На правах рукописи

Уколов Станислав Сергеевич

Разработка алгоритмов оптимальной маршрутизации режущего инструмента для машин термической резки с ЧПУ

Специальность 05.13.12 —
«Системы автоматизации проектирования (промышленность)»

Автореферат
диссертации на соискание учёной степени
кандидата технических наук

Екатеринбург — 2021

Работа выполнена на кафедре «Информационные технологии и автоматизация проектирования» Института новых материалов и технологий ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Научный руководитель: доктор технических наук, доцент
Петунин Александр Александрович

Официальные оппоненты:

Защита состоится XX.XX.XXXX в XX:00 на заседании диссертационного совета УрФУ 05.09.24 по адресу: 620002, г. Екатеринбург, ул. Мира, д.19, ауд. И-420 (зал Ученого совета).

С диссертацией можно ознакомиться в библиотеке и на сайте ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Автореферат разослан «____» _____ 20____ г.

Ученый секретарь
диссертационного совета Огородникова Ольга Михайловна

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования. Современное производство предъявляет высокие требования к качеству заготовок, технико-экономическому уровню выпускаемой продукции, что приводит к увеличению затрат на проектирование и технологическую подготовку производства. Одним из направлений повышения эффективности использования производственных ресурсов является совершенствование безотходных технологий в металлообрабатывающих производствах и возрастание степени их автоматизации.

Раскройно-заготовительные операции, являясь началом большинства производственных процессов, оказывают существенное влияние на трудоёмкость и экономичность изготовления детали. Для получения заготовок сложной геометрической формы из листового материала в условиях мелкосерийного и единичного производства широко применяются машины фигурной резки с числовым программным управлением (ЧПУ). К данному типу оборудования относятся станки газовой, лазерной, плазменной, электроэрозионной и гидроабразивной резки металла. Станки листовой резки имеют множество преимуществ: возможность обработки многих видов материалов различной толщины, высокая скорость резки, возможность обработки контуров различной сложности, адаптация к постоянным изменениям номенклатуры выпускаемой продукции. Использование оборудования с ЧПУ, предполагает применение средств автоматизации проектирования управляющих программ (CAM-систем). При использовании современных CAD/CAM систем, предназначенных для автоматизированного проектирования раскроя и подготовки управляющих программ (далее – УП) для машины с ЧПУ, возникает несколько различных взаимосвязанных задач, поэтому обычно проектирование УП для технологического оборудования листовой резки состоит из нескольких этапов. Первый этап предполагает предварительное геометрическое моделирование заготовок и разработку раскройной карты листового материала. На этапе проектирования раскроя возникает известная задача оптимизации фигурного раскроя листового материала, которая с точки зрения геометрической оптимизации относится к классу трудно решаемых проблем раскроя-упаковки (*Cutting & Packing*). На следующем этапе проектирования УП осуществляется процесс назначения траектории перемещения режущего инструмента (маршрута резки) для полученного на первом этапе варианта раскроя. На этом этапе возникают актуальные научно-практические задачи оптимизации маршрута режущего инструмента. Целью этих задач обычно является минимизация стоимости и / или времени процесса резки, связанного с обработкой требуемых контуров деталей из листового материала, за счет определения оптимальной последовательности вырезки контуров и выбора необходимых точек для термической врезки в листовой материал с учетом технологических ограничений процесса резки. Следует отметить, что современные специализированные САПР предоставляют базовый инструментарий для решения задач рационального раскроя материалов и подготовки УП для техно-

логического оборудования листовой резки с ЧПУ. Вместе с тем разработчики систем автоматизированного проектирования УП для оборудования листовой резки с ЧПУ не уделяют должного внимания проблеме оптимизации маршрута резки. Существующее программное обеспечение САПР не гарантирует получение оптимальных траекторий перемещения инструмента при одновременном соблюдении технологических требований резки, обусловленных необходимостью уменьшения термических деформаций материала, которые могут приводить к существенным искажениям геометрии вырезаемых деталей. Отметим также, что пользователи САПР зачастую используют интерактивный режим проектирования УП. При этом при проектировании маршрута резки зачастую применяется стандартная техника резки «по замкнутому контуру» и путь инструмента строится только с точки зрения минимизации холостых переходов режущего инструмента. В связи с этим актуальным направлением исследования является применение эвристических и метаэвристических подходов, которые позволяют получить решение задачи оптимальной маршрутизации режущего инструмента за приемлемое время.

Степень разработанности темы исследования. Методы проектирования технологических процессов раскроя, включая методы формирования маршрута резки, исследовались в работах как отечественных так и зарубежных ученых. Хотя разработка оптимизационных методов решения задачи раскроя не входит в круг рассматриваемых в диссертационной работе задач, тем не менее, следует упомянуть о значительном вкладе советских и российских исследователей в теорию оптимизации раскроя-упаковки. Работы в этой предметной области были начаты выдающимися учёными В.А. Залгаллером и Л.В. Канторовичем и продолжены в уфимской научной школе Э.А. Мухачевой и её учениками: А.Ф. Валеевой, М.А. Верхотуровым, В.М. Картаком, В.В. Мартыновым, А.С. Филипповой и др. Методологические и теоретические основы создания САПР листового раскроя были заложены Н.И. Гилем, А.А. Петуниным, Ю.Г. Стояном, В.Д. Фроловским.

Разработкой алгоритмов для маршрутизации инструмента машин листовой резки с ЧПУ занимались, в частности, следующие российские исследователи: М.А. Верхотуров, Т.А. Макаровских, Р.Т. Мурзакаев, А.А. Петунин, А.Г. Ченцов, П.А. Ченцов, В.Д. Фроловский, М.Ю. Хачай и др., а также зарубежные исследователи: E. Arkin, N. Ascheuer, D. Cattrysse, R. Dewil, L. Gambardella, J. Hoeft, Y. Jing, Y. Kim , M. Lee, S.U. Sherif, W. Yang и др. В подавляющей части работ используется дискретизация граничных контуров деталей, что позволяет применять различные хорошо разработанные математические модели дискретной оптимизации. Можно отметить только отдельные публикации, где оптимизационные алгоритмы ориентированы на поиск решений среди континуальных множеств.

Общеизвестно, что задачи маршрутизации инструмента машин листовой резки с ЧПУ относятся к NP-трудным задачам При этом следует отметить,

что в настоящее время не существует единой математической модели проблемы оптимизации траектории инструмента для технологического оборудования листовой резки с ЧПУ. Имеются отдельные группы ученых, которые занимаются исследованием частных случаев этой проблемы. Кроме того, в рамках CAD/CAM систем, предназначенных для проектирования раскроя и управляющих программ для машин листовой резки с ЧПУ, есть отдельные модули, которые позволяют решать некоторые оптимизационные задачи, например минимизацию холостого хода инструмента, однако при этом не обеспечивают соблюдение технологических требований резки материала на машинах с ЧПУ и не позволяют получать маршруты резки, близкие к оптимальным с точки зрения критерия стоимости резки с учетом рабочего хода инструмента, затрат на врезку и т.д. Следует подчеркнуть, что алгоритмы, реализованные в коммерческом программном обеспечении, не описываются, как правило, в научной литературе.

В общей проблеме маршрутизации инструмента машин листовой резки с ЧПУ можно выделить несколько классов задач: задача непрерывной резки (CCP), задача резки с конечными точками (ECP), задача прерывистой резки (ICP), задача обхода многоугольников (TPP), задача коммивояжера (TSP) и обобщенная задача коммивояжера (GTSP). Любая задача оптимизации термической резки может рассматриваться как ICP, тем не менее, литература по ICP очень скудна, и большинство программных и научных статей вводят искусственные ограничения, которые упрощают ICP до задач других классов. Поиск хороших алгоритмов оптимизации или эффективного упрощения ICP мог бы заполнить явный и существующий пробел в исследованиях.

В целом можно отметить, что за рамками исследований современных отечественных и зарубежных коллег остаются 3 принципиальных момента:

1. Разработка алгоритмов, обеспечивающих получение глобального оптимума оптимизационной задачи маршрутизации инструмента.

2. Отсутствие адекватного учета тепловых искажений заготовок при термической резке с целевой функцией стоимости, что приводит к не технологичным решениям и искажению геометрии получаемых заготовок.

3. Рассмотрение задач маршрутизации из класса ICP и задач с набором возможных точек врезки из континуального множества.

Применение эффективных классических метаэвристических алгоритмов дискретной оптимизации (метод ветвей и границ, метод эмуляции отжига, метод муравьиной колонии, эволюционные алгоритмы, метод переменных окрестностей и др.) для дискретных моделей оптимизации траектории инструмента машин с ЧПУ возможно только при адаптации этих алгоритмов к требованиям технологических ограничений листовой резки. Таким образом, необходимость в создании специализированных оптимизационных постановок задач, алгоритмов

и программного обеспечения представляется доминантой развития методов решения исследуемой оптимизационной проблемы маршрутизации инструмента машин листовой резки с ЧПУ.

Цель работы заключается в разработке алгоритмов решения задачи оптимальной маршрутизации режущего инструмента и методик применения данных алгоритмов в системах автоматизированного проектирования УП для машин термической резки с ЧПУ. Для достижения поставленной в работе цели необходимо решить следующие **задачи**:

- Разработать эвристику поиска оптимального положения точек врезки в контур детали в процессе решения задачи непрерывной резки
- Подобрать алгоритм поиска решения задачи дискретной оптимизации допускающий совместное использование с разработанной эвристикой поиска точек врезки
- Разработать точный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP), позволяющий оценивать качество решений на основе вычисления нижней оценки
- Разработать программное обеспечение, реализующее разработанные алгоритмы
- Разработать методику использования алгоритмов оптимальной маршрутизации режущего инструмента в CAD/CAM-системах на примере САПР «Сириус»

Научная новизна результатов.

1. Разработан эвристический алгоритм непосредственного решения задачи непрерывной резки без сведения к обобщённой задаче коммивояжера и применения дискретизации
2. Предложена схема учёта ограничений предшествования на основе геометрических соображений, эффективно уменьшающая вычислительную сложность задачи
3. Разработан точный алгоритм, позволяющий находить нижние оценки для обобщённой задачи коммивояжера с ограничениями предшествования
4. Сформулированы более строгие ограничения, позволяющие сократить дерево перебора для обобщённой задачи коммивояжера с ограничениями предшествования

Теоретическая и практическая значимость работы заключается:

1. Прямое решение задачи непрерывной резки позволяет уменьшить длину холостого хода режущего инструмента по сравнению с используемыми в настоящее время технологиями на основе решения задачи GTSP и её аналогов

2. Новая схема учёта ограничений предшествования для задачи непрерывной резки, уменьшая размерность задачи и время счёта, позволяет получать решения задач большего размера, в особенности, имеющих большую вложенность контуров

3. Кроме ускорения процесса проектирования УП, алгоритм решения задачи непрерывной резки также может применяться как составная часть других алгоритмов, что в свою очередь позволяет использовать его для решения задач класса GSCCP (обобщённая сегментная непрерывная резка), то есть искать подходы к решению самой общей задачи маршрутизации режущего инструмента – ICP (прерывистой резки)

4. Предложенные схемы учёта ограничений предшествования и поиска точек врезки могут сочетаться с другими алгоритмами дискретной оптимизации, кроме использованного в данной работе метода переменных окрестностей, что может привести к созданию новых алгоритмов решения задачи непрерывной резки

5. Представленные в данной диссертационной работе необходимые условия глобального минимума могут быть использованы при разработке новых алгоритмов, например, для ограничения перебора или наоборот, его возобновления с целью улучшения качества решения.

6. Предложенный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования способен находить точные решения для задач, которые были ранее недоступны для точного решения

7. За счёт вычисления нижних оценок, данный алгоритм может использоваться также для оценки качества решений, полученных при помощи других алгоритмов

8. В предложенной схеме использованы несколько способов получения нижней оценки, что позволяет получать более точные результаты и сокращать объём перебора

9. Допустимо использование также других способов получения оценок, что открывает дорогу новым теоретическим и практическим исследованиям

10. Предложенные алгоритмы могут использоваться для подготовки УП для машин термической резки с ЧПУ в составе САПР «Сириус», а также других CAD/CAM-систем

Методология и методы исследования. Методологическую базу исследования составили фундаментальные и прикладные работы отечественных и зарубежных ученых в области автоматизированного проектирования маршрута резки для машин листовой резки с ЧПУ, геометрического моделирования, разработки алгоритмов оптимальной маршрутизации, методы вычислительной геометрии и компьютерной графики. В качестве инструментов исследования использовались следующие методы: анализ, синтез, классификация, формализация, математические методы обработки данных. Оценка эффективности предложенных методов и алгоритмов осуществлялась с помощью вычислительных экспериментов на различных раскройных картах и тестовых примерах. Проводилось их сравнение с результатами, полученными при работе алгоритмов других авторов.

Положения, выносимые на защиту:

1. Схема сведения задачи непрерывной резки с ограничениями предшествования к аналогичной меньшего размера и не имеющей ограничениями
2. Эвристика поиска точек врезки в плоские контура, не использующая дискретизации
3. Достаточные условия, при которых полученный маршрут доставляет глобальный минимум длины холостого хода инструмента
4. Схема использования ограничений предшествования для сокращения перебора в обобщённой задаче коммивояжера
5. Точный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования с обновлением нижней границы
6. Форматы файлов для обмена геометрической и маршрутной информацией и визуализации для использования алгоритмов оптимальной маршрутизации в CAD/CAM-системах

Достоверность результатов диссертационной работы подтверждается результатами экспериментальных исследований, приведенными в ряде публикаций и полученными при использовании методик, алгоритмов и программных средств, созданных при непосредственном участии соискателя. Основные положения диссертации были представлены на международных и всероссийских научных конференциях, опубликованы в изданиях ВАК, Scopus, WoS, известны в научном сообществе и положительно оценены специалистами.

Апробация результатов работы. Основные результаты работы докладывались и обсуждались на всероссийских и международных конференциях, в том числе:

- *Applications of Mathematics in Engineering and Economics*, Созополь, Болгария, 2016 год
- *MiM2016: on Manufacturing, Modelling, Management & Control*, Труа, Франция, 2016 год
- *ASRTU 2017 International Conference on Intellectual Manufacturing*, Харбин, Китайская Народная Республика, 2017 год
- *Mathematical Optimization Theory And Operations Research*, Екатеринбург, Россия, 2019 год
- *Manufacturing Modelling, Management and Control - 9th MIM 2019*, Берлин, Германия, 2019 год
- *XVI Всеросиийская научно-практическая конференция «Перспективные системы и задачи управления»*, Домбай, Россия, 2021 год

Личный вклад автора состоит в проведении теоретических и экспериментальных исследований по теме диссертационной работы, проведении аналитических расчетов на основе полученных результатов. В опубликованных совместных работах постановка и разработка алгоритмов для решения задач осуществлялись совместными усилиями соавторов при непосредственном активном участии соискателя.

По теме диссертационной работы опубликовано ? научных работ, среди которых ? статей в журналах, определенных ВАК и Аттестационным советом УрФУ, включая ? статей в изданиях, индексируемых в международных базах WoS и Scopus.

Структура и объем диссертации. Диссертация состоит из введения, четырех глав, заключения, списка литературы и ? приложений. Общий объем диссертации составляет ? с., в том числе ? рисунков, ? таблиц. Список литературы включает ? наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность диссертационной работы, определены цель и задачи исследования, представлена научная новизна, теоретическая и практическая значимость работы. Приведены основания для выполнения работы, ее апробация и структура.

В **первой главе** проводится анализ проблемы автоматизированного проектирования УП в раскройно-заготовительном производстве для оборудования термической фигурной резки с ЧПУ. Описаны основные технологические особенности и ограничения термической резки, которые необходимо соблюдать при проектировании УП.

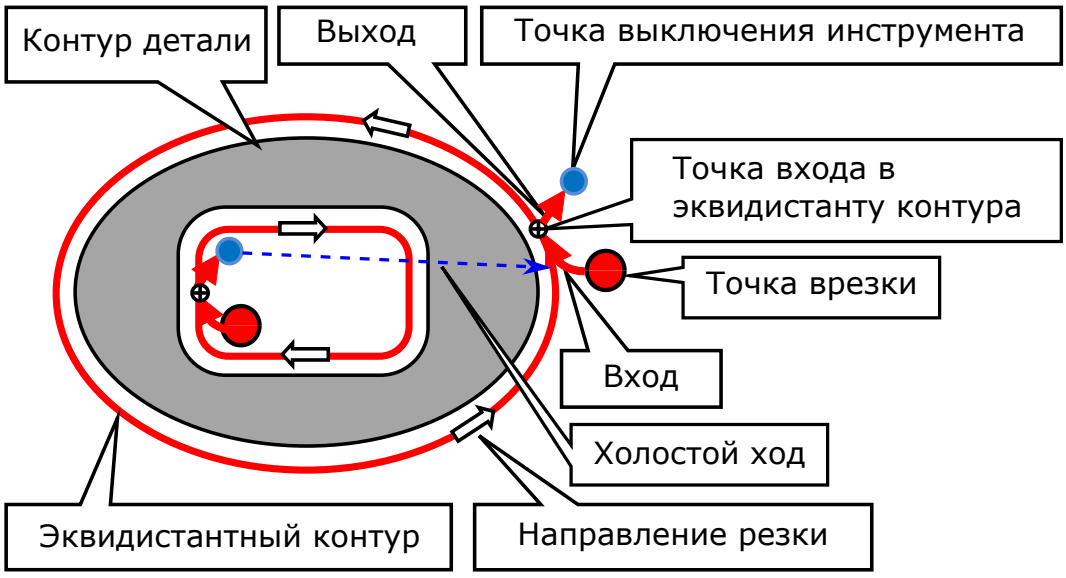

Рис. 1. Элементы маршрута резки

В общем случае маршрут инструмента содержит несколько компонент: начальную M_0 и возможно отличную от неё конечную точку маршрута M_{N+1}, N точек врезки M_i, $i \in \overline{1,N}$ и соответствующих им точек выключения инструмента M_i^*, рабочий ход инструмента от точки врезки M_i до точки выключения M_i^* (в дальнейшем будем называть его траекторию *сегментом резки* $S_i = M_i M_i^*$) а также холостой ход от M_i^* до следующей точки врезки M_{i+1}, см. рис. 1. В определение маршрута также входит порядок резки, то есть последовательность посещения точек врезки, представляющая собой перестановку $I = (i_1, i_2, ...i_N)$. Таким образом, маршрут резки можно определить в терминах сегментов резки как кортеж

$$\mathfrak{R} = \langle M_0, M_1, S_1, M_1^*, M_2, S_2, M_2^*, \ldots, M_N, S_N, M_N^*, i_1, i_2, \ldots, i_N \rangle \qquad (1)$$

В качестве целевой функции при оптимизации часто используется время резки

$$T_{cut} = \frac{L_{on}}{V_{on}} + \frac{L_{off}}{V_{off}} + N_{pt} \cdot t_{pt}, \qquad (2)$$

где L_{on} – длина реза с включенным режущим инструментом; V_{on} – скорость рабочего хода режущего инструмента; L_{off} – длина переходов с выключенным режущим инструментом (холостой ход); V_{off} – скорость холостого хода; N_{pt} – количество точек врезки; t_{pt} – время, затрачиваемое на одну врезку. В подавляющем большинстве исследований, включая данную диссертационную работу, V_{on} считается константой (в рамках конкретной УП), однако вообще говоря это не так, см. [7].

Важнейшей экономической характеристикой качества разработанной управляющей программы является стоимость (себестоимость) резки деталей на машине с ЧПУ. По аналогии с (2) его можно определить по формуле

$$F_{cost} = L_{on} \cdot C_{on} + L_{off} \cdot C_{off} + N_{pt} \cdot C_{pt}, \tag{3}$$

где C_{on} – стоимость единицы пути с включенным режущим инструментом; C_{off} – стоимость единицы пути с выключенным режущим инструментом; C_{pt} – стоимость одной врезки.

Задача оптимизации маршрута инструмента для машин фигурной листовой резки с ЧПУ может быть представлена в общем виде как задача минимизации некоторой числовой функции $\mathfrak{F}$ (например, (2) или (3)) на множестве $\mathfrak{G}$ допустимых кортежей (1):

$$\mathfrak{F}(\mathfrak{R}) \to \min_{\mathfrak{R} \in \mathfrak{G}} \tag{4}$$

В маршрут (1) помимо перестановки $I = (i_1, i_2, ...i_N)$ входят также точки M_i и M_i^*, которые в общем случае должны ещё быть выбраны из некоторого непрерывного множества, что делает простую формулировку задачи (4) чрезвычайно сложной в решении. Более того, набор сегментов резки S_i в общем случае не совпадает (даже в количестве) с набором контуров деталей, подлежащих резке, так как на практике используются различные техники резки:

1. *Резка по замкнутому контуру (стандартная техника)*: сегмент резки содержит ровно один замкнутый контур заготовки и вырезается целиком.

2. *Мультисегментная резка*: для вырезки одного контура используются не менее двух сегментов резки.

3. *Мультиконтурная резка*: резка предполагает вырезку нескольких контуров в одном сегменте.

В зависимости от используемой техники резки и способа выбора точек врезки можно выделить несколько классов задач резки[1], изображенных на рис. 2:

[1] *Dewil R.* A review of cutting path algorithms for laser cutters / R. Dewil, P. Vansteenwegen, D. Cattrysse // International Journal of Advanced Manufacturing Technology. 2016. Т. 87, № 5. С. 1865—1884.

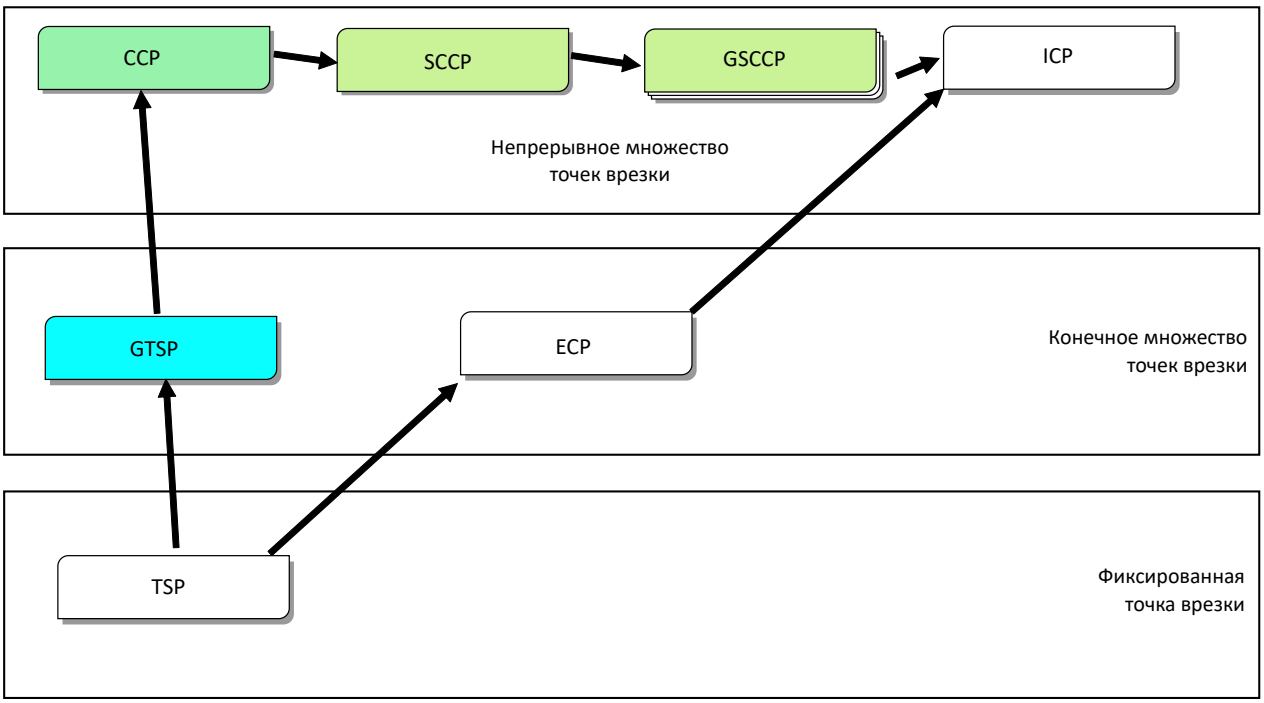

Рис. 2. Классификация задач резки

– **Задача непрерывной резки** (Continuous Cutting Problem, CCP): каждый контур 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): сегмент резки может быть частью контура или объединением нескольких контуров и / или их частей. Каждый сегмент вырезается целиком, от начала до конца, таким образом $CCP \subset SCCP$.

– **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.

На практике задача маршрутизации режущего инструмента чаще всего решается как задача дискретной оптимизации, для этого непрерывный контур детали заменяется на конечное число потенциальных точек врезки, как правило расположенных на нём с некоторым шагом ε, то есть фактически сводится к ECP или её частному случаю – GTSP.

Полученный любым способом маршрут движения режущего инструмента, должен быть исполнен на конкретном промышленном оборудовании – режущей машине с ЧПУ. Это накладывает ряд существенных ограничений на решение задачи резки, то есть ограничивает множество $\mathfrak{G}$ в (4).

Наиболее популярным и хорошо описанным в литературе является так называемое *ограничение предшествования* (Precedence Constraint, PC), возникающее из-за того, что после вырезания замкнутого контура, его внутренняя часть ничем не удерживается и может сдвигаться, поворачиваться, наклоняться или даже падать. Поэтому внутренние отверстия деталей следует вырезать до того, как будет завершена резка внешнего контура детали. Аналогично, если меньшая деталь размещается (в целях экономии расхода материала) в отверстии большей детали, она должна целиком быть вырезана до того, как будет завершена резка содержащего её отверстия и тем более – внешнего контура большей детали. В терминах маршрута (1) не все перестановки $I = (i_1, i_2, ...i_N)$ оказываются допустимы.

Большинство технологий резки (например, лазерная, газовая или плазменная) требуют, чтобы резак двигался не строго по контуру детали, а с некоторым припуском, так как часть материала повреждается в процессе резки. Введение этого припуска может происходить на разных этапах технологической подготовки производства – при решении задачи резки; при генерации управляющей программы для станка с ЧПУ; самим станком ЧПУ прямо в процессе резки. Точка врезки в контур (включения резака) должна располагаться как правило ещё на большем расстоянии от контура детали во избежание её повреждения. В данной диссертационной работе, однако, эти ограничения не рассматриваются, то есть далее везде предполагается, что режущий инструмент движется точно по контуру детали и точка врезки (которая в данном случае является и точкой выключения инструмента после окончания вырезания контура) всегда расположена точно на контуре.

В литературе описан ещё целый ряд ограничений, накладываемых на маршрут режущего инструмента, порождаемых технологическими свойствами современных машин термической резки с ЧПУ, которые также должны учитываться при практическом применении, см. в частности [3; 4]. В данной диссертационной работе эти технологические ограничения также не рассматриваются.

Во **второй главе** рассматривается задача непрерывной резки *CCP* на Евклидовой плоскости $\mathbb{R} \times \mathbb{R}$. Возьмём N попарно непересекающихся плоских контуров $\{C_1, C_2, ...C_N\}$, ограничивающих n деталей $\{A_1, A_2, ...A_n\}$. В общем случае $n \leqslant N$. В данной работе рассматриваются только контуры C_i, 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. Выберем также две точки M_0, M_{N+1} (почти всегда $M_0 = M_{N+1}$), которые будут использоваться как начало и конец маршрута резки. Задача непрерывной резки (*Continuous Cutting Problem, CCP*) состоит в поиске:

1. N точек врезки $M_i \in C_i, i \in \overline{1, N}$
2. Последовательности обхода контуров C_i, то есть перестановки N элементов $I = (i_1, i_2, ...i_N)$

Результатом решения задачи будет являться маршрут

$$\mathcal{R} = \langle M_0, M_{i_1}, M_{i_2}, \dots M_{i_N}, M_{N+1} \rangle \tag{5}$$

Целевая функция сводится фактически к минимизации длины холостого хода:

$$\mathcal{L} = \sum_{j=0}^{N} |M_{i_j} M_{i_{j+1}}| \tag{6}$$

$$\mathcal{L} \to \min$$

где, для простоты записи мы полагаем $M_{i_0} = M_0$, $M_{i_{N+1}} = M_{N+1}$.

Кроме того, решение должно удовлетворять ограничению предшествования: если $\widetilde{C}_i$ обозначает 2-мерную фигуру, ограниченную контуром C_i (в более традиционных обозначениях $C_i = \partial\widetilde{C}_i$), то $\widetilde{C}_p \subset \widetilde{C}_q \Rightarrow i_p < i_q$, то есть вложенный контур должен быть посещён раньше, чем содержащий его, и не все перестановки $I = (i_1, i_2, ...i_N)$ допустимы.

Предлагаемый алгоритм (см. [2; 8]) решения задачи:

1. Удаление «внешних» контуров
2. Поиск положений точек врезки (непрерывная оптимизация)
3. Поиск порядка обхода контуров (дискретная оптимизация без учёта ограничений предшествования)
4. Восстановление удалённых контуров

На первом шаге мы удаляем все контура, внутри которых содержатся другие, то есть оставляем только

$$\{C_i | \forall j \neq i \colon C_j \cap \widetilde{C}_i = \varnothing\},$$

тем самым как правило сокращая размерность задачи с N до некоторого $N' \leqslant N$.

На втором шаге мы предполагаем перестановку контуров $I = (i_1, i_2, ...i_N)$ фиксированной, выбираем произвольные положения точек врезки $M_i \in C_i$ на контурах и подвергаем их последовательной релаксации: для каждой точки M_i мы полагаем все остальные M_j $(i \neq j)$ фиксированными и находим положение M_i, минимизирующее функционал

$$|M_{i-1}M_i| + |M_iM_{i+1}| \to \min_{M_i \in C_i}$$

На практике этот процесс очень быстро сходится, давая за время $O(N')$ позиции точек врезки на всех контурах.

На третьем шаге предполагается воспользоваться каким-либо методом дискретной оптимизации для поиска перестановки $I = (i_1, i_2, ...i_N)$. В данной диссертационной работе использован метод переменных окрестностей (Variable Neighborhood Search, VNS[2]). Мы также начинаем с произвольной (случайной) перестановки I, строим окрестность этой перестановки $\mathcal{N}(I)$ (например, все перестановки, полученные из неё всеми однократными попарными перестановками контуров), для каждой перестановки $I' \in \mathcal{N}(I)$ находим оптимальные позиции точек врезки для минимизации холостого хода

$$\mathcal{L}(I') = \min_{M_i \in C_i, \forall i} \mathcal{L}(M_1, M_2 \dots M_N | I')$$

(как описано выше на втором шаге алгоритма) и выбираем ту из перестановок I', которая даёт наименьшее значение длины холостого хода (6), и к ней применяется этот же процесс. Если же понизить длину холостого хода не получается, рассматриваются всё более широкие окрестности $\mathcal{N}(I)$ (например, полученные тройными перестановками контуров и т.п.), пока не будет обнаружена перестановка I, которая уже не может быть улучшена. Она и считается решением задачи вместе с соответствующими ей позициями точек врезки.

На четвёртом и последнем шаге алгоритма мы восстанавливаем контуры, удалённые на первом шаге и находим точки врезки и для них как пересечение маршрута

$$\mathcal{R} = \langle M_0, M_1, M_2, \dots M_{N'}, M_{N+1} \rangle \tag{7}$$

с каждым из (удалённых) контуров $M_i = C_i \cap \mathcal{R}$, причём из нескольких таких точек выбирается самая последняя по ходу маршрута (7). 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. Получаемый таким образом за линейное время $O(N)$ полный маршрут является оптимальным решением исходной задачи непрерывной резки, соблюдающим ограничение предшествования.

[2] *Hansen P.* Variable neighbourhood search: methods and applications / P. Hansen, N. Mladenović, J. A. Moreno Pérez // Annals of Operations Research. 2010. Т. 175, № 1. С. 367—407.

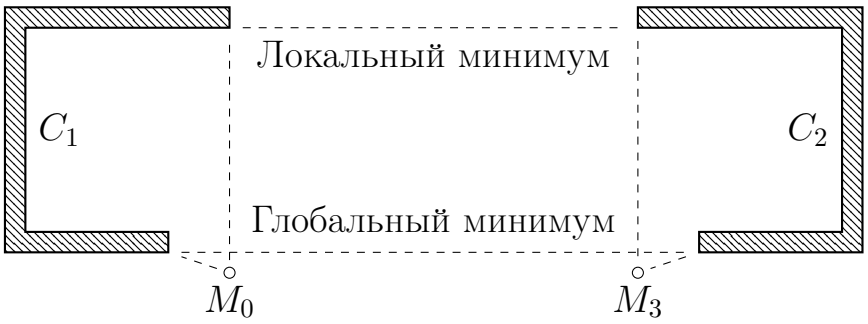

Рис. 3. Два маршрута резки для одной задачи ССР

Как для любой эвристики, возникает вопрос оценки качества решения, получаемого данным алгоритмом. Действительно, легко построить пример, см. рис. 3, когда (даже для фиксированного порядка посещения контуров), может быть получено два маршрута резки, в зависимости от начальных точек врезки.

В ходе данной диссертационной работы были доказаны несколько свойств получаемого маршрута в предположении, что все контуры являются многоугольниками:

– *Локальный минимум*: сдвиг нескольких соседних точек врезки на маршруте (5) в пределах **тех же самых** сегментов контуров, на которых они расположены, не приводит к уменьшению полной длины маршрута (6).

– *Глобальный минимум*. сдвиг нескольких соседних точек врезки на маршруте (5) в пределах содержащих их контуров не приводит к уменьшению полной длины маршрута (6), если для каждой точки врезки $M_i \in C_i$ выполнено одно из условий:

  1. Сегмент $M_{i-1}M_{i+1}$ пересекает контур C_i, то есть $M_i \in M_{i-1}M_{i+1}$
  2. Касательная в точке M_i к эллипсу с фокусами M_{i-1} и M_{i+1} и проходящему через M_i, разделяет эллипс и контур C_i

     Это условие легко проверяется программно, однако можно выделить несколько практически полезных частных случаев, когда оно проверяется просто визуально:

     – Вершина M_i является внутренней точкой одного из отрезков контура C_i и при этом весь контур расположен по одну сторону линии, проходящей через этот отрезок
     – Вершина M_i является также вершиной контура C_i (принадлежит сразу двум его отрезкам) и при этом весь контур находится внутри угла, образованного лучами, идущими из вершины M_i вдоль этих двух отрезков
     – Контур C_i ограничивает собой выпуклый многоугольник $\widetilde{C}_i$.

С практической точки зрения алгоритм работает очень хорошо, быстро находя хорошие маршруты резки. Однако, объективная оценка качества решения сложна. В данной работе в качестве базы сравнения использовался алгоритм на основе динамического программирования[3], который находит точное решение задачи GTSP для количества контуров $N \leqslant 33$. Использовались несколько раскройных планов, содержащих реальные детали, см. табл. 1.

Таблица 1

Сравнение качества решений задач CCP и GTSP

Задание	№ 229	№ 464	№ 3211
Кол-во деталей	11	14	17
Кол-во контуров	12	21	22
Общий периметр, м	24.609	21.717	25.051
Кол-во точек GTSP	491	429	493
$\mathcal{L}_{GTSP}$, м	7.729	4.743	4.557
$\mathcal{L}_{CCP}$, м	7.727	4.706	4.536

На рис. 4 показано точное решение задачи GTSP. Хорошо видны возможные положения точек врезки, которые получены путём дискретизации контура, то есть сведения задачи непрерывной оптимизации к задаче дискретной оптимизации. На рис. 5 показано решение задачи непрерывной резки, полученное вышеописанным алгоритмом для того же раскройного плана.

Видно, что оба алгоритма дают практически идентичные маршруты резки. Основное отличие вызвано необходимостью дискретизации контуров в ходе сведения задачи непрерывной резки к GTSP. Это приводит, в частности к тому, что длина холостого хода в задаче GTSP получается немного больше, чем в задаче CCP, что видно в табл. 1.

В **третьей главе** рассматривается обобщённая задача коммивояжера с ограничениями предшествования ($PCGTSP$) – это хорошо известная задача комбинаторной оптимизации, имеющая множество приложений помимо оптимизации траектории режущего инструмента и привлекшая внимание многих исследователей. К сожалению, в отличие от задачи GTSP, алгоритмические подходы к решению именно PCGTSP всё ещё малочисленны. Алгоритм, предложенный в данной диссертационной работе, на основе синтеза нескольких идей других исследователей, представляет собой попытку создать первый точный алгоритм ветвей и границ для решения задачи PCGTSP в общем виде.

Задача определяется тройкой $(G, \mathcal{C}, \Pi)$, где $G = (V, E, c)$ – взвешенный ориентированный граф на n вершинах, задающий веса $c(u, v)$ для всех своих ребер $(u, v) \in E, u, v \in V$; $\mathcal{C} = \{V_1, \ldots, V_m\}$ – разбиение вершин графа G на m

[3] *Chentsov A. G.* Model of megalopolises in the tool path optimisation for CNC plate cutting machines / A. G. Chentsov, P. A. Chentsov, A. A. Petunin, A. N. Sesekin // International Journal of Production Research. 2018. Т. 56, № 14. С. 4819—4830.

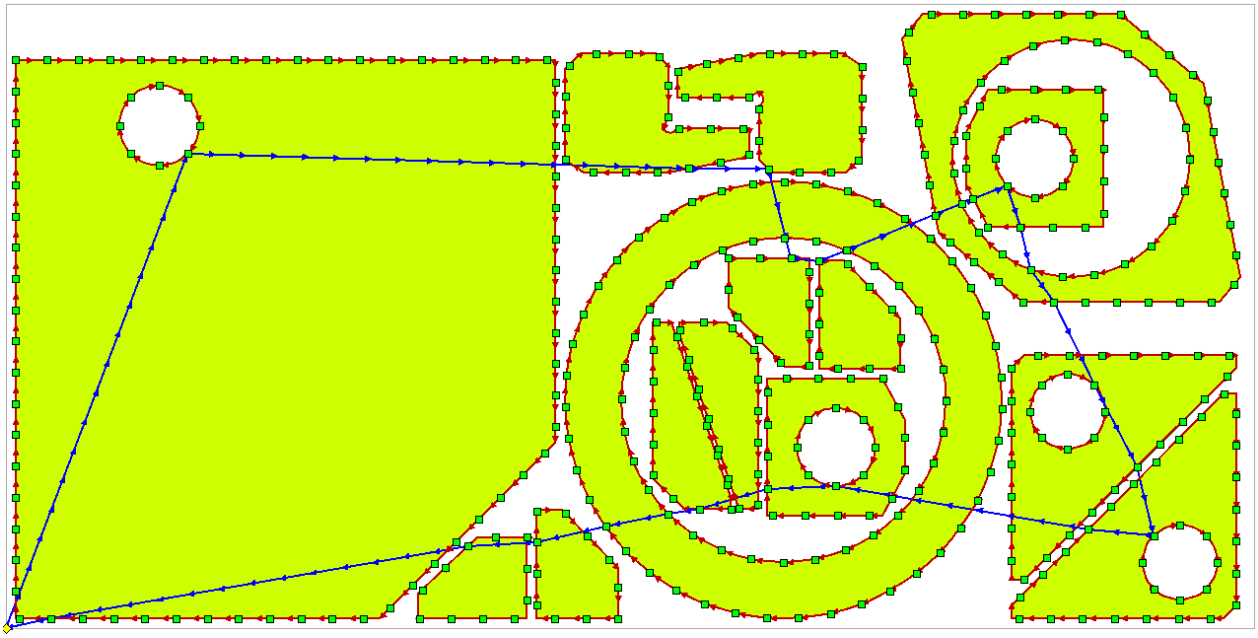

Рис. 4. Точное решение задачи GTSP для задания № 464

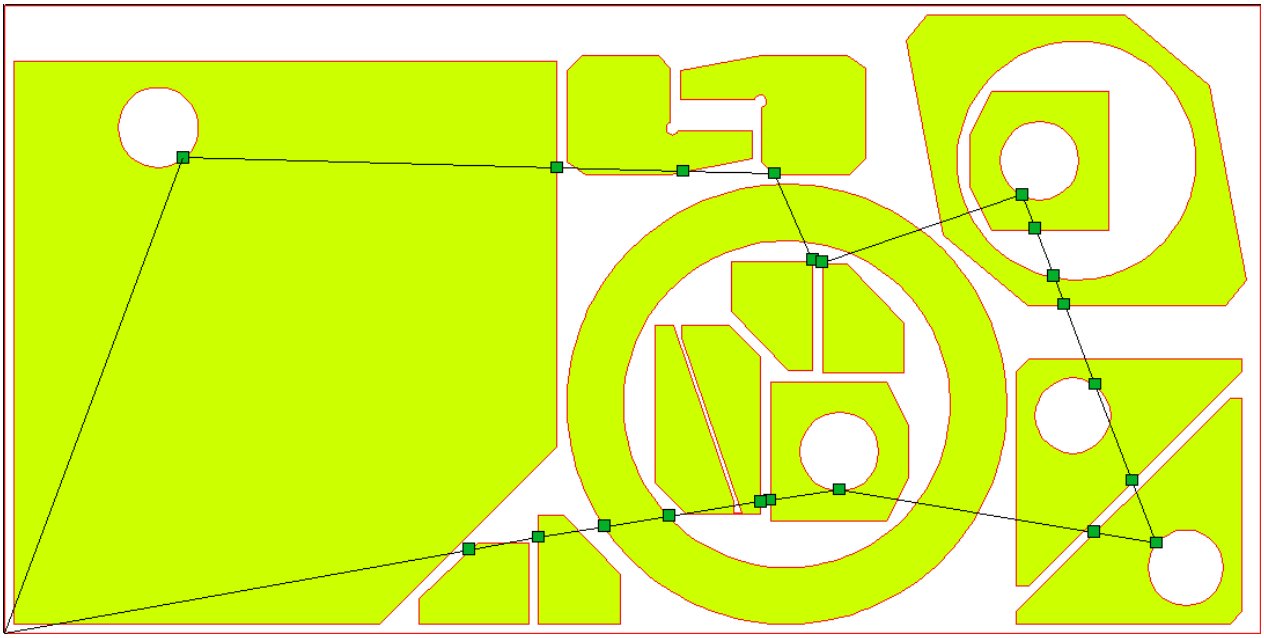

Рис. 5. Решение задачи непрерывной резки для задания № 464

кластеров; $\Pi = (\mathcal{C}, A)$ – частичный порядок, заданный на множестве кластеров. Для каждой вершины $v \in V$, за $V(v)$ обозначим (единственный) кластер $V_p \in \mathcal{C}$, такой что $v \in V_p$.

Допустимым решением задачи $(G, \mathcal{C}, \Pi)$ называется тур (замкнутый путь) T, удовлетворяющий условиям:

- имеет длину $|T| = m$
- начинается и заканчивается в некоторой вершине $v_1 \in V_1$
- посещает каждый кластер $V_p \in \mathcal{C}$
- каждое ребро (v_i, v_j) в T (кроме ребра (v_m, v_1)) удовлетворяет ограничению предшествования, то есть $(V(v_i), V(v_j)) \in A$.

Каждому решению $T = v_1, v_2, \ldots, v_m$ мы назначаем стоимость

$$cost(T) = c(v_m, v_1) + \sum_{i=1}^{m-1} c(v_i, v_{i+1}) \tag{8}$$

Требуется найти допустимый тур T с минимальной стоимостью

$$cost(T) \to \min_T$$

Ключевой идеей алгоритма является построение нижней оценки стоимости решения. Для этого в каждой вершине дерева поиска исходная задача разделяется на две:

1. Фиксируется *префикс* маршрута $\sigma = \{V_1, \ldots, V_r\}$. Обозначим c_{min} нижнюю границу длины всех путей, проходящих из вершин кластера V_1 в вершины кластера V_r строго в порядке σ.
2. Построим вспомогательную задачу $\mathcal{P}$, удалив из исходной задачи все кластеры, являющиеся внутренними в σ и соединив все вершины из V_1 с вершинами V_r ребрами нулевого веса. Полученная таким образом задача всё ещё сложна, однако она может быть несколькими способами упрощена (релаксирована) $\mathcal{P} \to \mathcal{P}_{rel}$ и для неё найдено оптимальное решение $\mathrm{OPT}(\mathcal{P}_{rel})$

Нижняя оценка LB находится как

$$LB = c_{\min} + \mathrm{OPT}(\mathcal{P}_{rel}) \tag{9}$$

В качестве релаксации $\mathcal{P}_{rel}$ задачи $\mathcal{P}$ могут применяться:

- хорошо известная трансформация Noon-Bean, преобразующая задачу GTSP в обычную задачу коммивояжера TSP

– построение вспомогательного *графа кластеров* H_1 с весами, индуцированными весами исходного графа G:

$$c(V_i, V_j) = \min_{v_i \in V_i, v_j \in V_j} c(v_i, v_j)$$

– построение графа кластеров H_2, веса в котором также индуцированы весами путей длины 2 в исходном графе G:

$$c(V_i, V_j) = \min_{\substack{v_i \in V_i, v_j \in V_j \\ v_k \notin V_i \cup V_j}} \frac{c(v_i, v_k) + c(v_k, v_j)}{2}$$

при этом маршрут v_i, v_k, v_j должен удовлетворять ограничению предшествования П. Этот способ позволяет точнее учитывать взаимное расположение контуров на плоскости в случае Евклидовой задачи PCGTSP

– по аналогии с H_2 могут строиться кластерные графы с весами на основе путей большей ($\geqslant 3$) длины, однако алгоритм их расчёта существенно сложнее и не входит в рамки данной диссертационной работы

Наконец, для (быстрого) поиска нижней оценки на решение задачи $TSP(\mathcal{P}_{rel})$ можно использовать:

– Решение задачи о минимальном остовном дереве (Minimal Spanning Arborescence, MSAP), так как $MSAP(\mathcal{P}) \leqslant TSP(\mathcal{P})$

– Решение задачи о цикловом покрытии (Cycle cover); оно находится как решение задачи о назначениях (Assignment Problem, AP) для двудольного графа, обе доли которого представляют $\mathcal{P}_{rel}$; опять $AP(\mathcal{P}) \leqslant TSP(\mathcal{P})$

– в некоторых случаях можно прямо решить задачу коммивояжера $TSP(\mathcal{P})$. В данной работе для этого использовался решатель Gurobi и MIP-модель ATSPxy[4].

Все возможные методы оценки нижней границы показаны в табл. 2, используемые в описываемой реализации обозначены L_1–L_3, не используемые – E_1–E_6. Проведённые численные эксперименты показывают, что часть методов (E_1–E_4) систематически дают более слабые оценки, а часть (E_1–E_2, E_5–E_6) требуют значительных временных затрат по сравнению с выбранными.

Получив описанными методами оценку нижней границы для данного префикса по формуле (9), алгоритм принимает решение об отсечении ветви, порождаемой текущим префиксом σ, при условии

$$LB > UB, \tag{10}$$

[4] *Sarin S. C.* New Tighter Polynomial Length Formulations for the Asymmetric Traveling Salesman Problem with and without Precedence Constraints / S. C. Sarin, H. D. Sherali, A. Bhootra // Oper. Res. Lett. NLD, 2005. Т. 33, № 1. С. 62—70.

Таблица 2

Методы оценки нижней границы

	Noon-Bean	$\mathbf{H_1}$	$\mathbf{H_2}$	$\mathbf{H_{3+}}$
AP	E_1	$\mathbf{L_1}$	$\mathbf{L_2}$	-
MSAP	E_2	E_3	E_4	-
Gurobi	E_5	$\mathbf{L_3}$	E_6	-

где UB – стоимость наилучшего известного допустимого решения исходной задачи. В данной работе для получения UB используется эвристика PCGLNS[5], позволяющая буквально за несколько секунд получить близкое к оптимальному решение исходной задачи PCGTSP, что резко сокращает перебор, позволяя отбрасывать $\approx 50\%$–90% ветвей.

Предлагаемый алгоритм ветвей и границ обходит дерево поиска, начиная с корня $\sigma = \{V_1\}$ в ширину (*Breadth-first search*), для каждого узла находя оценку нижней границы (9), проверяя условие отсечения (10) и для «выживших» узлов применяя процедуру ветвления. Для этого он находит все кластера, которые можно добавить в конец текущему префиксу, не нарушая ограничение предшествования П. По окончании подсчёта префиксов одной длины, алгоритм обновляет текущую оценку нижней границы:

$$LB = \max(LB, LB_r)$$

$$LB_r = \min_{|\sigma|=r} LB(\sigma),$$

в этот момент алгоритм может быть остановлен по достижении нужной точности. Если же алгоритм доходит до обработки префиксов длины $|\sigma| = m$, то для них вместо оценки (9) по известному порядку обхода кластеров σ находится решение исходной задачи путём поиска кратчайшего тура, посещающего кластера в этом порядке, и из всех таких решений выбирается оптимальное.

Такой алгоритм оказывается вполне работоспособен, однако в ходе его тестирования были выявлены некоторые недостатки:

– в некоторых задачах использованные методы оценки нижней границы (см. табл. 2) дают очень низкие значения

– Сведение всех путей вдоль префикса σ к минимальному c_{min} представляется слишком грубым

– все префиксы σ, оканчивающиеся на один и тот же кластер V_r, но разный порядок «внутренних» кластеров, оказываются тесно связаны; это делает сложным попытки распараллелить выполнение алгоритма

[5] *Khachay M.* PCGLNS: A Heuristic Solver for the Precedence Constrained Generalized Traveling Salesman Problem / M. Khachay, A. Kudriavtsev, A. Petunin // Optimization and Applications. T. 12422 / под ред. N. Olenev, Y. Evtushenko, M. Khachay, V. Malkova. Cham : Springer International Publishing, 2020. C. 196—208. (Lecture Notes in Computer Science).

Поэтому была разработана вторая версия того же алгоритма, устроенная по классической схеме динамического программирования (DP) Хелда-Карпа[6], модифицированной для задачи PCGTSP и дополненной стратегией отсечения (10). Как часто бывает в DP, алгоритм состоит из двух этапов.

1. Таблица поиска строится индуктивно, в прямом направлении, слой за слоем. Слой соответствует всем префиксам одной длины. Оптимальная стоимость для решаемой задачи находится после вычисления последнего m-го слоя.

2. Оптимальный маршрут реконструируется обратным просмотром на основе данных, хранящихся в таблице поиска.

Каждое состояние DP (запись в таблице поиска) соответствует частичному v-u-пути и индексируется кортежем $(V_1, \mathcal{C}', V_r, v, u)$, где V_1 и V_r – начальный и конечный кластеры маршрута, v и u – начальная и конечная его вершины ($v \in V_1$, $u \in V_r$), $\mathcal{C}'$ – *порядковый идеал* частично упорядоченного множества $\mathcal{C}$ и играет здесь роль, аналогичную префиксу σ в первой версии алгоритма. По определению идеала $\forall V \in \mathcal{C}', V' \in \mathcal{C}\ ((V', V) \in A) \Rightarrow (V' \in \mathcal{C}')$, поэтому V_1 принадлежит произвольному идеалу $\mathcal{C}' \subset \mathcal{C}$. Пусть $\mathfrak{I}_k$ – подмножество идеалов одного размера $k \in \{1, \ldots, m\}$. Очевидно, $\mathfrak{I}_1 = \{\{V_1\}\}$, а значит первый слой $\mathcal{L}_1$ таблицы поиска строится тривиально. Индуктивное построение остальных слоев описано в Алгоритме 1.

В данной диссертационной работе для повышения быстродействия мы вычисляем оценку L_3 на шаге 9 только для небольшого количества состояний ($\approx 1\%$) с наименьшей нижней границей, что позволяет повысить нижнюю оценку на $\approx 10\%$.

По построению, размер таблицы поиска $O(n^2 m \cdot |\mathfrak{I}|)$. Значит, время работы нашего алгоритма $O(n^3 m^2 \cdot |\mathfrak{I}|)$. В частности, в случае частичного порядка фиксированной *ширины*[7] w, $|\mathfrak{I}| = O(m^w)$ и значит, оптимальное решение PCGTSP может быть найдено в этом случае за полиномиальное время даже без применения отсечения на шаге 10.

Для оценки производительности предложенных алгоритмов использовалась общедоступная библиотека PCGTSPLIB[8]: В качестве базы сравнения использовалось решение, полученное решателем Gurobi. Во всех случаях для теплого старта, всем алгоритмам предоставлено одно и то же допустимое решение, полученное эвристическим решателем PCGLNS. В качестве критерия остановки использовалось понижение разрыва ниже 5%, где разрыв определяется по

[6] *Held M.* A Dynamic Programming Approach to Sequencing Problems / M. Held, R. M. Karp // Journal of the Society for Industrial and Applied Mathematics. 1962. Т. 10, № 1. С. 196—210. URL: `http://www.jstor.org/stable/2098806`.

[7] *Steiner G.* On the complexity of dynamic programming for sequencing problems with precedence constraints / G. Steiner // Annals of Operations Research. 1990. Т. 26, № 1. С. 103—123.

[8] *Salman R.* Branch-and-bound for the Precedence Constrained Generalized Traveling Salesman Problem / R. Salman, F. Ekstedt, P. Damaschke // Operations Research Letters. 2020. Т. 48, № 2. С. 163—166.

Алгоритм 1 DP :: индуктивное построение таблицы поиска

Вход: орграф G, частичный порядок Π, слой таблицы поиска $\mathcal{L}_k$
Выход: $(k+1)$-ый слой $\mathcal{L}_{k+1}$

```
1: инициализация L_{k+1} = ∅
2: for all C' ∈ J_k do
3:    for all кластер V_l ∈ C \ C', s.t. C' ∪ {V_l} ∈ J_{k+1} do
4:       for all v ∈ V_1 и u ∈ V_l do
5:          if есть состояние S = (C', U, v, w) ∈ L_k, s.t. (w, u) ∈ E then
6:             создаем новое состояние S' = (C' ∪ {V_l}, V_l, v, u)
7:             S'[cost] = min{S[cost] + c(w, u): S = (C', U, v, w) ∈ L_k}
8:             S'[pred] = arg min{S[cost] + c(w, u): S = (C', U, v, w) ∈ L_k}
9:             S'[LB] = S'[cost] + max{L_1, L_2, L_3}
10:            if S'[LB] ⩽ UB then
11:               добавляем S' к L_{k+1}
12:            end if
13:         end if
14:      end for
15:   end for
16: end for
17: return L_{k+1}
```

формуле

$$gap = \frac{UB - LB}{LB}.$$

Полученные результаты эксперимента представлены в табл. 3, которая организована следующим образом: первая группа столбцов описывает задачу, включая её обозначение ID, количество вершин n и кластеров m, а также стоимость стартового решения UB, полученного эвристикой PCGLNS. Затем следуют три группы столбцов для решателя Gurobi и двух предлагаемых алгоритмов. Каждая группа содержит время счета в секундах, наилучшее значение нижней границы LB и наилучший разрыв gap в процентах. Задачи, в которых один из предлагаемых алгоритмов сработал лучше Gurobi, выделены жирным шрифтом.

Для 13 из 39 задач (33%) один из предложенных алгоритмов показал лучшую производительность (либо в точности решения, либо во времени счета), в частности задачи *ft70.1* и *kro124p.3*, где новые алгоритмы смогли найти допустимое решение, тогда как Gurobi не смог этого сделать. Для некоторых задач стартовое решение, полученное эвристикой PCGLNS оказалось настолько близким к оптимальному, что исследуемые алгоритмы завершают работу почти немедленно.

В **четвёртой главе** рассматривается методология использования алгоритмов решения разных классов задачи резки в существующих CAD/CAM-

Таблица 3

Сравнение производительности алгоритмов решения задачи PCGTSP

Задача					*Gurobi*			*Ветвей и границ*			*DP*		
№	*ID*	*n*	*m*	*UB*	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*
1	br17.12	92	17	43	107.28	43	0.00	**11.2**	43	**0.00**	27.3	43	0.00
2	ESC07	39	8	1730	0.07	1730	0.00	1.3	1726	0.23	8.37	1730	0.00
3	ESC12	65	13	1390	0.52	1390	0.00	4.3	1385	0.36	14.99	1390	0.00
4	ESC25	133	26	1418	4.45	1383	2.53	**32**	1383	**0.00**	60.69	1383	0.00
5	ESC47	244	48	1399	52.01	1063	31.61	36000	980	42.76	36000	981	42.61
6	ESC63	349	64	62	380.65	62	0.00	1.3	62	0.00	**0.52**	62	**0.00**
7	ESC78	414	79	14832	43200	14581	1.72	1.3	14594	1.63	**0.68**	14594	**1.63**
8	ft53.1	281	53	6207	42099	6022	2.96	36000	4839	28.27	36000	4839	28.27
9	ft53.2	274	53	6653	42137	6184	7.58	36000	4934	34.84	36000	4940	34.68
10	ft53.3	281	53	8446	42194	6936	21.77	36000	5465	54.55	36000	5465	54.55
11	ft53.4	275	53	11822	23239	11822	0.00	35865	11274	4.86	2225	11290	4.71
12	ft70.1	346	70	32848	Истечение времени			36000	31153	5.44	36000	31177	**5.36**
13	ft70.2	351	70	33486	42021	31840	5.05	36000	31268	7.09	36000	31273	7.08
14	ft70.3	347	70	35309	41173	32944	7.18	36000	32180	9.72	36000	32180	9.72
15	ft70.4	353	70	44497	41827	41378	7.53	36000	38989	14.13	36000	41640	**6.86**
16	kro124p.1	514	100	33320	34162	29926	11.34	36000	27869	19.56	36000	27943	19.24
17	kro124p.2	524	100	35321	35379	30101	17.34	36000	28155	25.45	36000	28155	25.45
18	kro124p.3	534	100	41340	Истечение времени			36000	28406	45.53	36000	28406	**45.53**
19	kro124p.4	526	100	62818	41035	46704	34.50	36000	38137	64.72	36000	38511	63.12
20	p43.1	203	43	22545	43150	22327	0.98	36000	738	2954.88	36000	788	2761.04
21	p43.2	198	43	22841	43132	22381	2.05	36000	749	2949.53	36000	877	2504.45
22	p43.3	211	43	23122	43058	22540	2.57	36000	898	2474.83	36000	906	2452.10
23	p43.4	204	43	66857	43193	45396	47.26	4470	66846	0.00	**333.02**	66846	**0.00**
24	prob.100	510	99	1474	38567	800	80.25	36000	632	133.23	36000	632	133.23
25	prob.42	208	41	232	1292.1	202	14.85	36000	149	55.70	36000	153	51.63
26	rbg048a	255	49	282	64.32	282	0.00	0.9	272	3.68	0.25	272	3.68
27	rbg050c	259	51	378	**26.46**	378	**0.00**	**0.2**	372	**1.61**	0.25	372	1.61
28	rbg109a	573	110	848	83.23	848	0.00	2407	812	4.43	682	809	4.82
29	rbg150a	871	151	1415	**29095**	1414	**0.07**	**0.4**	1353	**4.58**	0.53	1353	4.58
30	rbg174a	962	175	1644	5413.8	1641	0.18	0.4	1568	4.85	0.67	1568	4.85
31	rbg253a	1389	254	2376	43159	2369	**0.13**	**0.8**	2269	**4.72**	1.42	2269	4.72
32	rbg323a	1825	324	2547	40499	2533	**0.55**	**2**	2448	**4.04**	3.59	2448	4.04
33	rbg341a	1822	342	2101	30687	2064	1.41	36000	1840	14.18	36000	1840	14.18
34	rbg358a	1967	359	2080	32215	2021	2.38	36000	1933	7.60	36000	1933	7.60
35	rbg378a	1973	379	2307	42279	2231	2.38	36000	2032	13.53	36000	2031	13.59
36	ry48p.1	256	48	13135	43141	12125	8.33	36000	10739	22.31	36000	10764	22.03
37	ry48p.2	250	48	13802	43033	12130	13.78	36000	10912	26.48	36000	11000	25.47
38	ry48p.3	254	48	16540	43102	13096	26.30	36000	11732	40.98	36000	11822	39.91
39	ry48p.4	249	48	25977	43057	22266	16.67	18677	25037	3.75	**14001**	25043	**3.73**

системах на примере САПР «Сириус». Поскольку требуется обеспечить совместную работу программного обеспечения, разработанного в разное время разными командами разработчиков, чрезвычайно важными становятся вопросы организации эффективных программных интерфейсов.

Например, для хранения и обмена геометрической информацией в САПР «Сириус» используется унаследованный двоичный формат DBS [11], который обладает важными достоинствами:

– эффективное хранение больших данных за счёт хранения массивов вещественных чисел в формате IEE 754;

– возможность добавления новых типов записей для хранения ранее не предусмотренной информации; расширяемость формата

– механизм создания копий деталей и геометрических преобразований над ними.

В то же время, работа с ним сопряжена с рядом сложностей, прежде всего:

– Сложность чтения двоичного формата, особенно в некоторых языках программирования

– Структура DBS-файла, предназначенная для эффективного хранения, сильно отличается от удобного внутреннего представления геометрии; требуется нетривиальное преобразование при чтении файла

– Формат DBS создавался в том числе для экономии памяти, как дисковой, так и оперативной, что более неактуально; отказ от этого позволяет резко упростить процедуры экспорта–импорта.

В рамках данной диссертационной работы в целях упрощения взаимодействия различных подсистем было принято решение использовать по возможности открытые текстовые форматы для хранения и передачи данных. В качестве основного формата был выбран формат JavaScript Object Notation (JSON[9]), ввиду того, что он с одной стороны имеет готовые библиотеки для чтения и записи для практически всех современных языков программирования, является стандартом де-факто во многих современных приложениях для обмена данными, довольно прост, настолько, что может например, формироваться даже без использования специализированных библиотек, но при этом достаточно выразителен.

Для хранения и обмена информацией о геометрии деталей и раскройной карты была разработана наиболее простая схема JSON, не включающая сложности с копиями деталей и геометрическими преобразованиями. Пример такого файла для простейшей раскройной карты приведён в Листинге 1.

[9]Introducing JSON. URL: `https://www.json.org/`.

```
[{
  "partid": "LIST",
  "paths": [
  [
  [0, 0, 0],
  [0, 500, 0],
  [700, 500, 0],
  [700, 0, 0],
  [0, 0, 0]]
]},
{
  "partid": "RING",
  "paths": [
  [
  [205, 405, -1],
  [205, 5, -1],
  [205, 405, 0]],
  [
  [205, 305, 1],
  [205, 105, 1],
  [205, 305, 0]]
]}]
```

Листинг 1. JSON-файл с геометрией простейшей раскройной карты

Использование JSON в качестве формата обмена данными оказалось удобным на практике, поэтому позднее были разработаны другие форматы файлов, в частности задания на резку и результатов резки. Все они были формально описаны в виде JSON-схемы[10], см. [12].

Использование открытых форматов файлов позволило также значительно упростить вопросы, связанные с визуализацией обрабатываемой информации. Традиционно для этого приходится писать отдельный код, имеющий зачастую довольно сложную структуру и решающий множество задач, включая геометрические расчёты и организацию пользовательского интерфейса. В рамках данной диссертационной работы в качестве средства визуализации использовался экспорт в формат Scalable Vector Graphics (SVG), который является стандартом де-факто, содержит богатые возможности визуализации и широко поддержан всеми современными браузерами. Листинг 2 показывает пример простейшего SVG-файла, сгенерированного для раскройной карты, представленной на Листинге 1.

Переход к широкому использованию SVG позволяет также использовать зрелые современные технологии каскадных таблиц стилей (Cascading Style Sheets, CSS) для управления внешним видом визуализации (включая цвета, заливки и штриховки и анимацию) и язык JavaScript для добавления к визуализации элементов интерактивности. Один из вариантов оформления SVG-файла из Листинга 2 приведён на рис. 6. Пользовательский интерфейс (масштаби-

[10] *Pezoa F.* Foundations of JSON Schema / F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, D. Vrgoč // WWW '16: Proceedings of the 25th International Conference on World Wide Web. Republic, Canton of Geneva, CHE : International World Wide Web Conferences Steering Committee, 2016. С. 263—273.

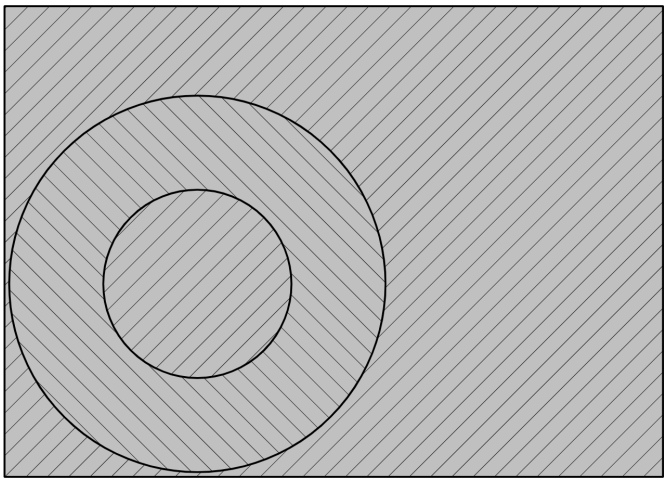

Рис. 6. Визуализация раскроя из Листинга 1

рование и прокрутка) обеспечивался при помощи подключения библиотеки с открытым кодом svg-pan-zoom[11].

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <svg
3     xmlns="http://www.w3.org/2000/svg"
4  ><g><g transform = "scale(1, -1)">
5  <path name="LIST" d="M 0 0
6  V 500
7  H 700
8  V 0
9  H 0 Z"/>
10 <path name="RING" d="M 205 405
11 A 200 200 0 0 0 405 205 A 200 200 0 0 0 205 5
12 A 200 200 0 0 0 5 205 A 200 200 0 0 0 205 405 Z
13 M 205 305
14 A 100 100 0 0 1 105 205 A 100 100 0 0 1 205 105
15 A 100 100 0 0 1 305 205 A 100 100 0 0 1 205 305 Z"/>
16 </g></g></svg>
```

Листинг 2. SVG-файл для визуализации раскроя

В ходе диссертационной работы была разработан пакет утилит [10], обеспечивающих конвертацию между различными форматами файлов (включая DBS, JSON, YAML, DXF и SVG для визуализации). Для визуализации решения задачи PCGTSP (на основе комбинации информации, полученной из нескольких источников), была разработана специализированная утилита [9], первоначально в форме утилиты командной строки но позднее преобразованная для удобства использования в Single Page Application (SPA). Пример созданного ею изображения приведён на рис. 7.

В **заключении** сформулированы основные научные и практические результаты диссертационной работы.

В **приложениях** приведены описание формата файла DBS и JSON-схемы разработанных в ходе работы форматов файлов.

[11] JavaScript library that enables panning and zooming of an SVG in an HTML document, with mouse events or custom JavaScript hooks. URL: `https://github.com/bumbu/svg-pan-zoom`.

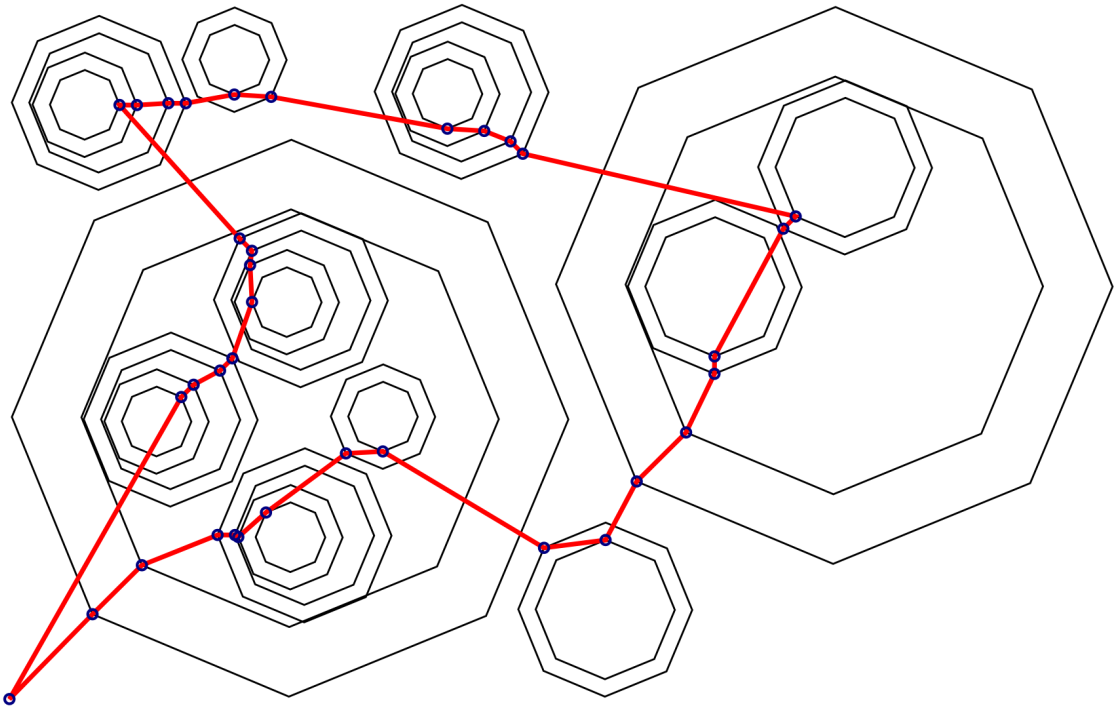

Рис. 7. Пример визуализации решения задачи PCGTSP

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

В соответствии с целью и задачами исследования получены следующие научные и практические результаты:

1. Разработана схема эффективного учёта ограничений предшествования для задачи непрерывной резки.

2. Разработана основанная на геометрических соображениях эвристика размещения точек врезки на плоских контурах.

3. Доказано, что данная эвристика доставляет локальный минимум длины холостого хода и сформулированы два набора достаточных условий того, что полученное решение является глобальным минимумом.

4. Эти две схемы могут соединяться с различными алгоритмами дискретной оптимизации, тем самым получается полный алгоритм решения задачи непрерывной резки. В данной работе для дискретной оптимизации используется метод переменных окрестностей.

5. Полученный алгоритм даёт решения хорошего качества за разумное время. В случае, когда известно точное решение соответствующей обобщённой задачи коммивояжера, они визуально совпадают, но длина холостого хода для решения задачи непрерывной резки оказывается короче за счёт отсутствия дискретизации.

6. Продемонстрировано, что полученный алгоритм может использоваться для решения задач сегментной непрерывной резки (SCCP) и обобщённой сегментной непрерывной резки (GSCCP), тем самым открывая подход к решению общей задачи прерывистой резки (ICP)

7. Разработан алгоритм Branch-and-Bound для точного решения обобщённой задачи коммивояжёра с ограничениями предшествования, рассчитывающий нижнюю границу

8. Алгоритм способен находить точные решения для задач большего размера, чем другие алгоритмы. В проведённых экспериментах было найдено решение для задачи со 151 кластером.

9. Данный алгоритм также решает важную задачу оценки качества решений, полученных другими алгоритмами, даже в случае, когда нахождение точного решения непрактично.

10. Алгоритм может быть реализован в классической схеме, а также в парадигме динамического программирования. В последнем случае он естественным образом допускает распараллеливание и демонстрирует лучшую производительность.

11. Разработаны форматы данных для обмена геометрической и маршрутной информацией и визуализации для использования в CAD/CAM-системах, а также алгоритмы преобразования на примере САПР «Сириус». Аналогичные конвертеры могут легко разрабатываться для других САПР.

12. Разработано программное обеспечение для реализации всех алгоритмов на языках C, Python и JavaScript.

Перспективы дальнейшей разработки темы. Можно выделить следующие направления дальнейшего развития и совершенствования алгоритмического и программного обеспечения САПР УП для оборудования листовой фигурной резки с ЧПУ:

1. Учёт дополнительных ограничений в алгоритме решения задачи непрерывной резки, в частности того, что точка врезки располагается не на самом контуре, а на некотором расстоянии от него, а также того, что существуют зоны, где не могут размещаться зоны врезки, а также других технологических ограничений, порождаемых современным оборудованием термической резки с ЧПУ

2. Использование других алгоритмов дискретной оптимизации в задаче непрерывной резки и оценка производительности и качества получаемых алгоритмов.

3. Разработка других методов получения оценок для частичных подзадач GTSP с целью повышения нижней границы; например, за счёт более точного учёта расстояний между точками, а не только между кластерами.

СПИСОК ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

1. *Petunin A. A.* Optimum routing algorithms for control programs design in the CAM systems for CNC sheet cutting machines / A. A. Petunin, P. A. Chentsov, E. G. Polishchuk, S. S. Ukolov, V. V. Martynov // Proceedings of the X All-Russian Conference «Actual Problems of Applied Mathematics and Mechanics» with International Participation, Dedicated to the Memory of Academician A.F. Sidorov and 100th Anniversary of UrFU: AFSID-2020. — American Institute of Physics Inc., 2020. — С. 020005.
2. *Petunin A. A.* On the new Algorithm for Solving Continuous Cutting Problem / A. A. Petunin, E. G. Polishchuk, S. S. Ukolov // IFAC-PapersOnLine. — 2019. — Т. 52, № 13. — С. 2320—2325.
3. *Petunin A. A.* About some types of constraints in problems of routing / A. A. Petunin, E. G. Polishuk, A. G. Chentsov, P. A. Chentsov, S. S. Ukolov // AIP Conference Proceedings. — 2016. — Т. 1789, № 1. — С. 060002.
4. *Petunin A. A.* The termal deformation reducing in sheet metal at manufacturing parts by CNC cutting machines / A. A. Petunin, E. G. Polyshuk, P. A. Chentsov, S. S. Ukolov, V. I. Krotov // IOP Publishing. — 2020. — Т. 613. — С. 012041.
5. *Petunin A.* Library of Sample Image Instances for the Cutting Path Problem / A. Petunin, A. Khalyavka, M. Khachay, A. Kudriavtsev, P. Chentsov, E. Polishchuk, S. Ukolov // Pattern Recognition. ICPR International Workshops and Challenges, 2021, Proceedings. — Berlin, Germany : Springer, 2021. — С. 227—233.
6. *Petunin A.* A Novel Algorithm for Construction of the Shortest Path Between a Finite Set of Nonintersecting Contours on the Plane / A. Petunin, E. Polishchuk, S. Ukolov // Advances in Optimization and Applications. — Cham, Switzerland : Springer, 2021. — С. 70—83.
7. *Tavaeva A.* A Cost Minimizing at Laser Cutting of Sheet Parts on CNC Machines / A. Tavaeva, A. Petunin, S. Ukolov, V. Krotov // SpringerLink. — 2019. — С. 422—437.
8. *Петунин А. А.* Новый алгоритм построения кратчайшего пути обхода конечного множества непересекающихся контуров на плоскости / А. А. Петунин, Е. Г. Полищук, С. С. Уколов // Известия ЮФУ. Технические науки. — 2021. — № 1. — С. 149—164.

9. *Уколов С. С.* Визуализация решения задачи PCGTSP / С. С. Уколов. — URL: https://ukoloff.github.io/j2pcgtsp/.

10. *Уколов С. С.* Конвертеры открытых форматов для САПР «Сириус» / С. С. Уколов. — URL: https://github.com/ukoloff/dbs.js.

11. *Уколов С. С.* Описание формата DBS / С. С. Уколов, В. И. Кротов. — URL: https://github.com/ukoloff/dbs.js/wiki/DBS.

12. *Уколов С. С.* JSON-схемы файлов, используемых в САПР «Сириус» / С. С. Уколов, П. А. Ченцов. — URL: https://ukoloff.github.io/dbs.js/json-schema/.