длина ломаной P, ,,S меньше длины ломаной P,,,S. Ясно, что  и . Обозначим v – вектор из точки  в точку  и w - вектор из точки  в . Пусть Q(s) =  + sv, R(t) = +tw и функция f(s,t) есть длина ломаной P,Q(s),R(t),S ( s,t  ). В силу положения точек ,  имеем  Если , т.е. производная функции g= - длина ломаной P, Q(s),,S – обращается в 0 при s=0, то сумма длин отрезков [P, Q(s)] и [Q(s),] при движении точки Q(s) вдоль прямой, содержащей отрезок , минимальна при s=0. Значит, точки P,, лежат на одной прямой (считаю, что точки P и  по разные стороны от прямой отрезка ; в противном случае можно заменить точку P на симметричную относительно прямой) и если , то можно считать, что точки ,,S лежат на одной прямой. Поэтому, если  и , то точки P,,,S лежат на одной прямой и поэтому ломаная кратчайшая. Пусть хотя бы одна из этих производных не равна 0. Рассмотрим функцию  Имеем  Поэтому существует , что . По предположению , т.е. имеем  Функция  - это сумма трех слагаемых вида . Поэтому функция  на отрезке [0,1] выпуклая.

Действительно, пусть  , где  и  линейные функции. Тогда вторая производная функции  будет

 т.е. , а значит и  - выпуклые функции.

Во внутренней точке отрезка  функция  принимает значение больше, чем на концах . Это невозможно для выпуклой функции.

Рассмотрен сдвиг двух соседних вершин. Для большего количества вершин доказательство аналогично.

Итак, ломаная L\* доставляет локальный минимум.

Приведем условие, достаточное, чтобы ломаная L\* доставляла глобальный минимум.

Пусть M - вершина ломаной L\* и находится на контуре C. Пусть N и T coседние с M вершины L\*. (В силу замечания 1, точки N, T cнаружи от контура.) По построению L\* для  Q  C сумма длин отрезков [Q,N] и [Q,T] не меньше суммы длин [M,N] и [M,T].

**Условие.** Пусть выполнено одно из следующих требований.

1. Отрезок [N,T] пересекает контур C (тогда точка M принадлежит отрезку [N,T] ).
2. Касательная в точке M к эллипсу с фокусами N и T и проходящему через M, разделяет эллипс и контур C.

**Замечание 3.** Пустьотрезок [N,T] не пересекает контур C.

1. Если вершина M – внутренняя точка отрезка-звена ломаной контура, то касательной из условия 2 является прямая вдоль этого отрезка. (Иначе, на этом отрезке существует точка, для которой сумма расстояний до N и T меньше, чем для M.) Значит, условие 2 выполняется, если весь контур находится по одну сторону от прямой звена.
2. Если вершина M концевая (принадлежит двум звеньям контура), то для выполнения условия 2 достаточно, чтобы весь контур находился внутри угла с лучами из точки M вдоль этих звеньев.
3. Из замечаний a),b) следует: еслиобласть, ограниченнаяконтуром, выпуклая, то условие 2 выполняется.

**Утверждение 2.** Пусть для L\* выполняется **Условие.** Если сдвинуть несколько соседних вершин ломаной L\* так, что они остались на контурах, то длина полученной ломаной не уменьшится, т.е. ломаная L\* доставляет глобальный минимум.

**Доказательство**. Рассмотрим сдвиг двух соседних вершин ломаной L\*, находящихся в точке  на контуре и точке  на контуре . Точки , , соответственно вершины ломаной перед  и после . Нужно показать, что если сдвинуть  в любую точку  и  в любую точку , то длина ломаной ,,, будет не меньше длины ломаной ,,,.

Для  длина ломаной ,,, наименьшая, когда  и для  длина ломаной ,,, наименьшая, когда =. Предположим, что есть точки  и  такие, что длина ломаной ,,, меньше длины ломаной ,,,. Ясно, что  и . Обозначим v – вектор из точки  в точку  и w - вектор из точки  в . Пусть Q(s) =  + sv, R(t) = +tw и функция  = длина ломаной ,Q(s),R(t),. Пусть , т.е. длина ломаной , Q(s),, . В силу условия 2  и значит, , т.е. . Аналогично,  Если  и , то либо точки ,,, лежат на одной прямой, либо это будет после симметрий точек  и/или  относительно прямой, проходящей через точки  и поэтому ломаная ,,, кратчайшая. Пусть хотя бы одна из этих производных не равна 0. Тогда существует , что . Функция на отрезке [0,1] выпуклая и во внутренней точке отрезка принимает значение больше, чем на концах, что невозможно для выпуклой функции. Доказано, что сдвиг точек , не приведет к более короткой траектории.

Рассмотрен сдвиг двух соседних вершин. Для большего количества вершин доказательство аналогично.

**Замечание 4.** Пустькроме траектории L\* из утверждения 2**,** естьдругая траектория, доставляющая глобальный минимум. Тогда из доказательства следует, что как линии они совпадают, т.е. отличие может быть только в точках пересечения контура (см. рис.??? )

**Замечание 5.** Приведем пример, когда траектория, не улучшаемая сдвигами вершин поодиночке, может не доставлять глобального минимума.

На рис. ????? изображены два маршрута, каждый из них не улучшаем сдвигами вершин по одиночке, но зелёный короче красного и только он доставляет глобальный минимум.

C1

C2

M0

M3

C1

C2

M0

M3

# Численные эксперименты

Оценка качества решений описанного алгоритма проводилась на нескольких раскройных планах, содержащих реальные детали. В качестве базы сравнения использовался алгоритм (см. [3]) решения задачи GTSP, который даёт точное решение при количестве контуров .

На рис. 9 приведено точное решение, видны фиксированные положения точек врезки. На рис. 10 показано решение задачи CCPдля того же раскройного плана.

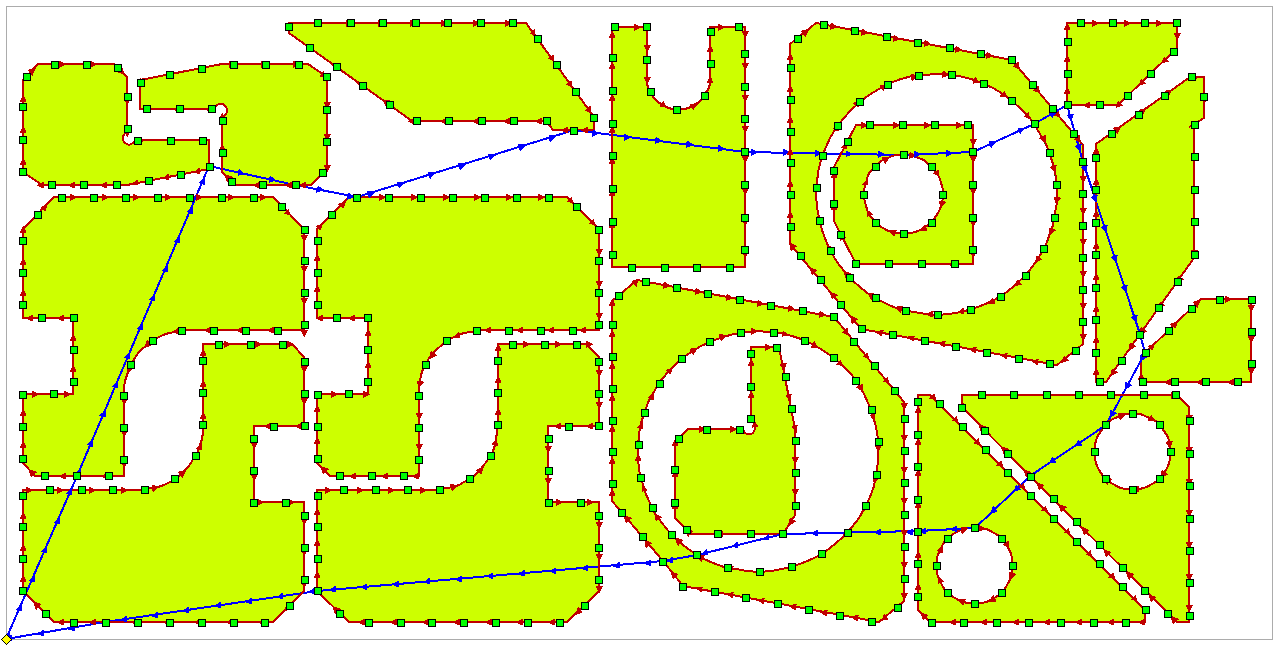


Рис ? Точное решение задачи GTSP

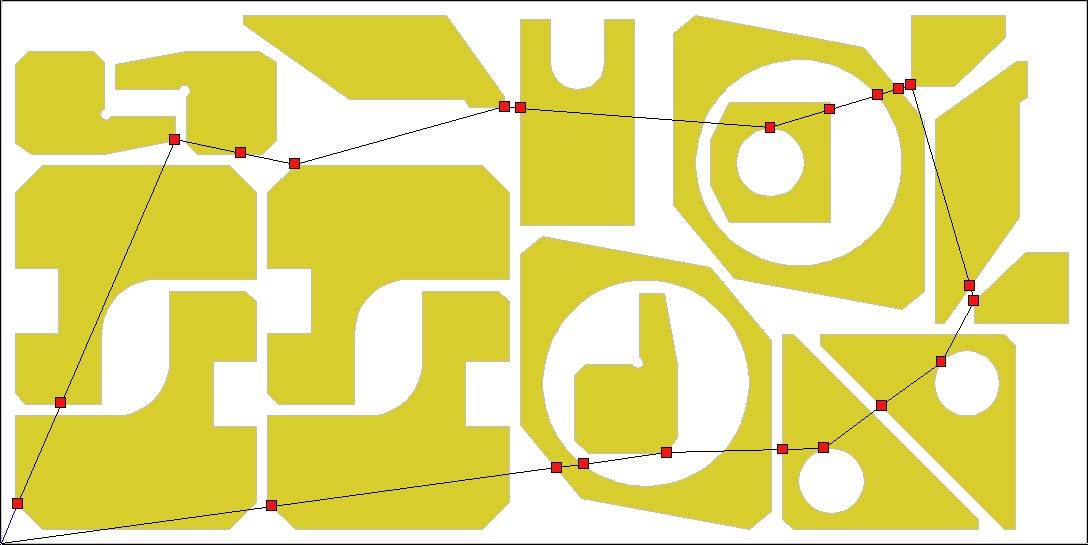


Рис 10. Решение задачи CCP

Видно, что оба алгоритма построили практически идентичный маршрут. Основное отличие вызвано процессом дискретизации для получения задачи GTSP. Из-за этого участки маршрута, прямые в решении CCP, оказываются слегка изломанными в решении GTSP, и суммарная длина холостого хода оказывается незначительно большей. Численно это видно в таблице 1.

| **Задание** | **№ 229** | **№ 464** | **№ 3211** |
| --- | --- | --- | --- |
| Кол-во деталей | 11 | 14 | 17 |
| Кол-во контуров | 12 | 21 | 22 |
| Периметр всех контуров, м | 24.609 | 21.717 | 25.051 |
| Кол-во точек в задаче GTSP | 491 | 429 | 493 |
| Холостой ход GTSP, м | 7.729 | 4.743 | 4.557 |
| Холостой ход CCP, м | 7.727 | 4.706 | 4.536 |

Таблица1. Сравнение качества решения задач GTSPи CCP

# Заключение

1. Показано, что задача минимизации холостого инструмента машин листовой резки с ЧПУ для задачи маршрутизации из класса ССP может быть редуцирована до задачи без условий предшествования, что обеспечивает сокращение числа контуров и время работы алгоритма.
2. Предложен эвристический алгоритм решения задачи CCP, не использующий дискретизацию контуров.
3. Для любой фиксированной последовательности обхода контуров разработан эффективный алгоритм получения локального экстремума и описаны условия, при которых данный локальный экстремум является глобальным минимумом.

В дальнейшем предполагается реализация алгоритма для общего случая, когда точка врезки лежит вне контура в соответствии с технологическими требования листовой резки.

# Библиографический список

1. Dewil R., Vansteenwegen P., Cattrysse D. A review of cutting path algorithms for laser cutters //The International Journal of Advanced Manufacturing Technology. – 2016. – Т. 87. – №. 5-8. – С. 1865-1884.
2. Dror M. et al. Touring a sequence of polygons //Proceedings of the thirty-fifth annual ACM symposium on Theory of computing. – ACM, 2003. – С. 473-482.
3. Chentsov A. G. et al. Model of megalopolises in the tool path optimisation for CNC plate cutting machines //International Journal of Production Research. – 2018. – Т. 56. – №. 14. – С. 4819-4830.
4. Hansen P., Mladenović N., Pérez J. A. M. Variable neighbourhood search: methods and applications //Annals of Operations Research. – 2010. – Т. 175. – №. 1. – С. 367-407.
5. Hoeft J., Palekar U. S. Heuristics for the plate-cutting traveling salesman problem //IIE transactions. – 1997. – Т. 29. – №. 9. – С. 719-731.