Федеральное государственное автономное образовательное учреждение
высшего образования «Уральский федеральный университет имени первого
Президента России Б.Н. Ельцина»
Институт новых материалов и технологий
Кафедра информационных технологий и автоматизации проектирования

На правах рукописи

Уколов Станислав Сергеевич

Разработка алгоритмов оптимальной маршрутизации инструмента для САПР управляющих программ машин листовой резки с ЧПУ

Специальность 05.13.12 —
Системы автоматизации проектирования (промышленность)

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, доцент
Петунин Александр Александрович

Екатеринбург — 2021

Оглавление

Введение

Актуальность темы исследования. Современное производство предъявляет высокие требования к качеству заготовок и 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 трудоемкость и экономичность изготовления деталей. 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 оборудования 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, здесь возникает известная задача оптимизации фигурного раскроя листового материала, которая относится к классу трудно решаемых проблем раскроя-упаковки (*Cutting & Packing*). На следующем этапе проектирования УП осуществляется процесс назначения маршрута резки — траектории перемещения режущего инструмента для полученного на первом этапе варианта раскроя, здесь возникают актуальные научно-практические задачи опти-

мизации маршрута режущего инструмента. Их целью 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 врезки в материал листа, а также направления движения резака 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 технологического 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. Зачастую пользователи САПР используют интерактивный режим проектирования УП. Кроме того, отсутствуют способы оценки точности полученных решений. В связи с этим актуальным направлением исследования являются вопросы разработки и применения эвристических и метаэвристических подходов, а также точных алгоритмов, которые позволяют получить решение задачи оптимальной маршрутизации режущего инструмента в режиме автоматического проектирования за приемлемое время и обеспечивают при этом эффективные оценки результатов проектирования.

Степень разработанности темы исследования. Методы проектирования технологических процессов раскройно-заготовительного производства исследовались в работах как отечественных так и зарубежных ученых. Хотя задача 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 ее 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, А.Ф. Таваева 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 подавляющем большинстве работ используется дискретизация граничных контуров деталей, что позволяет применять хорошо разработанные математические модели дискретной оптимизации. Можно отметить только отдельные публикации, где оптимизационные алгоритмы ориентированы на поиск решений в непрерывных множествах.

Задачи маршрутизации инструмента машин листовой резки с ЧПУ относятся, как известно, к NP-трудным задачам. Следует отметить, что до настоящего времени 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 УП для машин листовой резки с ЧПУ, есть отдельные модули, которые позволяют решать в автоматическом режиме *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 критериев стоимости и времени резки с учетом рабочего хода инструмента, затрат на врезку и т.д. К тому же, реализованные в коммерческом программном обеспечении алгоритмы как правило не описываются 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 с эффективными оценками точности

для нескольких подклассов задачи ICP мог бы заполнить явный существующий пробел в исследованиях. В частности, отметим, что актуальна разработка алгоритмов для подкласса Segment CCP, базирующегося на понятии «сегмента резки».

В целом можно отметить, что за рамками исследований отечественных и зарубежных коллег остаются следующие принципиальные моменты: разработка алгоритмов, обеспечивающих получение глобального оптимума оптимизационной задачи маршрутизации инструмента; учет тепловых искажений заготовок при термической резке с целевыми функциями стоимости и времени резки, что приводит к не технологичным решениям и искажению геометрии получаемых заготовок; рассмотрение задач маршрутизации из класса ICP, в частности, и задач с набором возможных точек врезки из континуального множества, включая мульти-контурные 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 алгоритмов и программного обеспечения для САПР управляющих программ машин листовой резки с ЧПУ остается доминантой развития методов решения исследуемой оптимизационной проблемы маршрутизации инструмента.

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 точный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP), позволяющий оценивать качество решений на основе вычисления нижней оценки

– Разработать эвристики поиска оптимального положения точек врезки в контур детали и последовательности обхода контуров в процессе решения задач непрерывной резки (CCP, SCCP)

– Разработать программное обеспечение, реализующее разработанные алгоритмы

– Разработать схемы информационного обмена и методику использования алгоритмов оптимальной маршрутизации режущего инструмента в CAD/CAM-системах при автоматическом проектировании управляющих программ машин листовой резки с ЧПУ.

Научная новизна результатов.

1. Разработан алгоритм ветвей и границ для обобщенной задачи коммивояжера с ограничениями предшествования PCGTSP, позволяющий строить нижние оценки для решений указанной задачи, в том числе, полученных другими алгоритмами и эвристиками. Этот алгоритм способен находить точные решения для задач значительно большей размерности, чем известные алгоритмы (до ≈ 150 кластеров в зависимости от уровня вложенности).

2. Разработаны алгоритм поиска точек врезки в контуры, не использующий механизм дискретизации, а также схема выбора последовательности резки контуров на основе метода переменных окрестностей, совместно решающие задачи CCP и SCCP.

3. Сформулированы схемы использования ограничений предшествования для уменьшения вычислительной сложности алгоритмов оптимальной маршрутизации, как в моделях дискретной, так и непрерывной оптимизации.

Теоретическая и практическая значимость работы

1. Разработанные алгоритмы могут применяться для автоматического проектирования УП машин листовой резки с ЧПУ. Для ряда задач впервые удалось получить эффективные оценки точности решений.

2. Использование непрерывных моделей оптимизации позволяет уменьшить длину холостого хода (в некоторых случаях — до 10%) по сравнению с используемыми в настоящее время дискретными моделями.

3. Разработанные алгоритмы могут применяться для решения более общей задачи маршрутизации резки, например обобщённой сегментной резки GSCCP.

4. Разработанные схемы информационного обмена, форматы файлов и методика использования алгоритмов оптимальной маршрутизации инструмента позволяют интегрировать разработанное программное обеспечение в существующие российские САПР «Сириус» и САПР «T-Flex»

5. Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ (государственный контракт № 075-03-2020-582/4).

6. Результаты исследований используются в учебном процессе ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина»

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. Точный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP) с обновлением нижней границы.

2. Эвристика поиска точек врезки в плоские контура, не использующая дискретизацию контура.

3. Достаточные условия, при которых полученный маршрут доставляет глобальный минимум длины холостого хода инструмента.

4. Форматы файлов и схемы 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* (AMEE'16), Созополь, Болгария, 08.06.2016 – 13.06.2016.

– *Manufacturing, Modelling, Management & Control*, (8th MiM 2016) Труа, Франция, 28.06.2016 – 30.06.2016.

– *ASRTU 2017 International Conference on Intellectual Manufacturing*, Харбин, Китайская Народная Республика, 15.06.2017 – 18.06.2017.

– *Mathematical Optimization Theory And Operations Research* (MOTOR 2019), Екатеринбург, Россия, 08.07.2019 – 12.07.2019.

– *Manufacturing Modelling, Management and Control*, (9th MiM 2019) Берлин, Германия, 28.08.2019 – 30.08.2019.

– *X Всероссийская конференция «Актуальные проблемы прикладной математики и механики»* с международным участием, посвященная памяти

академика А.Ф.Сидорова и 100-летию Уральского федерального университета, пос. Абрау-Дюрсо, Россия, 01.09.2020 – 06.09.2020.

– *ICPR International Workshops and Challenges Virtual Event*, Milan, Italy, 10.01.2021 – 15.01.2021.

– *XVI Всероссийская научно-практическая конференция «Перспективные системы и задачи управления»*, Домбай, Россия, 5.04.2021 – 9.04.2021.

– *XII International Conference Optimization and Applications* (OPTIMA-2021), Petrovac, Черногория, 27.09.2021 – 1.10.2021.

– *XIV-я Всероссийская Мультиконференция по проблемам управления*, с. Дивноморское, Геленджик, Россия, 27.09.2021 – 02.10.2021.

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 9 научных работ в рецензируемых научных журналах, определенных ВАК РФ и Аттестационным советом УрФУ, из них 8 публикаций проиндексировано в международных базах данных WoS и Scopus.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения и 4 приложений. Полный объём диссертации составляет 135 страниц, включая 24 рисунка и 7 таблиц. Список литературы содержит 121 наименование.

Глава 1. Задача оптимизации маршрута режущего инструмента для машин листовой резки с ЧПУ. Анализ современного состояния проблемы исследования

В машиностроении, производстве металлоконструкций и многих других отраслях промышленности значительная часть продукции производится из заготовок, получаемых из листовых материалов. В процессе планирования производства на предприятиях используются отечественные и зарубежные системы автоматизированного проектирования (САПР), предназначенные для разработки управляющих программ (УП) для машин листовой резки с ЧПУ (числовым программным управлением). Хотя они позволяют автоматизировать разработку УП, на зачастую не способны решать оптимизационные задачи. По этой причине пользователям САПР в процессе проектирования маршрута инструмента иногда все еще приходится использовать интерактивные методы проектирования УП, поскольку алгоритмы автоматической генерации УП во многих случаях не позволяют генерировать оптимальные управляющие программы, а также обеспечивать соблюдение технологических требований листовой резки.

В качестве критериев оптимизации обычно используются длительность процесса листовой резки и его стоимость. Проблема разработки методов, алгоритмов и соответствующего программного обеспечения, позволяющих оптимизировать параметры процесса резки заготовок из листовых материалов на машинах с ЧПУ в неинтерактивном режиме, включая сюда также алгоритмы маршрутизации движения инструмента, которые бы обеспечивали минимизацию времени резки и стоимости процесса, остается актуальнейшей задачей для раскройно-заготовительного производства.

1.1. Основные понятия

Проектирование управляющих программ для технологического оборудования термической резки — это сложный, многоступенчатый процесс, в котором можно выделить по крайней мере следующие этапы:

1. Геометрическое моделирование и кодирование геометрии деталей / заготовок

2. Разработка раскройной карты листового материала,

3. Проектирование маршрута движения режущего инструмента по раскройной карте с учетом технологических ограничений оборудования

4. Собственно генерирование управляющей программы для конкретного вида станка с ЧПУ [112]

Хотя вопросы, связанные с разработкой раскройной карты [86; 108] (или говоря кратко — *раскроем*), находятся вне темы данной диссертационной работы, тем не менее, невозможно не упомянуть значительный вклад советских и российских исследователей в теорию оптимизации раскроя. Работы в этой предметной области были начаты выдающимися учёными В.А. Залгаллером и Л.В. Канторовичем [98] и продолжены в уфимской научной школе Э.А. Мухачевой и ее учениками: А.Ф. Валеевой, М.А. Верхотуровым, В.М. Картаком, В.В. Мартыновым, А.С. Филипповой, и др., см. например [101; 102].

В процессе раскроя порождается графический файл, содержащий информацию о геометрии и расположении вырезаемых заготовок, называемый «*раскройная карта*». Каждая деталь на листе описывается при помощи внешнего замкнутого контура и возможно одним или несколькими внутренними, их расположение фиксировано в процессе раскроя и служит входными данными для следующего этапа — маршрутизации режущего инструмента (или коротко — *резки*).

Рассмотрим круг понятий, связанных с *маршрутом инструмента* (маршрутом резки) применительно к современным технологиям фигурной листовой резки [110]. В промышленном производстве единичного и мелкосерийного типа для раскроя листовых материалов используются в основном такие технологии, как лазерная, плазменная, газовая и гидроабразивная. Целесообразность их применения определяется разнообразными технологическими факторами, включая свойства материала, экономические требованиями к процессу резки, технологические требованиями к качеству реза и пр.

Современные технологии резки предполагают как правило, что для сохранения требуемой геометрии заготовки траектория движения режущего инструмента не совпадает с граничным контуром заготовки, а проходит на некотором расстоянии от него (по *эквидистанте* контура, см. рис. 1.1), которое как правило составляет величину равную половине ширины реза. Это вызвано тем, что часть материала вырезается («сгорает», «вымывается» и т.п.) в процессе резки. Ширина же реза определяется выбранной технологией резки, толщиной

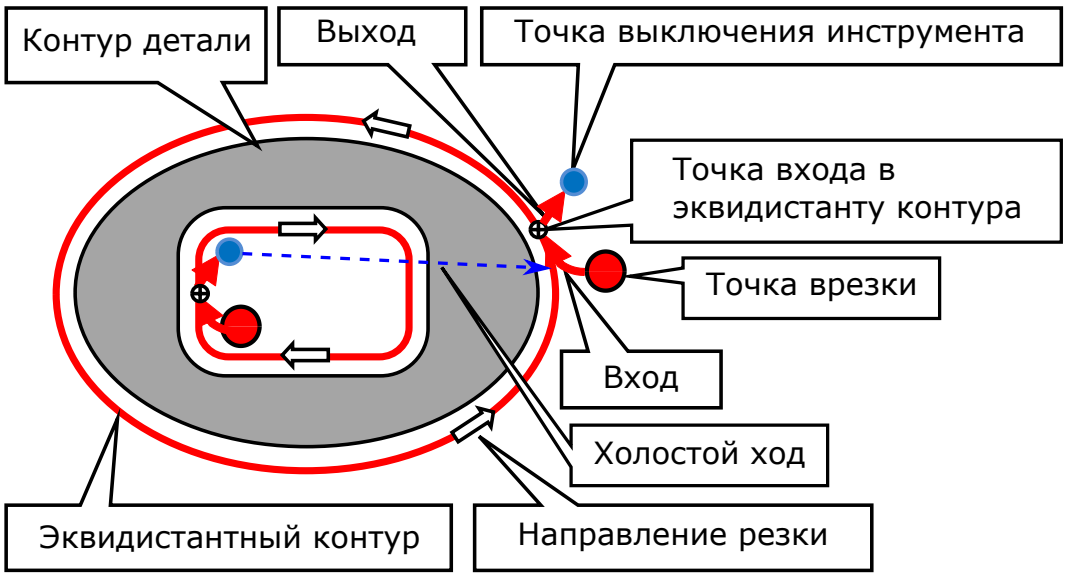


Рис. 1.1. Элементы маршрута резки

и маркой материала, заданной скоростью резки и особенностями конкретного технологического оборудования, используемого для резки.

Следующая особенность листовой резки — необходимость предварительной *врезки* (пробивки) материала перед началом резки непосредственно по эквидистантному контуру. Пробивка материала сопровождается дополнительными деформациями материала, поэтому производится на еще большем расстоянии от контура заготовки, чем дистанция до эквидистантного контура (за исключением случаев, когда отверстия для точек врезки в листовом материале механическим способом готовятся, например, просверливаются). Иногда врезка может осуществляться непосредственно на границе материала («врезка с края листа»), что позволяет значительно сократить деформации материала и время на врезку.

Если используется *стандартная техника резки*, то каждый замкнутый контур вырезается целиком, и переход к следующей точке врезки происходит с выключенным инструментом на *холостом ходу*. При этом точка выключения инструмента в общем случае также не совпадает с точкой входа в эквидистантный контур, по которому осуществлялась резка, и также, как и точка врезки, может лежать вне эквидистантного контура. Во многих случаях программирование точки выключения инструмента допускается непосредственно на эквидистантном контуре.

Стратегия минимизации тепловых деформаций при термической резке и требования к качеству реза порождают необходимость управления не только

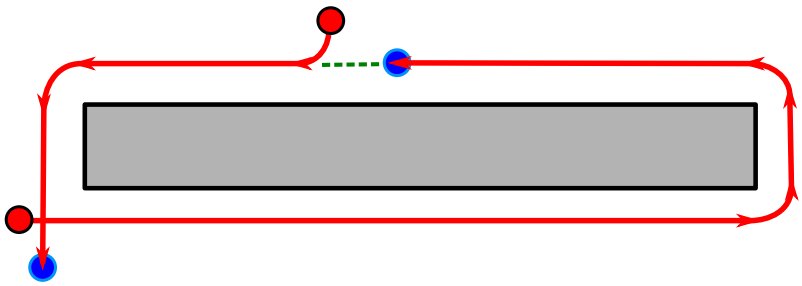
выбором точек врезки, но и траекториями подхода к контуру (lead-in) и выхода из контура (lead-out). В зависимости от конкретных условий (вида термической резки, марки и толщины материала, скорости резки, геометрической формы контура и пр.) подход к контуру может осуществляться по дуге окружности, касательная к которой совпадает с касательной к контуру в точке входа, либо по прямой линии. Аналогично, после завершения резки выход из контура также может осуществляться с включенным инструментом либо по дуге, либо по прямой линии. Необходимость этого может быть вызвана тем, что в точке выключения инструмента может возникнуть «вырыв» или оплавление части материала, что приводит к искажению геометрии детали. Уменьшение эффекта деформации заготовок может достигаться специальными техниками резки, например, врезкой в «угловые» точки заготовок.

На рис. 1.1 на стр. 15 представлены основные элементы маршрута резки:

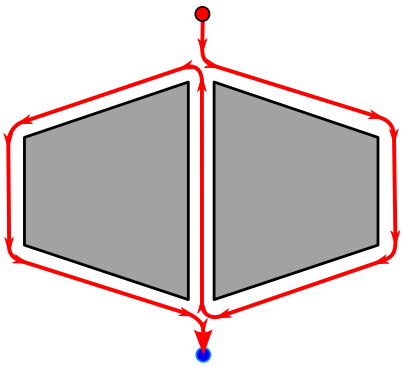
- Точка врезки
- Вход в контур (lead-in)
- Точка входа в эквидистанту контура
- Собственно траектория резки по эквидистанте контура детали
- Выход из контура (lead-out)
- Точка выключения инструмента
- Холостой путь до следующей точки врезки (как правило по прямой)

Проектирование маршрута резки заключается в выборе этих элементов, а также определении последовательности обработки контуров. Задача поиска оптимальной (в соответствии с некоторым критерием или группой критериев) траектории перемещения режущего инструмента является одной из наиболее важных задач, возникающих при проектировании УП в системах автоматизированного проектирования для оборудования термической листовой резки с ЧПУ.

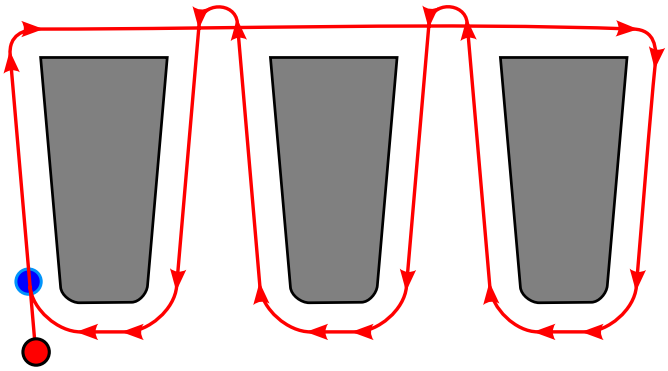
В настоящее время кроме *стандартной техники резки* «по замкнутому контуру» (показанной на рис. 1.1) при проектировании маршрута резки широко используются так называемые *нестандартные техники резки* (см. рис. 1.2). Все техники фигурной резки на машинах с ЧПУ можно разделить на 3 класса [109]:



а) Резка с перемычкой



б) «Восьмерка»



в) «Змейка»

Рис. 1.2. Примеры нестандартных техник резки

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

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

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

В качестве примеров мультисегментной резки можно привести резку с перемычкой (рис. 1.2а), а мультиконтурной — «восьмерку» (рис. 1.2б), «змейку» (рис. 1.2в), «цепную резку», «мост», см. [110]. На практике используются и многие другие специальные техники резки, но все их можно отнести к тому или иному из этих классов.

1.2. Формализация общей задачи маршрутизации режущего инструмента

Для формального определения маршрута резки введем следующие обозначения [64; 110]. Пусть $A_1, A_2, \ldots, A_n$ — двумерные геометрические объек-

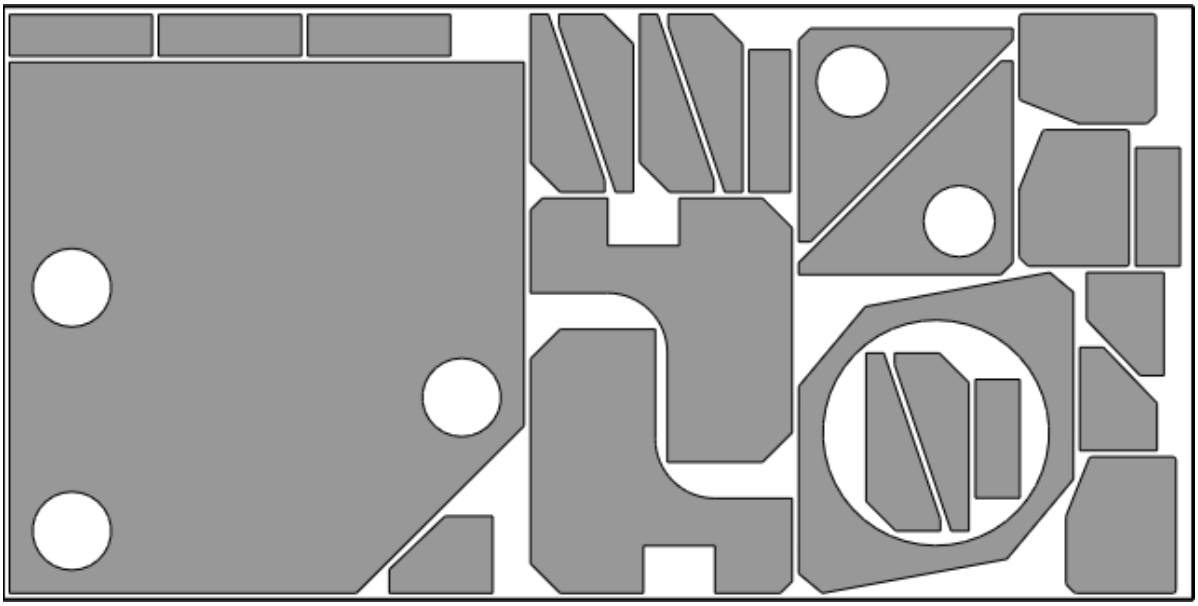


Рис. 1.3. Пример раскроя листа 2000 × 1000 мм с заданным минимальным расстоянием между деталями 10 мм

ты (точечные замкнутые множества), представляющие собой односвязные или многосвязные области евклидовой плоскости $\mathbb{R} \times \mathbb{R}$, ограниченные одной или несколькими замкнутыми кривыми (граничными контурами) $C_1, C_2, \ldots, C_N$ $(A_i, C_J \subset \mathbb{R} \times \mathbb{R}; i \in \overline{1,n}; j \in \overline{1,N}; N \geqslant n)$. Объекты $A_1, A_2, \ldots, A_n$ являются геометрическими моделями плоских заготовок / деталей.

Пусть также определена область размещения объектов $\mathcal{B} \subset \mathbb{R} \times \mathbb{R}$, которая является геометрической моделью листового материала, из которого вырезаются детали. Как правило, это прямоугольник, но иногда на практике могут встречаться более сложные случаи.

Будем полагать, что раскройная карта зафиксирована, при этом выполнены условия взаимного непересечения объектов. Полагаем также, что выполнены другие дополнительные условия, обусловленные технологическими требованиями резки деталей на конкретном технологическом оборудовании с ЧПУ, в частности, условие соблюдения необходимой ширины реза. Другими словами, фиксированный вариант размещения объектов является допустимым вариантом раскроя листового материала для заданного набора n деталей.

Пример размещения в прямоугольной области 24 объектов ($n = 24$), описываемых 30 замкнутыми контурами ($N = 30$) с заданным минимальным расстоянием между объектами, приведен на рис. 1.3.

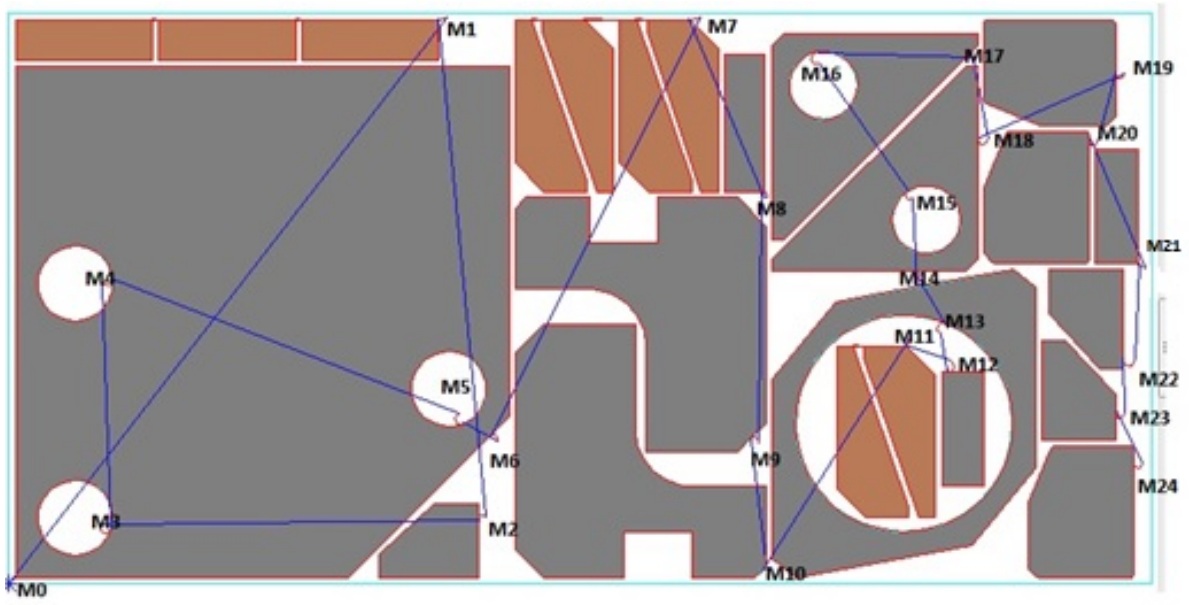


Рис. 1.4. Пример маршрута резки, содержащего 24 сегмента резки

Определение 1.1. **Сегмент резки** $S = MM^*$ — это траектория рабочего хода инструмента между точкой врезки M и соответствующей ей точкой выключения инструмента M^*.

Геометрически сегмент резки представляет собой определенную на евклидовой плоскости $\mathbb{R}\times\mathbb{R}$ кривую. ($S \subset \mathbb{R}\times\mathbb{R}; M = (x, y) \in \mathbb{R}\times\mathbb{R}, M^* = (x^*, y^*) \in \mathbb{R}\times\mathbb{R}$). Сегмент содержит в себе вход в контур (*lead-in*) и выход из него (*lead-out*), а также часть или целый контур детали или нескольких деталей. В случае стандартной техники резки имеется однозначное соответствие между контурами деталей и сегментами резки, но в общем случае оно отсутствует.

Предположим, что для вырезки деталей было использовано K сегментов резки $S_k = M_k M_k^*; k \in \overline{1, K}$. Тогда маршрут резки деталей можно определить в терминах сегментов резки как кортеж

$$\mathfrak{R} = \langle M_0, M_1, S_1, M_1^*, M_2, S_2, M_2^*, \ldots, M_K, S_K, M_K^*, i_1, i_2, \ldots, i_K \rangle , \qquad (1.1)$$

где M_0 – начальная точка положения инструмента, $i_1, i_2, \ldots, i_K$ – последовательность, в соответствии с которой вырезаются используемые сегменты резки $S_1, S_2, \ldots, S_K$. Линейное перемещение инструмента на холостом ходу между точкой выключения инструмента и следующей точкой врезки однозначно определяется этой последовательностью.

На рис. 1.4 показана схема одного из возможных маршрутов резки для примера, приведенного на рис. 1.3. Маршрут содержит 24 сегмента. Для резки

внешних контуров трех групп деталей с точками врезки M_1 (три детали в группе), M_7 (четыре детали в группе) и M_{11} была использована мультиконтурная резка. Все остальные контуры вырезаны с применением стандартной техники резки. Последовательность резки сегментов соответствует номерам точек врезки M_j $(j = 1, 2, \ldots, 24)$.

Визуализация траектории инструмента на рис. 1.4 осуществляется точно по граничным контурам деталей, а не по эквидистантным контурам, в нарушение условия, что траектория реза должна отстоять от граничного контура. Это связано с тем, что в большинстве CAM-систем программирование движения инструмента первоначально осуществляется по граничным контурам деталей, а вычисление реальной траектории производится либо непосредственно самой системой ЧПУ, либо специальной программой-постпроцессором, предназначенной для конвертирования информации о маршруте резки из внутреннего формата системы в формат команд конкретного технологического оборудования с ЧПУ.

В дальнейшем мы будем полагать, что траектория инструмента в маршруте резки $\mathfrak{R}$ программируется по граничным контурам, и сегменты резки $S_k = M_k M_k^*; k \in \overline{1, K}$ содержат все граничные контуры деталей $C_1, C_2, \ldots, C_N$, т. е.

$$\bigcup_{j=1}^{N} C_j \subset \bigcup_{k=1}^{K} S_k$$

На рис. 1.5 на стр. 21 показан фрагмент управляющей программы (G-кода) для машины листовой газовой резки типа «Комета» с системой ЧПУ 2Р32М. Программа сгенерирована на основе маршрута резки (спроектированного в интерактивном режиме в CAD/CAM-системе «Сириус» и показанного на рис. 1.4) соответствующим постпроцессором со следующими числовыми параметрами резки:

- число строк в УП – 333;
- количество точек врезки – 24;
- путь инструмента на рабочей скорости – 27,36 м;
- путь инструмента на холостом ходу – 8,39 м;
- время движения на рабочей скорости – 62,04 мин;

```
%УП_2P32M_01 N1G91
N2G00X7662Y9909F6000
N3M70T1
N4M71T1
N5G01X-141Y-48F460
N6X-2400
N7X-40
N8X-67
N9X-2400
N10X-40
N11X-67
N12X-2400
N13Y-700
N14X2400
N15Y700
N16Y40
N17X107Y-40
N18Y-700
N19X2400
N20Y700
N21Y40
N22X107Y-40
N23Y-700
N24X2400
N25Y700
N26Y40
N27M74T1
N28G00X817Y-8745F6000
N29M71T1
N30G03X-108Y0I-54J0F460
N31G01Y-1048
N32X-1740
N33Y400
N34X940Y900
N35X800
N36Y-252
N37X20Y-30
...
N314M71T1
N315G02X-130Y0I-65J0F460
N316G01Y267
N317G03X-50Y50I-50J0
N318G01X-1366
N319G03X-46Y-31I0J-50
N320G01X-384Y-960
N321G03X-4Y-19I46J-19
N322G01Y-1120
N323G03X14Y-35I50J0
N324G01X122Y-121
N325G03X37Y-14I35J35
N326G01X1627Y-1
N327G03X50Y50I0J50
N328G01Y1933
N329X20Y30
N330M74T1
N331M75T1
N332M02
M30
```

Рис. 1.5. Фрагмент управляющей программы для машины листовой резки «Комета» с ЧПУ 2P32M

– время движения на холостом ходу – 1,64 мин;

– общее время резки: 63,68 мин.

В зависимости от выбранного маршрута резки числовые параметры резки могут существенно различаться. Таким образом, при разработке управляющих программ для машин фигурной листовой резки с ЧПУ возникают различные задачи оптимизации маршрута инструмента. В качестве критерия оптимизации (целевой функции) в этих задачах чаще всего рассматривается общее время резки T_{cut}, рассчитываемое по формуле:

$$T_{cut} = \frac{L_{on}}{V_{on}} + \frac{L_{off}}{V_{off}} + N_{pt} \cdot t_{pt}, \tag{1.2}$$

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 точку врезки.

В (1.2) значение скорости холостого хода инструмента L_{off} — константа, определяемая техническими характеристиками используемого технологического оборудования. Значение скорости рабочего хода V_{on} программируется при

разработке управляющей программы в соответствии с используемой технологией резки и параметрами листового материала (марка материала и толщина). Предполагается, что заданная величина V_{on} в (1.3) также является константой, однако на практике фактическая скорость резки может меняться в зависимости от различных технологических факторов, а также характеристик спроектированной управляющей программы, подробнее см. [113]

Важнейшей экономической характеристикой качества разработанной управляющей программы является стоимость (себестоимость) резки деталей на машине с ЧПУ. Это сложный интегрированный показатель, который включает в себя произведенные во время резки затраты на электроэнергию и расходные материалы, на обслуживание машины с ЧПУ, а также другие эксплуатационные затраты. Стоимость резки не всегда пропорциональна времени резки, поскольку зависит еще и от различных режимов резки. По аналогии с формулой времени резки (1.2) показатель стоимости резки можно определить по следующей формуле:

$$F_{cost} = L_{on} \cdot C_{on} + L_{off} \cdot C_{off} + N_{pt} \cdot C_{pt}, \tag{1.3}$$

где C_{on} – стоимость единицы пути с включенным режущим инструментом; C_{off} – стоимость единицы пути с выключенным режущим инструментом; C_{pt} – стоимость одной точки врезки, а L_{on}, L_{off}, N_{pt} имеют тот же смысл, что и в формуле (1.2). При этом величины C_{on}, C_{off}, C_{pt} зависят от типа машины с ЧПУ, технологии резки, используемой скорости рабочего хода инструмента, толщины и марки материала.

Значения целевых функций (1.2), (1.3) однозначно определяются маршрутом резки, задаваемым кортежем (1.1), поскольку геометрия сегментов резки $S_1, S_2, \ldots, S_K$ позволяет вычислить длину рабочего хода инструмента L_{on}, а координаты точек M_0, M_1, M_1^*, M_2, M_2^*, $\ldots$, M_K, M_K^* и перестановка i_1, i_2, $\ldots$, i_K (последовательность, в которой вырезаются используемые сегменты резки) задают набор холостых перемещений инструмента, который определяет суммарную длину холостого хода L_{off}.

Таким образом, сформулированные задачи оптимизации маршрута инструмента для машин фигурной листовой резки с ЧПУ можно представить в самом общем виде как задачу минимизации некоторой числовой функции $\mathfrak{F}$ (например, (1.2) или (1.3)) на множестве $\mathfrak{G}$ допустимых кортежей (1.1):

$$\mathfrak{F}(\mathfrak{R}) \to \min_{\mathfrak{R} \in \mathfrak{G}} \tag{1.4}$$

Поскольку элементы кортежа содержат (помимо последовательности резки $i_1, i_2, \ldots, i_K$, выбираемой из дискретного множества перестановок) точки врезки и точки выключения инструмента $M_k M_k^*, k \in \overline{1, K}$, которые, в свою очередь, могут быть выбраны из континуальных подмножеств евклидовой плоскости $\mathbb{R}\times\mathbb{R}$, даже в случае наложения существенных ограничений на возможность выбора допустимых сегментов S_k оптимизационная задача (1.4) может быть отнесена к классу очень сложных задач непрерывно-дискретной оптимизации.

1.3. Технологические ограничения современных машин листовой резки с ЧПУ

Полученный любым способом маршрут движения режущего инструмента (1.1), должен быть исполнен на конкретном промышленном оборудовании — режущей машине с ЧПУ. Это накладывает ряд существенных ограничений на решение задачи резки, в терминах формулы (1.4) — существенно ограничивает проблемное пространство $\mathfrak{G}$. Рассмотрим некоторые из этих ограничений.

1.3.1. Позиции точек врезки и выключения инструмента

Это естественно возникающие технологические ограничения, которые естественно называть «геометрическими», вызваны тем, что, как сказано выше, точка врезки должна располагаться на некотором ненулевом расстоянии от контура детали. Кроме того, они не должны попадать внутрь других деталей, с учетом припуска на ширину реза. Конкретные расстояния определяются используемыми технологиями.

Введем обозначения: эквидистанты замкнутых контуров $C_1, C_2, \ldots C_N$, удаленные от них на расстояние d, обозначим за $C_1^{+d}, C_2^{+d}, \ldots C_N^{+d}$, а ограниченные этими эквидистантами двумерные геометрические объекты — $\widetilde{C}_1^{+d}, \widetilde{C}_2^{+d}, \ldots \widetilde{C}_N^{+d}$, $\widetilde{C}_i^{+d} \in \mathbb{R} \times \mathbb{R}$. Для внешних контуров берется внешняя эквидистанта, для внутренних — соответственно внутренняя. Далее, пусть $OUT = \{i_1^+, i_2^+, \ldots i_{N^+}^+\} \subseteq \overline{1, N}$ — множество индексов внешних контуров, а $IN = \{i_1^-, i_2^-, \ldots i_{N^-}^-\} \subset \overline{1, N}$ — множество индексов внутренних, $|OUT| = N^+$, $|IN| = N^-$, если внутренних контуров на раскройной карте нет, то $N^+ = N$, $N^- = 0$, $IN = \varnothing$.

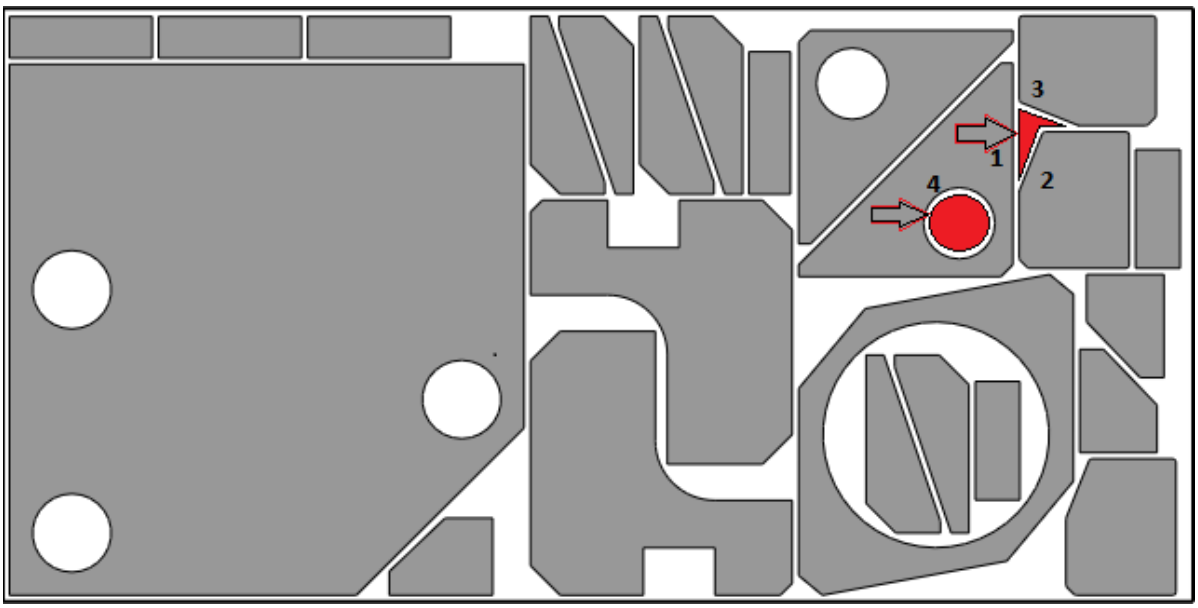


Рис. 1.6. Пример двух геометрических областей на раскройной карте, допустимых для задания точек врезки

Если минимальное расстояние от контура детали до точки врезки d_1, то позиции точек врезки должны удовлетворять условию $M_i \in G_M, \forall i \in \overline{1,N}$, где

$$G_M = \left(\mathcal{B} \setminus \bigcup_{i \in OUT} \widetilde{C}_i^{+d_1} \right) \cup \bigcup_{i \in IN} \widetilde{C}_i^{+d_1} \tag{1.5}$$

Аналогично может быть записано ограничение для позиций точек выключения инструмента $M_i^* \in G_{M^*}, \forall i \in \overline{1,N}$:

$$G_{M^*} = \left(\mathcal{B} \setminus \bigcup_{i \in OUT} \widetilde{C}_i^{+d_2} \right) \cup \bigcup_{i \in IN} \widetilde{C}_i^{+d_2}, \tag{1.6}$$

где d_2 — допустимое расстояние от контуров деталей до точек выключения инструмента; как правило $0 \leqslant d_2 < d_1$.

На рис. 1.6 показаны стрелками и выделены цветом две геометрические области листа, одна, определяемая внешними граничными контурами деталей, обозначенных цифрами *1, 2* и *3*, и вторая — внутренним граничным контуром *4*. Минимально допустимое расстояние от граничных контуров *1–4* до возможных точек врезки $d_1 = 9.5$ мм.

1.3.2. Условия предшествования

Это одно из самых «популярных» технологических ограничений на элементы маршрута (1.1), оно обусловлено особенностями машин портального типа [11; 13; 103]. После вырезания замкнутого контура, его внутренняя часть

ничем не удерживается и может сдвигаться, поворачиваться, наклоняться или даже падать. В некоторых случаях возможно столкновение режущей головки и заготовки с повреждением дорогостоящего инструмента. Во избежание этого, следует придерживаться следующих правил:

1. Если внешний контур содержит один или более внутренних контуров, которые представляют собой границы отверстий в деталях, то прежде, чем будет начата вырезка внешнего контура, должны быть вырезаны все внутренние контуры.

2. Если внутренний контур детали на раскройной карте содержит внешний контур/контуры другой детали, то сначала должна быть вырезана эта другая деталь с соблюдением предыдущего пункта.

Эти правила и называются *условиями предшествования* для перестановки $I = (i_1, i_2, \ldots, i_K)$. В терминах ее элементов условие означает следующее:

– если в перестановке $I = (i_1, i_2, \ldots, i_K)$ сегмент i_k содержит внешний контур, то все соответствующие внутренние контуры должны содержаться в сегментах, предшествующих сегменту i_k в перестановке;

– если в перестановке $I = (i_1, i_2, \ldots, i_K)$ сегмент i_k содержит внутренний контур, который на раскройной карте содержит внутри внешний контур, соответствующий другому объекту A_l $(l = 1, 2, \ldots, n)$, то этот внешний контур должен быть вырезан в сегментах, предшествующих сегменту i_k в перестановке I.

Таким образом, ограничения предшествования ограничивают множество допустимых перестановок $I = (i_1, i_2, \ldots, i_K)$ в маршруте $\mathfrak{R}$ и могут называться *комбинаторными* или дискретными ограничениями. Следует отметить, что они (так же, как ограничения предыдущего раздела 1.3.1) однозначно вычисляются по раскройной карте и не зависят от других элементов кортежа (1.1).

1.3.3. Эвристические правила термической резки заготовок из листовых материалов

В отличие от вышеописанных, следующий тип технологических требований накладывает условия на выбор точки врезки и порядка резки сегментов в зависимости от того, какие параметры маршрута резки были выбраны на

предыдущих шагах. Этот тип ограничений обусловлен геометрическими искажениями материала в процессе термической резки деталей. Эти правила разработаны сотрудниками ОАО «Уралхиммаш» В.И. Кротовым и А.Д. Гуртовенко на основе опыта резки листовых материалов на машинах термической резки с ЧПУ в котельно-заготовительном комплексе предприятия.

Термические воздействия на вырезаемые заготовки можно подразделить на два типа:

– общие изменения геометрических размеров заготовки (уменьшение) вследствие ее вырезания из нагретой части материала;

– изменение геометрической формы заготовок (изменение радиусов у секторов, отклонения от прямолинейности у прямоугольных деталей) и др. Чем больше геометрические размеры заготовки, тем больше изменения. Наиболее подвержены изменениям узкие длинные заготовки.

На величину термических деформаций оказывают влияние:

– тип резки: газовая, плазменная, лазерная, ...

– марка материала, его теплопроводность;

– состояние поставки металла (наличие внутренних напряжений), его термообработка;

– толщина металла;

– выбор порядка резки заготовок;

– выбор точек врезки для каждого контура;

– направление обхода контура (по/против часовой стрелки).

– и т.п.

Наиболее важные из технологических требований резки, обусловленные наличием термических деформаций материала, сформулированы в виде двух правил — «жесткости заготовки / детали» и «жесткости листа / материала», см. [62; 63].

Правило «жесткости детали»

При резке контура точка врезки и направление резки контура выбираются таким образом, чтобы сначала вырезались участки контура, расположенные в непосредственной близости к границе материала, либо к границе вырезанной области, а завершение резки происходило по участку контура, граничащего с «жесткой» (не вырезанной) частью области.

Важно отметить, что при изменении порядка вырезки заготовок (например, в последовательности сверху вниз) изменится и набор допустимых точек врезки и направлений реза.

Правило «жесткости листа»

Это правило накладывает ограничение на последовательность $i_1, i_2, \ldots, i_k$, то есть порядок, в котором вырезаются используемые сегменты резки $S_1, S_2, \ldots, S_K$. Фактически оно состоит из нескольких эмпирических условий.

- Если среди заготовок имеются длинномерные детали (один из габаритов больше другого в ≈ 10 раз и более) и они расположены вблизи узкой границы материала, то процесс резки следует начинать с них, так как именно такого рода заготовки подвержены максимальным тепловым деформациям.

- Если на материале есть крупный отход, то процесс резки следует начать с противоположной стороны, поскольку аккумулирующееся в материале в процессе резки тепло в конечной стадии резки должно быть несколько скомпенсировано «жестким» остатком.

- Иначе резку следует начинать с той стороны, где суммарные тепловыделения от резки больше (больше мелких деталей, либо больше суммарный периметр реза).

- При выборе последовательности вырезаемых заготовок на материале не должно оставаться узких полос и «островов», содержащих не вырезанные заготовки.

Следует отметить, что, как показала практика, правила «жесткости заготовки» и «жесткости материала» целесообразно учитывать не только при разработке управляющих программ для машин газовой, плазменной и лазерной резки с ЧПУ, но и при применении машин гидроабразивной фигурной листовой резки. Этот факт свидетельствует о том, что изменения геометрических характеристик материала связаны не только с термическими деформациями, но и с механическими трансформациями материала при листовой резке заготовок на машинах с ЧПУ, см. [110].

Вопросы математической формализации ограничений, связанных именно с технологическими требованиями термической резки, все еще остаются наименее изученными.

Если обозначить через $\mathfrak{R}_\nu$ частичный маршрут резки первых ν сегментов $(\nu < K)$

$$\mathfrak{R}_\nu = \langle M_0, M_1, S_1, M_1^*, \ldots, M_\nu, S_\nu, M_\nu^*, i_1, i_2, \ldots, i_\nu \rangle ,$$

то правила «жесткости заготовки» и «жесткости материала» при формировании допустимого маршрута $\mathfrak{R}$, помимо соблюдения условий предшествования для перестановки $i_1, i_2, \ldots, i_K$ и условий (1.5) и (1.6), формирует следующее дополнительное условие: если $\mathfrak{R}_\nu$ – частичный маршрут, допустимый с точки зрения всех технологических требований листовой резки, сформулированных в этом параграфе, то сегмент с номером $(\nu + 1)$ и соответствующая точка врезки $M_{(\nu+1)}$ для него в маршруте $\mathfrak{R}_{(\nu+1)}$ должны выбираться с учетом уже выбранного частичного маршрута $\mathfrak{R}_\nu$, что фактически означает либо запрет на некоторые «плохие» номера сегментов $i_{(\nu+1)}$ и «плохие» точки врезки $M_{(\nu+1)}$ в области G_M, либо наложение «штрафа» на «плохие» значения этих параметров кортежа посредством включения наложенного штрафа в целевые функции (1.2) – (1.3) при решении оптимизационной задачи (1.4).

Таким образом, условия «жесткости заготовки» и «жесткости материала» порождают для задачи непрерывно-дискретной оптимизации своего рода *динамические* ограничения, формируемые только в процессе вычисления допустимого решения задачи.

1.4. Классификация задач маршрутизации инструмента машин листовой резки

Общая задача оптимальной маршрутизации режущего инструмента для машин термической резки листового материала с ЧПУ (1.4), хотя и решается на евклидовой плоскости $\mathbb{R} \times \mathbb{R}$, содержит в себе задачу коммивояжера (Traveling Salesman Problem, TSP), поэтому является NP-трудной, и ее полное решение — непрактично или даже невозможно в подавляющем большинстве случаев, возникающих в реальном производстве (сотни и тысячи деталей / контуров). Поэтому вместо общей задачи резки как правило рассматривается один из ее частных случаев. Различные постановки отличаются между собой

- типом использованной техники резки;
- способом выбора положений точек врезки;
- набором учитываемых технологических ограничений.

В литературе можно встретить большое количество видов задач маршрутизации режущего инструмента, см. например, [11; 28; 65], в рамках данной диссертационной работы используется следующая классификация (см. рис. 1.7):

- **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

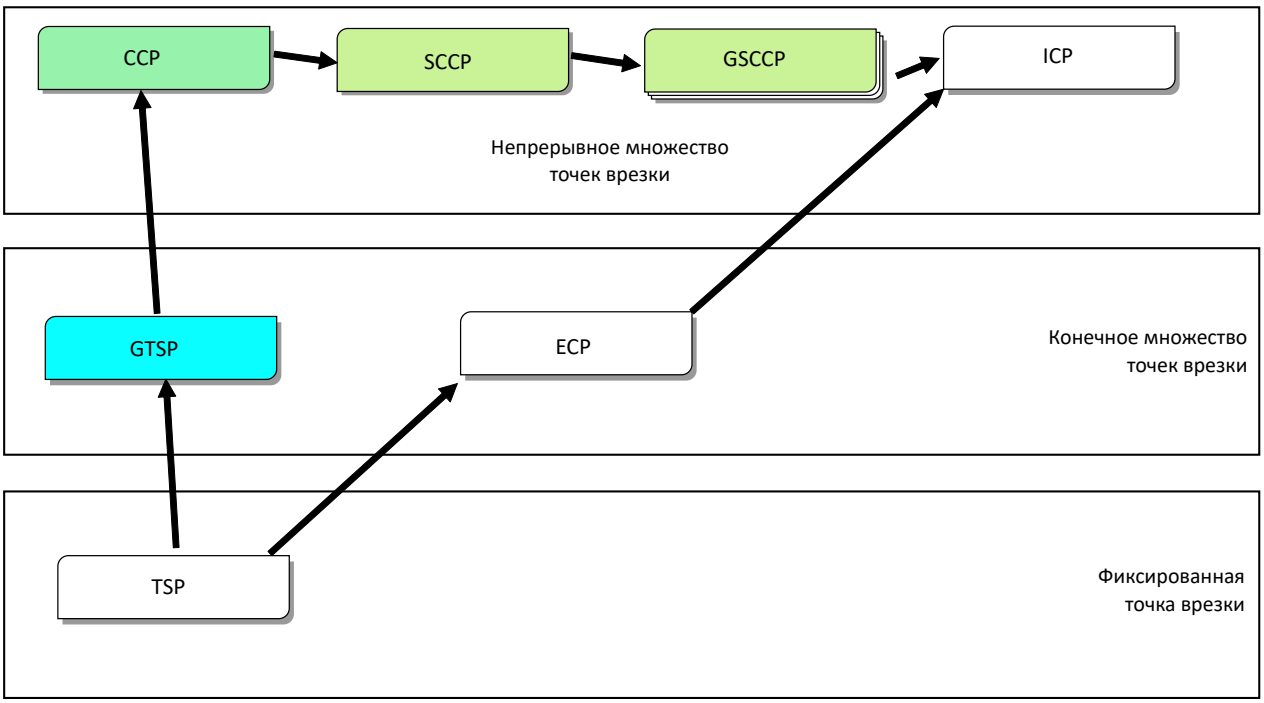


Рис. 1.7. Классификация задач резки

– **Сегментная задача непрерывной резки** (Segment Continuous Cutting Problem, SCCP): вводится понятие сегмента как обобщение понятия контура (см. Определение 1.1); сегмент 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.

В подавляющем большинстве исследований непрерывно-дискретная задача маршрутизации режущего инструмента (1.4) как правило сводится к задаче только дискретной оптимизации. Для этого непрерывный контур детали заменяется на конечное число потенциальных точек врезки, как правило расположенных на нём с некоторым шагом ε, с учётом условия (1.5), то есть фактически сводится к ECP [12; 30; 75] или её частному случаю — GTSP [5; 58; 89; 90]. В

этом случае, полная ошибка в расчёте длины маршрута резки составляет $N \cdot \varepsilon$, где N – количество контуров, подлежащих резке. С ростом количества контуров (что актуально для современного производства), растёт и ошибка, и для того, чтобы гарантировать точность δ, необходимо выбирать малый шаг дискретизации $\varepsilon \approx \delta/N$, в результате чего общее количество точек на контурах растёт ($\sim O(N)$) и время полного перебора также растёт, но уже экспоненциально. Тем не менее, существуют изощрённые эвристики решения таких задач даже для больших размерностей [8]. Использование непрерывной оптимизации, без предварительной дискретизации — сравнительно мало исследованная область, см. например [2; 84].

1.5. Современное состояние проблемы исследования и применение алгоритмов маршрутизации для автоматизированного проектирования управляющих программ

Если говорить в целом о проблеме оптимизации траектории инструмента для технологического оборудования листовой резки с ЧПУ, то в настоящее время не существует единой теоретической базы для решения этой проблемы. Существуют отдельные научные группы, которые занимаются исследованием частных случаев проблемы резки. Кроме того, в состав CAD/CAM систем, предназначенных для проектирования раскроя и управляющих программ для машин листовой резки с ЧПУ, входят отдельные модули, позволяющие решать некоторые оптимизационные задачи, например минимизацию холостого хода инструмента, которые, однако не обеспечивают соблюдение технологических требований резки материала на машинах с ЧПУ и не позволяют получать маршруты резки, близкие к оптимальным с точки зрения интегрированного критерия стоимости резки с учетом рабочего хода инструмента, затрат на врезку и т.п. Вместе с тем в сочетании с интерактивными методами проектирования они обеспечивают рациональные и технологически допустимые варианты траекторий инструмента машины с ЧПУ. Следует подчеркнуть, что алгоритмы, реализованные в коммерческом программном обеспечении, не принято описывать в научной литературе.

В нашей стране первые работы по оптимизации проектирования маршрута листовой резки на машинах с ЧПУ были опубликованы проф. М.А. Верхотуровым (Уфа) [95—97] и проф. В.Д. Фроловским (Новосибирск) [21; 111;

120]. Авторы использовали простую модель, эквивалентную классической задаче коммивояжера. В последние годы появилось несколько публикаций, проводимых под руководством проф. А.В. Панюкова (Челябинск) по этой тематике [47—49]. Эти работы представляют более теоретический интерес в смысле общей маршрутизации в графах, поскольку получаемые траектории не всегда могут быть реализованы на машинах листовой резки с ЧПУ. Можно отметить также работы, проведенные в Перми [94; 99; 100; 119].

Из зарубежных конкурентов следует особо выделить группу ученых из Бельгии [11—14]. Они занимаются оптимизацией траектории инструмента преимущественно для лазерного оборудования и разработкой алгоритмов для двух классов задач с дискретными моделями — GTSP и ECP. Ряд исследователей из Китая, Гонконга, Японии и др. стран также периодически публикуют свои результаты (см., например, [18; 27; 35; 45; 84; 85; 89; 92]). Однако они, как правило, касаются разработки отдельных алгоритмов только для одного из вышеприведенных в разделе 1.4 классов задач, при этом часто не учитывают важные технологические ограничения листовой резки на машинах с ЧПУ. В частности, учет термических деформаций заготовок и искажение их геометрических размеров не являются предметом большинства исследований, что делает эти работы в достаточной степени академическими и не в полной мере приемлемыми для практики. Вместе с тем, эти работы вносят свой вклад в теорию и практику экстремальных задач маршрутизации инструмента машин листовой резки с ЧПУ.

В целом можно отметить, что за рамками современных исследований отечественных и зарубежных коллег, в основном, остаются следующие принципиальные моменты:

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

– Адекватный учет тепловых искажений заготовок при термической резке материала на машине с ЧПУ.

– Разработка комплексного подхода к решению оптимизационных задач всех вышеперечисленных в разделе 1.4 классов, в особенности задачи прерывистой резки ICP, для различного технологического оборудования листовой фигурной резки с ЧПУ.

– Разработка алгоритмов решения задач нескольких классов с применением специальных техник резки и ориентированных на минимизацию стоимости процесса резки, а не только на минимизацию холостого хода инструмента.

– Разработка алгоритмов, позволяющих эффективно решать задачи с непрерывными моделями (CCP, SCCP), для которых точки врезки в материал могут выбираться из континуальных множеств на плоскости.

– Разработка оценок вычислительной сложности и точности разрабатываемых алгоритмов для практических задач маршрутизации инструмента с дополнительными ограничениями.

Применение эффективных классических метаэвристических алгоритмов дискретной оптимизации (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. Однако, эта адаптация наталкивается на серьёзные проблемы, связанные с невозможностью учета некоторых технологических ограничений при «лобовом» применении упомянутых методов [22; 23; 27; 37; 56; 68].

Таким образом, необходимость в создании новых оптимизационных постановок задач, алгоритмов и программного обеспечения представляется доминантой развития методов решения задачи оптимальной маршрутизации режущего инструмента для машин термической резки с ЧПУ.

1.6. Выводы по Главе 1

Анализ состояния вопросов автоматизации технологической подготовки УП для машин листовой термической резки с ЧПУ позволяет сделать следующие выводы:

1. Задача оптимальной маршрутизации режущего инструмента представляет собой чрезвычайно сложную задачу непрерывно-дискретной оптимизации, требующую решения множества теоретических и практических вопросов, включая:

- Использование различных техник резки, помимо стандартной;
- Учет разнообразных технологических ограничений на маршрут резки.

2. Несмотря на большой объем уже проведенных исследований, продолжение изучения задачи представляет большой интерес как с теоретической точки зрения, так и для практического применения, в частности:

   - Разработка алгоритмов, решающих задачи разных классов, имея в виду решение задачи прерывистой резки ICP
   - Развитие подходов к использованию непрерывной оптимизации в противовес чисто дискретным
   - Оценка качества решений, получаемых эвристиками и метаэвристиками

Глава 2. Разработка алгоритмов маршрутизации инструмента на основе дискретных оптимизационных моделей

2.1. Использование модели обобщенной задачи коммивояжера с ограничениями предшествования PCGTSP для формализации задачи маршрутизации

Обобщенная задача коммивояжера (Generalized Traveling Salesman Problem, GTSP) – широко известная задача комбинаторной оптимизации, впервые сформулированная в основополагающей статье Шриваставы и др. [77] и привлекшая внимание многих исследователей, см., например, обзор в [67]. В GTSP для заданного взвешенного орграфа $G = (V, E, c)$ и разбиения $V_1 \cup \ldots \cup V_m$ набора узлов V графа G на непустые взаимно непересекающиеся кластеры требуется найти замкнутый тур с минимальной стоимостью, который посещает каждый кластер V_i в точности один раз.

В этой главе рассматривается обобщенная задача коммивояжера с ограничениями предшествования (Precedence Constrained Generalized Traveling Salesman Problem, PCGTSP), в которой необходимо посещать кластеры в соответствии с заранее заданным частичным порядком. Эта модификация задачи GTSP имеет множество практических применений, среди которых задачи оптимизации траектории инструмента для станков с ЧПУ [4], минимизации времени холостого хода в процессе раскроя листового металла [8; 47], настройки координатно-измерительного оборудования [70], оптимизации траектории при множественном сверлении отверстий [10], маршрутизации беспилотных летательных аппаратов [106].

Задача GTSP является обобщением классической задачи коммивояжера (Traveling Salesman Problem, TSP), поэтому она также является NP-трудной даже на евклидовой плоскости, будучи параметризованной количеством кластеров m [57]. В то же время хорошо известная схема динамического программирования Хелда и Карпа [26], адаптированная к GTSP, обладает трудоемкостью $O(n^3 m^2 \cdot 2^m)$, то есть GTSP принадлежит классу FPT относительно параметризации количеством кластеров. Более того, при $m = O(\log n)$ её оптимальное решение может быть найдено за полиномиальное время. Исследования в области

алгоритмического анализа задачи GTSP развивались по нескольким основным направлениям.

Первый подход основан на сведении исходной задачи к подходящей постановке асимметричной задачи коммивояжера (ATSP) и последующем решении полученной вспомогательной задачи [44; 53] Несмотря на математическое изящество, этот подход не свободен от ряда известных недостатков:

1. получаемые в результате такого сведения постановки задачи ATSP обладают специфической структурой, затрудняющей их численное решение даже на современных MIP-решателях, таких как Gurobi и CPLEX;

2. даже близкие по функционалу к оптимальным приближенные решения вспомогательной задачи ATSP могут соответствовать недопустимым решениям исходной задачи [36].

Другой известный подход связан с разработкой точных алгоритмов для частных случаев задачи GTSP и приближенных алгоритмов с теоретическими оценками, включая алгоритмы ветвей и границ (см., например, [20; 91]) и полиномиальные приближенные схемы (PTAS) [19; 38].

Наконец, третий подход заключается в разработке новых и адаптации известных эвристик и метаэвристик. Среди известных результатов в этом направлении выделяются: гибридный алгоритм Гутина и Карапетяна [24], адаптация известного солвера Лина-Кернигана-Хельсгауна [27] и метаэвристика адаптивного поиска в больших окрестностях (Adaptive Large Neighborhood Search, ALNS) [76], обладающая рекордной на сегодняшний день практической производительностью.

К сожалению, алгоритмические результаты для рассматриваемой в данной главе задачи PCGTSP до сих пор остаются немногочисленными и исчерпываются приведенным ниже списком.

1. Эффективные алгоритмы для специальных ограничений предшествования типа Баласа [1; 6; 7] и ограничений предшествования, приводящих к квази- и псевдопирамидальным оптимальным маршрутам [40; 41],

2. Общий подход к выводу нижних оценок в методе ветвей и границ [71],

3. Новый метаэвристический солвер PCGLNS [39; 43], развивающий результаты, полученные в [76] для GTSP.

Постановка задачи

Рассмотрим постановку задачи PCGTSP (обобщённой задачи коммивояжера с ограничениями предшествования) в самом общем виде [42]. Условие задачи определяется тройкой $(G, \mathcal{C}, \Pi)$, где

- $G = (V, E, c)$ – взвешенный ориентированный граф, вес произвольной дуги $(u, v) \in E$ которого задается соотношением $c(u, v)$,
- $\mathcal{C} = \{V_1, \ldots, V_m\}$ – разбиение множества V вершин графа G на m непустых попарно непересекающихся кластеров,
- ориентированный ациклический граф $\Pi = (\mathcal{C}, A)$ задает частичный порядок (ограничения предшествования) на множестве кластеров $\mathcal{C}$.

Каждой вершине $v \in V$ графа G сопоставим (единственный) кластер $V(v)$, содержащий данную вершину: $v \in V(v)$. Далее, без ограничения общности, полагаем

- орграф Π *транзитивно замкнутым*: соотношения $(V_i, V_j) \in A$ и $(V_j, V_k) \in A$ влекут $(V_i, V_k) \in A$ для произвольных индексов i, j и k;
- верным включение $(V_1, V_p) \in A$ для каждого $p \in \{2, \ldots, m\}$.

Договоримся замкнутый маршрут $T = v_1, v_2, \ldots, v_m$, называть *допустимым решением* задачи PCGTSP, если

- маршрут T начинается и заканчивается в произвольной вершине $v_1 \in V_1$,
- произвольный кластер $V_p \in \mathcal{C}$ посещается маршрутом T в точности один раз,
- маршрут T *соответствует* частичному порядку Π, то есть любой кластер V_q посещается маршрутом T только после всех кластеров, предшествующих ему в Π.

Стоимость решения T определяется соотношением

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

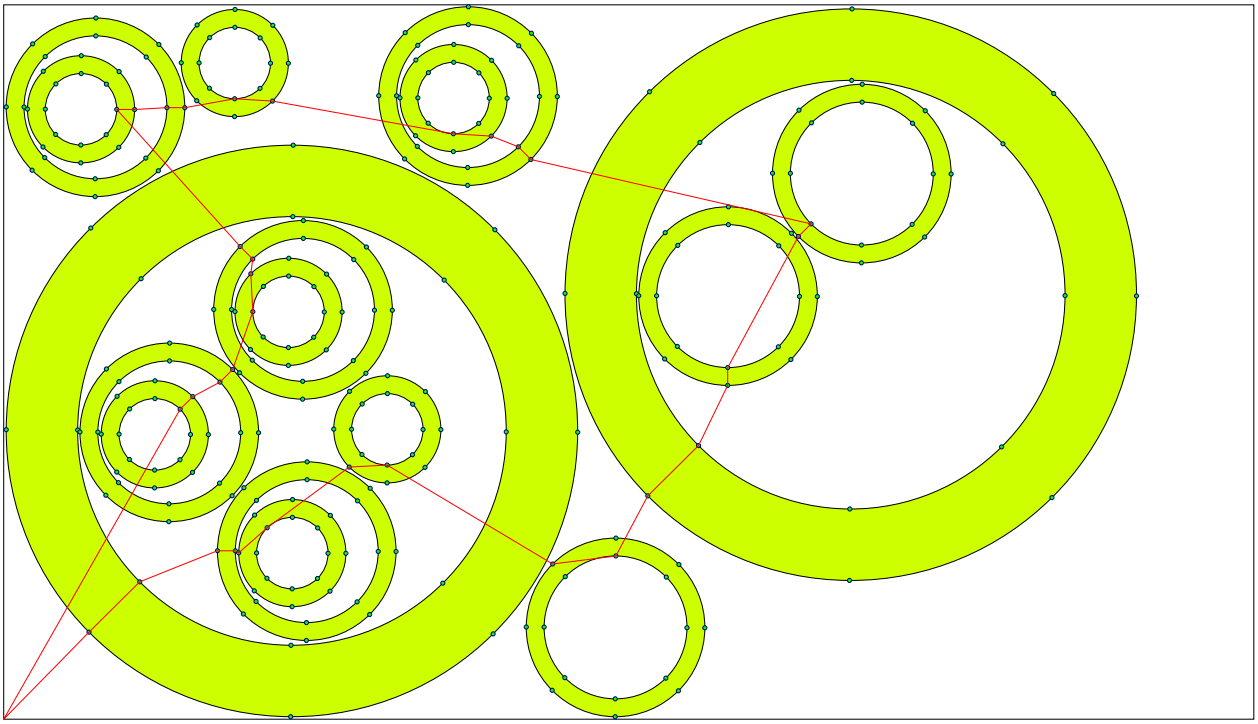


Рис. 2.1. Пример решения задачи PCGTSP, полученного эвристикой PCGLNS

Требуется найти допустимое решение T с минимальной стоимостью $cost(T) \to \min$.

Пример оптимального решения задачи PCGTSP для $m = 34$ кластеров на евклидовой плоскости, полученного эвристикой PCGLNS [39], приведен на рис. 2.1.

2.2. Общие соображения

Описываемые в этой главе алгоритмы опираются на несколько общих идей. Ветвление осуществляется по переменным V_i, то есть на каждом шаге мы выбираем какой кластер будет посещен следующим. Использование ограничений предшествования Π при этом позволяет уменьшить размер дерева поиска, для некоторых постановок — существенно. Каждый узел дерева поиска соответствует начальной части полного маршрута T, который мы будем называть *префиксом* $\sigma = (V_{i_1}, V_{i_2}, \dots V_{i_r})$, где по постановке $V_{i_1} = V_1$. Кластеры, входящие в префикс, посещаются в порядке σ, а все остальные — в произвольном порядке, соблюдая ограничения предшествования.

2.2.1. Получение нижних оценок

Используя подход динамического программирования (Dynamic Programming, DP), легко найти кратчайший путь вдоль σ между всеми парами вершин (v_1, v_r), где $v_1 \in V_{i_1}, v_r \in V_{i_r}$

$$c_{min}(\sigma) = \min_{\substack{v_1 \in V_{i_1} \\ v_r \in V_{i_r}}} \{cost\,(P_{v_1,v_r}) : P_{v_1,v_r} \text{ — частичный путь в порядке } \sigma\} \qquad (2.1)$$

Таким образом, исходная постановка задачи PCGTSP может быть в каждом узле дерева поиска декомпозирована в две (более простых) задачи:

1. вычисление $c_{min}(\sigma)$
2. задачу PCGTSP $\mathcal{P}(\sigma)$, полученную релаксацией исходной задачи PCGTSP следующим образом:
   - использует те же ограничения предшествования
   - содержит те же узлы и кластера, за исключением внутренних кластеров префикса $V_{i_2}, \ldots V_{i_{r-1}}$
   - ребра, исходящие из V_{i_1} и входящие в V_{i_r}, удаляются
   - веса ребер $c(u, v)$ сохраняются,
   - для всех пар вершин $v_1 \in V_{i_1}, v_r \in V_{i_r}$ добавляются ребра нулевого веса $c(v_1, v_r) = 0$

Утверждение 2.1.

$$\min_T\{cost(T)|T \textit{ начинается с префикса } \sigma\} \geqslant$$
$$c_{min}(\sigma) + \min_T\{cost(T)|T \textit{ – допустимое решение задачи } \mathcal{P}(\sigma)\} \qquad (2.2)$$

Доказательство. Любой маршрут T, начинающийся с префикса σ, делится на две части — одну вдоль σ, вторую от V_{i_r} до V_{i_1}. Первая имеет вес не менее $c_{min}(\sigma)$ по определению. Вторая может быть дополнена подходящим ребром, идущим из V_{i_r} в V_{i_r} (имеющим нулевой вес) и тем самым давая решение задачи $\mathcal{P}(\sigma)$. □

Нижняя оценка решения исходной задачи PCGTSP, таким образом, в каждом узле дерева поиска σ оценивается по формуле (2.2). Важно заметить, что

многие задачи $\mathcal{P}(\sigma)$ оказываются идентичными для разных σ (отличающихся порядком посещения внутренних кластеров), что позволяет значительно уменьшить общее время вычислений.

В то же время, $\mathcal{P}(\sigma)$ все еще представляет собой экземпляр задачи PCGTSP, хотя и меньше размером, чем исходная постановка. Заменим ее решение в формуле (2.2) на нижнюю границу, полученную тем или иным способом

$$\min_T\{cost(T)|T \text{ – допустимое решение задачи } \mathcal{P}(\sigma)\} \geqslant LB(\mathcal{P}(\sigma))$$

В данной диссертационной работе для вычисления оценки $LB(\mathcal{P}(\sigma))$ вслед за [71] используется трехступенчатая релаксация:

1. $\mathcal{P}(\sigma) \to$ GTSP
2. GTSP $\to$ ATSP
3. ATSP $\to \mathcal{P}_{rel}(\sigma)$

и решение задачи $\mathcal{P}_{rel}(\sigma)$ принимается за нижнюю границу $LB(\mathcal{P}(\sigma))$:

$$\min_T\{cost(T)|T \text{ начинается с префикса } \sigma\} \geqslant c_{min}(\sigma) + \mathrm{OPT}(\mathcal{P}_{rel}) \qquad (2.3)$$

На всех шагах релаксации, кроме первого, используются несколько методов, так что в общем случае получается *несколько* оценок $\mathrm{OPT}(\mathcal{P}_{rel})$, в формулу (2.3) подставляется *наибольшая* из них. Опишем применявшиеся в данной диссертационной работе методы релаксации задачи $\mathcal{P}(\sigma)$ подробнее.

Сведение PCGTSP к GTSP

Для сведения $\mathcal{P}(\sigma)$ к GTSP необходимо снять ограничения предшествования. Это можно сделать не полностью, а частично — ослабить путем удаления ребер (v_p, v_q), которые заведомо запрещены частичным порядком Π, то есть $(V(v_q), V(v_p)) \in A$. При этом маршрут из v_p в v_q все равно остается возможным, но через одну или несколько промежуточных вершин.

Более строгий анализ показывает, что можно удалить еще больше ребер.

Так, рассмотрим ребро (v_p, v_q), которое формально соответствует частичному порядку Π ($(V(v_p), V(v_q)) \in A$), но существует промежуточный кластер V^*: $(V(v_p), V^*), (V^*, V(v_q)) \in A$. Тогда ребро (v_p, v_q) тоже оказывается запрещено ограничениями предшествования, так как любой допустимый маршрут

$v_p \to v_q$ должен проходить через некоторую вершину $v^* \in V^*$. Для эффективного поиска таких (запрещенных) ребер в данной диссертационной работе заранее вычисляется *транзитивное сокращение* орграфа Π, то есть минимальный набор ребер A', транзитивное замыкание которого дает A. Например, в библиотеке NetworkX [52] для вычисления транзитивного сокращения имеется функция *transitive_reduction()*. Тогда подлежат удалению ребра

$$\{(v_p, v_q) | (V(v_p), V(v_q)) \in A, (V(v_p), V(v_q)) \notin A'\}$$

Кроме того, отдельно следует рассмотреть ребра (v_p, v_q) в случае, когда $v_q \in V_1$. Для них условие удаления меняется на почти обратное: остаются в графе только те ребра, которые выходят из «финальных» кластеров, то есть $\nexists V^+ \colon (V(v_p), V^+) \in A$.

Поскольку мы ослабили ограничения предшествования, то $\mathrm{OPT}(\mathcal{P}) \geqslant \mathrm{OPT}(GTSP)$.

Сведение GTSP к ATSP

Частично выполнив таким образом ограничения предшествования, далее сводим получившуюся задачу GTSP к ATSP. Сделать это можно несколькими способами.

1. Преобразование Нуна и Бина [53], сводящее GTSP к ATSP того же размера n
   - Кластеры V_i (за исключением вырожденного случая $|V_i| = 1$) замыкаются в циклы нулевого веса, то есть для каждой вершины v_i добавляется ребро $c(v_i^-, v_i) = 0$, где $v_i^- \in V(v_i)$ — вершина, предшествующая v_i в цикле, порядок замыкания вершин кластера в цикл — произвольный
   - Ребро (v_i, v_j) заменяется на ребро (v_i^-, v_j) того же веса.
2. *Граф кластеров* $H_1 = (\mathcal{C}, A_1)$, индуцированный исходным графом G по правилу
   $$c(V_i, V_j) = \min_{\substack{v_i \in V_i \\ v_j \in V_j}} c(v_i, v_j)$$
   Тем самым получается задача ATSP размера m. Легко понять, что $LB(\mathcal{P}) \geqslant LB(H_1)$.

3. Граф кластеров $H_2 = (\mathcal{C}, A_2)$, также индуцированный графом G, но используются веса путей длины 2, удовлетворяющих ограничениям предшествования

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

   Можно доказать, что нижняя оценка на решение задачи ATSP для графа H_2 является также нижней оценкой для построенной выше задачи GTSP, а значит и для исходной задачи $\mathcal{P}$.

   Причина использования графа H_2 заключается в том, что граф H_1 фактически не требует, чтобы маршрут входил и покидал кластер V_i через одну и ту же вершину $v_i \in V_i$, а использование путей длины 2 хотя бы частично учитывает это требование.

   При построении H_2 ограничения предшествования учитываются сложнее, чем для преобразования Нуна и Бина или графа H_1: оба ребра (v_i, v_k) и (v_k, v_j) должны быть допустимы с точки зрения ослабленных ограничений предшествования, описанных выше.

4. Графы кластеров H_3, H_4 и т.д. — строятся аналогично H_2, но при помощи путей большей длины, что позволяет получить альтернативные нижние оценки. Ввиду того, что построение таких графов заметно сложнее алгоритмически, в данной диссертационной работе они не использовались.

Релаксация ATSP и ее решение

Теперь задача, подлежащая решению ещё упростилась, теперь это просто TSP (возможно, асимметричная) и возможно меньшего размера $m < n$, чем исходная. Однако она все еще экспоненциально сложная, поэтому еще раз упростим ее, причем это тоже можно сделать несколькими способами.

1. Дополнительно релаксируем полученную задачу ATSP, решая вместо нее задачу нахождения минимального остовного (ориентированного) дерева MSAP (Minimum Spanning Arborescence Problem) на том же графе. Фактически мы снимаем ограничение того, что степень каждой вершины полученного решения должна быть в точности два, и вместо цикла ищем вершинное покрытие в виде дерева, что может только понизить вес ответа: $\mathrm{OPT}(MSAP) \leqslant \mathrm{OPT}(ATSP)$

Таблица 2.1

Методы оценки нижней границы

	Нун и Бин	$\mathbf{H_1}$	$\mathbf{H_2}$	$\mathbf{H_{3+}}$
AP / Цикловое покрытие	E_1	$\mathbf{L_1}$	$\mathbf{L_2}$	-
MSAP / Остовное дерево	E_2	E_3	E_4	-
Gurobi + ATSPxy	E_5	$\mathbf{L_3}$	E_6	-

2. Вместо задачи ATSP решаем задачу циклового покрытия (Cycle cover) того же графа. В этом случае мы снимаем требование единственности цикла, в результате чего сложность задачи понижается с экспоненциальной до полиномиальной, а стоимость решения тоже может уменьшиться.

   С практической точки зрения, задача циклового покрытия решается как задача о назначениях (Assignment Problem, AP) на двудольном орграфе, обе доли которого конгруэнтны графу задачи ATSP.

3. И наконец, для задач ATSP небольшого размера *в некоторых случаях* она может быть прямо решена за разумное время одним из известных алгоритмов. В данной диссертационной работе для этого использовался MILP-солвер Gurobi [80], оснащенный моделью ATSPxy [72]. Последняя способна решать не только классическую задачу ATSP, но и ATSP с ограничениями предшествования, поэтому в этом варианте они прямо добавлялись в условие задачи.

   Для обращения к Gurobi использовался пакет gurobipy [81].

В результате комбинирования описанных выше методов получения нижней оценки для задачи $\mathcal{P}(\sigma)$, получается целый ряд различных оценок, которые сведены в Табл. 2.1. Ее столбцы представляют собой способы релаксации задачи GTSP к ATSP, а строки — способы релаксации и решения задачи ATSP.

Опираясь на результаты численных экспериментов (см. Табл. 2.2), были отобраны наилучшие оценки, обозначенные L_1–L_3, так как они оказались наиболее точными с доверительной вероятностью 95%. Оценки E_1–E_3 систематически оказываются ниже, а оценки E_5 и E_6 к тому же значительно более трудоемки в смысле времени счета. Таким образом, далее в данной диссертационной работе суммарная нижняя оценка рассчитывается по формуле

$$LB(\sigma) = c_{min}(\sigma) + \max_{i \in \{1,2,3\}} L_i(\mathcal{P}(\sigma)) \tag{2.4}$$

Таблица 2.2

Сравнение нижних оценок с оценкой L_3

E_1	$E_2 = E_3$	E_4
0.48 ± 0.03	0.54 ± 0.01	0.60 ± 0.002
L_1	L_2	L_3
0.91 ± 0.02	0.97 ± 0.02	1.00

2.2.2. Отсечение

Для каждого узла дерева поиска, задаваемого префиксом $\sigma = (V_{i_1}, V_{i_2}, \dots V_{i_r})$, алгоритм принимает решение о возможном отсечении. Для этого используются два соображения:

1. На основе минимальной длины $c_{min}(\sigma)$
2. На основе нижней оценки $LB(\sigma)$ (2.4)

Рассмотрим их подробнее.

Отсечение по длине путей вдоль префикса

Для быстрого инкрементального расчета минимального пути вдоль префикса $c_{min}(\sigma)$ методом динамического программирования на самом деле требуется рассчитывать и запоминать целую матрицу

$$D(\sigma)_{ij} = \min \left\{ cost(P_{v_i,v_j}) \colon v_i \in V_{i_1}, v_j \in V_{i_r}, P_{v_i,v_j} \text{ – путь } v_i\text{-}v_j \text{ в порядке } \sigma \right\}, \tag{2.5}$$

потому что она легко вычисляется инкрементально на основе такой же матрицы родительского префикса $\sigma' : |\sigma| = |\sigma'| + 1$, и очевидно

$$c_{min}(\sigma) = \min_{ij} D(\sigma)_{ij}. \tag{2.6}$$

Далее, как сказано выше, все префиксы σ, отличающиеся только порядком посещения внутренних кластеров, по построению имеют одну и ту же задачу $\mathcal{P}(\sigma)$, а значит одну и ту же ее нижнюю оценку. Построим кортеж

$$\mathcal{T}(\sigma) = (V_{i_1}, \{V_{i_1}, V_{i_2}, \dots V_{i_r}\}, V_{i_r}) \tag{2.7}$$

и множество $\mathbb{T}(\sigma) = \{\sigma' \colon \mathcal{T}(\sigma') = \mathcal{T}(\sigma)\}$ всех префиксов, дающих ту же вспомогательную задачу $\mathcal{P}(\sigma') \equiv \mathcal{P}(\sigma)$. На этом множестве найдём минимальную (поэлементно) матрицу расстояний

$$D(\mathbb{T}(\sigma))_{ij} = \min_{\sigma' \in \mathbb{T}(\sigma)} D(\sigma')_{ij}$$

Несложно понять, что если

$$D(\sigma)_{ij} > D(\mathbb{T}(\sigma))_{ij}, \forall i, j, \tag{2.8}$$

то вес любого пути, начинающего префиксом σ может быть уменьшен простой перестановкой кластеров внутри этого префикса, и следовательно заведомо не является оптимальным решением задачи PCGTSP. Значит, этот префикс может быть безопасно отброшен и не участвовать в дальнейших вычислениях.

Отсечение по нижней оценке

Если же условие (2.8) не соблюдается, алгоритм вычисляет c_{min} по формуле (2.6), строит вспомогательную задачу $\mathcal{P}(\sigma)$, как описано в разделе 2.2.1 (если она не была построена ранее, для некоторого префикса $\sigma' \in \mathbb{T}(\sigma)$) и рассчитывает полную нижнюю оценку $LB(\sigma)$ по формуле (2.4).

Вторая (и основная) стратегия отсечения заключается в отбрасывании узлов дерева поиска, для которых

$$LB(\sigma) > UB, \tag{2.9}$$

где UB — некоторая верхняя оценка на вес оптимального решения. Разумно в качестве такой оценки взять вес *любого* решения исходной задачи PCGTSP, полученного произвольным образом. В данной диссертационной работе для получения такого решения используется эвристика PCGLNS [39], которая запускается однократно перед началом работы алгоритма и за короткое время (буквально несколько секунд) дает решение, близкое к оптимальному.

Чем ниже верхняя граница UB, тем очевидно больше узлов дерева поиска будет отброшено и тем меньше будет полное время работы алгоритма.

2.2.3. Ветвление

После того, как префикс $\sigma = (V_{i_1}, V_{i_2}, \dots V_{i_r})$, полностью обработан и не подвергся отсечению, строится список его потомков, подлежащих обработке.

Иными словами, ищется множество $\mathbb{C}$ кластеров V_i, каждый из которых может быть добавлен в конец текущего префикса.

В простейшей реализации префикс может быть расширен на любой кластер, который не нарушает ограничения предшествования со всеми кластерами, уже находящимися в σ:

$$\mathbb{C}^0(\sigma) = \left\{V^+ | \forall V \in \sigma : (V^+, V) \notin A\right\} \setminus \sigma$$

Однако, так же как было с применением ограничения предшествования для релаксации PCGTSP $\rightarrow$ GTSP, более внимательно рассмотрение позволяет сильно сократить ветвление, что чрезвычайно хорошо влияет на общую производительность алгоритма. Более строгое правило заключается в том, что префикс может быть расширен на кластер, все предки которого в частичном порядке П уже находятся в префиксе:

$$\mathbb{C}(\sigma) = \left\{V^+ | \forall V \in \mathcal{C} : (V, V^+) \in A \Rightarrow V \in \sigma\right\} \setminus \sigma$$

По построенному множеству допустимых кластеров $\mathbb{C}(\sigma)$, множество префиксов-потомков строится как $\{\sigma + V | V \in \mathbb{C}(\sigma)\}$. Все они ставятся в очередь на обработку.

Сортировка ветвей

Кроме ограничения количества ветвей, оказалось, что существенное влияние на производительность алгоритма оказывает и порядок обработки ветвей. Вычислительные эксперименты показали, что размер дерева поиска систематически оказывается меньше, если первыми обрабатываются префиксы, которые сами имеют наименьшее количество потомков $|\mathbb{C}(\sigma + V)|$. Именно такая сортировка и реализована в данной диссертационной работе.

Причиной такого эффекта является по-видимому то, что префиксы, обрабатываемые первыми, имеют мало шансов быть отброшенными по правилу (2.8) на стр. 45, а последними — наоборот. Выгодно отсекать более крупные поддеревья, то есть те, которые содержат больше ветвей.

2.2.4. Обновление нижней оценки

После того, как обработаны все префиксы σ некоторой длины r (алгоритм обрабатывает префиксы в порядке увеличения длины, начиная с корня $\sigma = V_1$

длины 1, до листьев длины m), может быть вычислена текущая нижняя оценка для слоя r:

$$LB_{|r|} = \min_{|\sigma|=r} LB(\sigma) \tag{2.10}$$

по всем префиксам длины r, не подвергшимся отсечению, как описано в разделе 2.2.2.

Полная нижняя оценка при этом обновляется по формуле

$$LB \leftarrow \max(LB, LB_{|r|}) \tag{2.11}$$

2.2.5. Точное решение

Наконец, при обработке последнего «этажа» дерева поиска, префиксов длины $|\sigma| = m$ (если алгоритм не был остановлен ранее, например, по исчерпанию времени или достижению требуемой точности), процедура расчета нижней оценки уже не применяется, так как вспомогательная задача $\mathcal{P}(\sigma)$ очевидно вырождается.

Вместо этого каждый лист дерева (префикс длины m) представляет собой допустимое решение исходной задачи PCGTSP. Его вес легко считается на основе той же матрицы $D(\sigma)_{ij}$. Для этого нужно построить вспомогательный префикс $\hat{\sigma} = \sigma + V_1$, рассчитать для него матрицу $D(\hat{\sigma})_{ij}$ (она получается квадратной размера $|V_1|$) и взять ее наименьший диагональный элемент.

При этом можно обновить текущую верхнюю оценку по формуле

$$UB \leftarrow \min(UB, \min_i D(\hat{\sigma})_{ii})$$

Решение задачи PCGTSP дается префиксом σ длины m минимального веса. Сам по себе префикс содержит только кластера V_i, через которые проходит оптимальный маршрут, вершины $v_i \in V_i$ могут быть легко восстановлены стандартным для динамического программирования образом. Если в ходе вычислений матрицы $D(\sigma)_{ij}$ запоминались также индексы, дающие в нее вклад, это делается обратным ходом алгоритма. Если же для экономии памяти эти индексы не запоминались, можно повторить расчет матрицы $D(\sigma)_{ij}$ для (одного) найденного полного префикса $\hat{\sigma}$, запоминая индексы, которые и дадут номера вершин v_i в оптимальном маршруте T.

2.3. Алгоритм ветвей и границ

Соображения раздела 2.2 естественным образом соединяются в алгоритме ветвей и границ (Branch-and-Bound, BnB) классического дизайна. Дерево поиска строится начиная с корня $\sigma = V_1$ и обходится при помощи метода поиска в ширину, см. Алгоритм 2.1.

Алгоритм 2.1 BnB :: Главная процедура

Вход: орграф G, кластеры $\mathcal{C}$, частичный порядок Π

Выход: маршрут и его стоимость

1: инициализация Q = пустая очередь
2: начинаем с $Root = V_1$
3: Q.push($Root$)
4: **while not** Q.empty() **do**
5: берём следующий префикс для обработки: $\sigma = Q$.pop()
6: $process = Bound(\sigma)$ {Получение нижних границ}
7: **if not** $process$ **then**
8: префикс отсекается; **continue** {Отсечение}
9: **end if**
10: $UpdateLowerBound(\sigma)$ {Обновление нижних границ}
11: **for all** $child \in Branch(\sigma)$ **do**
12: помещаем префикс в очередь на обработку Q.push($child$)
13: **end for**
14: **end while**

К каждому узлу дерева поиска σ применяется процедура построения нижней оценки $Bound$ (Алгоритм 2.2), выполняющая следующие действия:

- на шаге 3 сопоставляем префиксу σ кортеж $\mathcal{T}(\sigma)$ (2.7)
- на шаге 4 находим матрицу (2.5)
- на шаге 6 отсекаем по условию (2.8)
- на шаге 11 вычисляем оценки L_1 и L_2 (см. табл. 2.1), сохраняя их в глобальной переменной $Opt^{\mathcal{T}}$, используя формулу

$$Opt^{\mathcal{T}(\sigma)} = \max(L_1, L_2)$$

Алгоритм 2.2 BnB :: Bound

Вход: префикс σ

Выход: подлежит ли префикс обработке?

1: **global** $D_{ij}^{\mathcal{T}}$
2: **global** $Opt^{\mathcal{T}}$
3: вычисляем кортеж $\mathcal{T} = (V_{i_1}, \{V_{i_1}, V_{i_2}, \dots V_{i_r}\}, V_{i_r})$
4: $D_{ij} = MinCosts(\sigma)$
5: **if** $D_{ij}^{(\sigma)} \geqslant D_{ij}^{\mathcal{T}}[\mathcal{T}], \forall i, j$ **then**
6: **return false**
7: **end if**
8: обновляем веса маршрутов $D_{ij}^{\mathcal{T}}[\mathcal{T}] = \min\left(D_{ij}^{\mathcal{T}}[\mathcal{T}], D_{ij}\right), \forall i, j$
9: $c_{min} = \min\limits_{i,j} D_{ij}$
10: **if** $\mathcal{T} \notin Opt^{\mathcal{T}}$ **then**
11: вычисляем нижнюю границу $Opt^{\mathcal{T}}[\mathcal{T}] = \max\left(L_1(\sigma), L_2(\sigma)\right))$
12: **end if**
13: $LB = c_{min} + Opt^{\mathcal{T}}[\mathcal{T}]$
14: **if** $LB > UB$ **then**
15: **return false**
16: **end if**
17: **return true**

– на шаге 13 рассчитываем нижнюю границу

– наконец, узел отсекается по условию (2.9)

Префиксы, которые избежали отсечения, обрабатываются процедурой ветвления $Branch$ (Алгоритм 2.3), которая пытается удлинить префикс σ на один кластер с соблюдением ограничений предшествования, как описано в разделе 2.2.3.

2.4. Динамическое программирование

Алгоритм, описанный в разделе 2.3, будучи реализован в ходе данной диссертационной работы, оказался работоспособным и может применяться как для оценки качества решений, полученных другими способами, так и для поиска точных решений задачи PCGTSP.

Алгоритм 2.3 BnB :: Branch

Вход: префикс σ

Выход: список потомков префикса для обработки

```
1: инициализация R = пустая очередь
2: for all V ∈ C \ σ do
3:    valid = true
4:    for all W ∈ C do
5:       if W ∉ σ and (W, V) ∈ Π then
6:          valid = false
7:          break
8:       end if
9:    end for
10:   if valid then
11:      добавляем новый префикс R.push(σ + V)
12:   end if
13: end for
14: return R
```

Вместе с тем, дизайн в виде классической схемы ветвей и границ проявил некоторые недостатки:

1. Использование для нижней оценки одного минимального значения c_{min} вместо полной матрицы $D(\sigma)_{ij}$ (2.6) представляется слишком сильным огрублением.

2. Префиксы σ обрабатываются независимо друг от друга, тогда как многие из них порождают одинаковую вспомогательную задачу $\mathcal{P}(\sigma)$ (2.7). Результаты ее решения требуется запоминать в глобальной переменной, что увеличивает расход памяти и затрудняет параллельное исполнение алгоритма.

В то же время, общее устройство алгоритма представляется близким к классической схеме динамического программирования (Dynamic Programming, DP) Хелда и Карпа [26], адаптированной для учёта ограничения предшествования и дополненной стратегией отсечения, представленной в основополагающей статье [50].

По этим причинам в рамках данной диссертационной работы была разработана вторая версия алгоритма, использующего соображения раздела 2.2, но в подходе DP. Она традиционно состоит из двух этапов.

1. Таблица динамического программирования строится в *прямом направлении* инкрементально, слой за слоем. Оптимальное значение решаемой задачи находится после построения последнего m-го слоя.

2. Оптимальный маршрут восстанавливается *обратным ходом* по таблице, построенной на первом этапе.

Каждое состояние DP (ячейка таблицы динамического программирования) соответствует частичному v-u-пути и индексируется кортежем $(\mathcal{C}', V_l, v, u)$, где

– $\mathcal{C}' \subset \mathcal{C}$ представляет собой *идеал* частично упорядоченного множества кластеров $\mathcal{C}$, то есть

$$\forall V \in \mathcal{C}', V' \in \mathcal{C} \colon (V', V) \in A \Rightarrow V' \in \mathcal{C}'$$

Идеал $\mathcal{C}'$ играет теперь ту же роль, что префикс σ в предыдущей версии алгоритма (раздел 2.3), тем самым сводя все «схожие» префиксы вместе для обработки. Очевидно, в наших условиях, V_1 принадлежит произвольному идеалу $\mathcal{C}' \subset \mathcal{C}$

– $V_l \subset \mathcal{C}'$, для которого нет $V \in \mathcal{C}'$, такого, что $(V_l, V) \in A$, то есть соблюдается ограничение предшествования.

– $v \in V_1$, $u \in V_l$.

Содержимое каждой записи DP S состоит из ссылки $S[pred]$ на предшествующее состояние, локальной нижней границы $S[LB]$ и стоимости $S[cost]$ соответствующего частичного v-u-пути.

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 2.4.

Алгоритм 2.4 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 $\mathcal{L}_{k+1} = \varnothing$
2: **for all** $\mathcal{C}' \in \mathfrak{I}_k$ **do**
3: **for all** кластер $V_l \in \mathcal{C} \setminus \mathcal{C}'$, s.t. $\mathcal{C}' \cup \{V_l\} \in \mathfrak{I}_{k+1}$ **do**
4: **for all** $v \in V_1$ и $u \in V_l$ **do**
5: **if** есть состояние $S = (\mathcal{C}', U, v, w) \in \mathfrak{L}_k$, s.t. $(w, u) \in E$ **then**
6: создаем новое состояние $S' = (\mathcal{C}' \cup \{V_l\}, V_l, v, u)$
7: $S'[cost] = \min\{S[cost] + c(w,u) \colon S = (\mathcal{C}', U, v, w) \in \mathcal{L}_k\}$
8: $S'[pred] = \arg\min\{S[cost] + c(w,u) \colon S = (\mathcal{C}', U, v, w) \in \mathcal{L}_k\}$
9: $S'[LB] = S'[cost] + \max\{L_1, L_2, L_3\}$
10: **if** $S'[LB] \leqslant UB$ **then**
11: добавляем S' к $\mathcal{L}_{k+1}$
12: **end if**
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **return** $\mathcal{L}_{k+1}$

Замечания

1. Оптимум для решаемой задачи дается классическим уравнением Беллмана

$$\mathrm{OPT} = \min_{v \in V_1} \min\{S[cost] + c(u,v) \colon S = (\mathcal{C}', V_l, v, u) \in \mathcal{L}_m\}$$

2. По построению, размер таблицы поиска $O(n^2 m \cdot |\mathfrak{I}|)$ и значит, время работы нашего алгоритма $O(n^3 m^2 \cdot |\mathfrak{I}|)$. В частности, в случае частичного порядка фиксированной ширины w, $|\mathfrak{I}| = O(m^w)$, см. [78]. Следовательно, оптимальное решение PCGTSP может быть найдено в этом случае за полиномиальное время даже без применения отсечения на шаге 10.

3. После построения любого из слоев $\mathcal{L}_k$, мы обновляем глобальное значение нижней границы, что улучшает точность аппроксимации.

4. Для повышения быстродействия оценка L_3 на шаге 9 вычисляется только для небольшого количества ($\approx 1\%$) состояний с наименьшей нижней границей.

2.5. Численные эксперименты

Разработанные в ходе данной диссертационной работы алгоритмы реализованы на кроссплатформенном языке программирования Python [55] и могут исполняться на всех современных операционных системах, включая Linux, MacOS и Microsoft Windows. Код оптимизирован за счёт использования библиотек NumPy [54], SciPy [74] для обработки матриц и NetworkX [52] для работы с графами, для параллельного исполнения используется модуль multiprocessing стандартной библиотеки Python. Исходный код доступен в [117].

В ходе вычислительных экспериментов сравнивалась производительность двух версий разработанного алгоритма (классическая схема ветвей и границ, раздел 2.3, и усовершенствованная схема динамического программирования Хелда и Карпа, раздел 2.4) между собой, а также с коммерческим солвером Gurobi [80], применяемым к модели целочисленной линейной оптимизации (MILP), предложенной в работе [39].

Все алгоритмы тестировались на примерах из общедоступной библиотеки PCGTSPLIB [71]. В качестве начального приближения всем алгоритмам задавалось одно и то же допустимое решение, полученное эвристикой PCGLNS [43]. Для всех алгоритмов вычисления проводятся на одном и том же оборудовании (16-ядерный Intel Xeon, 128G RAM) с предельным временем счета 10 часов. Численные эксперименты проводились на суперкомпьютере «Уран» Института математики и механики им. Н. Н. Красовского Уральского отделения Российской академии наук [121].

Для оценки точности вычислений мы используем верхнюю оценку относительной погрешности, вычисляемую по формуле

$$gap = \frac{UB - LB}{LB}, \tag{2.12}$$

критерием остановки является условие $gap < 5\%$.

Полученные результаты эксперимента представлены в табл. 2.3, 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_0, 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 оценку погрешности gap в процентах. Задачи, в которых один из предлагаемых алгоритмов превосходит Gurobi по производительности, выделены жирным шрифтом.

Как следует из табл. 2.3, для 13 из 39 задач (33%) один из разработанных алгоритмов показал рекордную производительность, в том числе в 12 случаях – по быстродействию, и в 7 – по точности.

Заметим, что предложенные алгоритмы смогли найти оптимальное решение в 6 из 39 случаях (хотя это не являлось целью данного эксперимента). Для 10 (15) постановок, включая одни из самых больших *rbg323a* и *rbg358a* (1825 и 1967 вершин соответственно) было получено решение с точностью 5% (10%). С другой стороны, для некоторых задач (например, *p43.1*, *p43.2* и *p43.3*), результаты новых алгоритмов значительно уступают Gurobi, что, по-видимому, объясняется недостаточно точными нижними оценками. В то же время для задач *p43.4* и *ry48p.4* предложенные алгоритмы значительно превзошли Gurobi.

В целом, хотя Gurobi демонстрирует в среднем чуть лучшую производительность, новые алгоритмы за редким исключением показывают вполне сопоставимые результаты. Следует добавить, что в экспериментах, проводимых в рамках данной диссертационной работы, солверу Gurobi, как и тестируемым алгоритмам, было предоставлено хорошее стартовое решение, полученное эвристикой PCGLNS [39], что, вообще говоря, не является обязательным при организации подобных сравнительных экспериментов.

Решение задач большой размерности

Предлагаемый алгоритм существенно увеличивает размеры экземпляров задач PCGTSP, для которых может быть получено точное решение с ≈ 30 кластеров [8] до 50–60, а в некоторых случаях до ≈ 150 (в зависимости от вложенности). Вместе с тем, задачи большой размерности (сотни и тысячи кластеров) все еще остаются недоступными для точного решения (за разумное время), однако для них легко находятся практически приемлемые решения.

Примеры таких маршрутов приведены на рис. 2.2. Раскройные планы предоставлены М.А. Верхотуровым, численные данные приведены в табл. 2.4.

Таблица 2.3

Сравнение решений задачи PCGTSP

Задача					*Gurobi*			*Ветвей и границ*			*DP*		
№	*ID*	n	m	UB_0	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*	*Время*	*LB*	*gap, %*
1	br17.12	92	17	43	82.00	43	0.00	**11.2**	**43**	**0.00**	27.3	43	0.00
2	ESC07	39	8	1730	0.24	1730	0.00	1.3	1726	0.23	8.37	1730	0.00
3	ESC12	65	13	1390	3.35	1390	0.00	4.3	1385	0.36	14.99	1390	0.00
4	ESC25	133	26	1418	10.61	1383	0.00	32	1383	0.00	60.69	1383	0.00
5	ESC47	244	48	1399	3773	1064	4.93	36000	980	42.76	36000	981	42.61
6	ESC63	349	64	62	25.35	62	0.00	1.3	62	0.00	**0.52**	**62**	**0.00**
7	ESC78	414	79	14872	1278.45	14630	1.66	1.3	14594	1.63	**0.68**	**14594**	**1.63**
8	ft53.1	281	53	6194	36000	5479	13.04	36000	4839	28.27	36000	4839	28.27
9	ft53.2	274	53	6653	36000	5511	20.7	36000	4934	34.84	36000	4940	34.68
10	ft53.3	281	53	8446	36000	6354	32.92	36000	5465	54.55	36000	5465	54.55
11	ft53.4	275	53	11822	20635	11259	5.00	35865	11274	4.86	**2225**	**11290**	**4.71**
12	ft70.1	346	70	32848	83.70	31521	4.21	36000	31153	5.44	36000	31177	5.36
13	ft70.2	351	70	33486	36000	31787	5.35	36000	31268	7.09	36000	31273	7.08
14	ft70.3	347	70	35309	36000	32775	7.73	36000	32180	9.72	36000	32180	9.72
15	ft70.4	353	70	44497	36000	41160	8.11	36000	38989	14.13	**36000**	**41640**	**6.86**
16	kro124p.1	514	100	33320	36000	29541	12.79	36000	27869	19.56	36000	27943	19.24
17	kro124p.2	524	100	35321	36000	29983	17.80	36000	28155	25.45	36000	28155	25.45
18	kro124p.3	534	100	41340	36000	30669	34.79	36000	28406	45.53	36000	28406	45.53
19	kro124p.4	526	100	62818	36000	46033	36.46	36000	38137	64.72	36000	38511	63.12
20	p43.1	203	43	22545	4691	21677	4.00	36000	738	2954.88	36000	788	2761.04
21	p43.2	198	43	22841	36000	21357	6.94	36000	749	2949.53	36000	877	2504.45
22	p43.3	211	43	23122	36000	15884	45.57	36000	898	2474.83	36000	906	2452.10
23	p43.4	204	43	66857	36000	45198	47.92	4470	66846	0.00	**333.02**	**66846**	**0.00**
24	prob.100	510	99	1474	36000	805	83.10	36000	632	133.23	36000	632	133.23
25	prob.42	208	41	232	13310	196	4.86	36000	149	55.70	36000	153	51.63
26	rbg048a	255	49	282	24.22	282	0.00	0.9	272	3.68	**0.25**	**272**	**3.68**
27	rbg050c	259	51	378	13.83	378	0.00	**0.2**	**372**	**1.61**	0.25	372	1.61
28	rbg109a	573	110	848	6	848	0.00	2407	812	4.43	682	809	4.82
29	rbg150a	871	151	1415	15	1382	2.38	**0.4**	**1353**	**4.58**	0.53	1353	4.58
30	rbg174a	962	175	1644	27	1605	2.43	**0.4**	**1568**	**4.85**	0.67	1568	4.85
31	rbg253a	1389	254	2376	61	2307	2.99	**0.8**	**2269**	**4.72**	1.42	2269	4.72
32	rbg323a	1825	324	2547	416	2490	2.29	**2.0**	**2448**	**4.04**	3.59	2448	4.04
33	rbg341a	1822	342	2101	18470	2033	4.97	36000	1840	14.18	36000	1840	14.18
34	rbg358a	1967	359	2080	17807	1982	4.95	36000	1933	7.60	36000	1933	7.60
35	rbg378a	1973	379	2307	32205	2199	4.91	36000	2032	13.53	36000	2031	13.59
36	ry48p.1	256	48	13135	36000	11965	9.78	36000	10739	22.31	36000	10764	22.03
37	ry48p.2	250	48	13802	36000	12065	14.39	36000	10912	26.48	36000	11000	25.47
38	ry48p.3	254	48	16540	36000	13085	26.40	36000	11732	40.98	36000	11822	39.91
39	ry48p.4	249	48	25977	36000	22084	17.62	18677	25037	3.75	**14001**	**25043**	**3.73**

Таблица 2.4

Результаты решения задач PCGTSP большой размерности

Кластеров	424				621			
Шаг, мм	25	50	100	1000	25	50	100	1000
Узлов	4023	2062	1134	431	4190	2253	1320	621
LB, мм	15818	17457	18561	25436	26132	27663	28349	34487
PCGLNS, мм	23374	24490	25421	34315	33522	34691	35410	45870
gap, %	47,77%	40,29%	36,96%	34,91%	28,28%	25,41%	24,91%	33,01%
CCP, мм	22609				32051			
GTSP / CCP	3,38%	8,32%	12,44%	51,78%	4,59%	8,24%	10,48%	43,12%

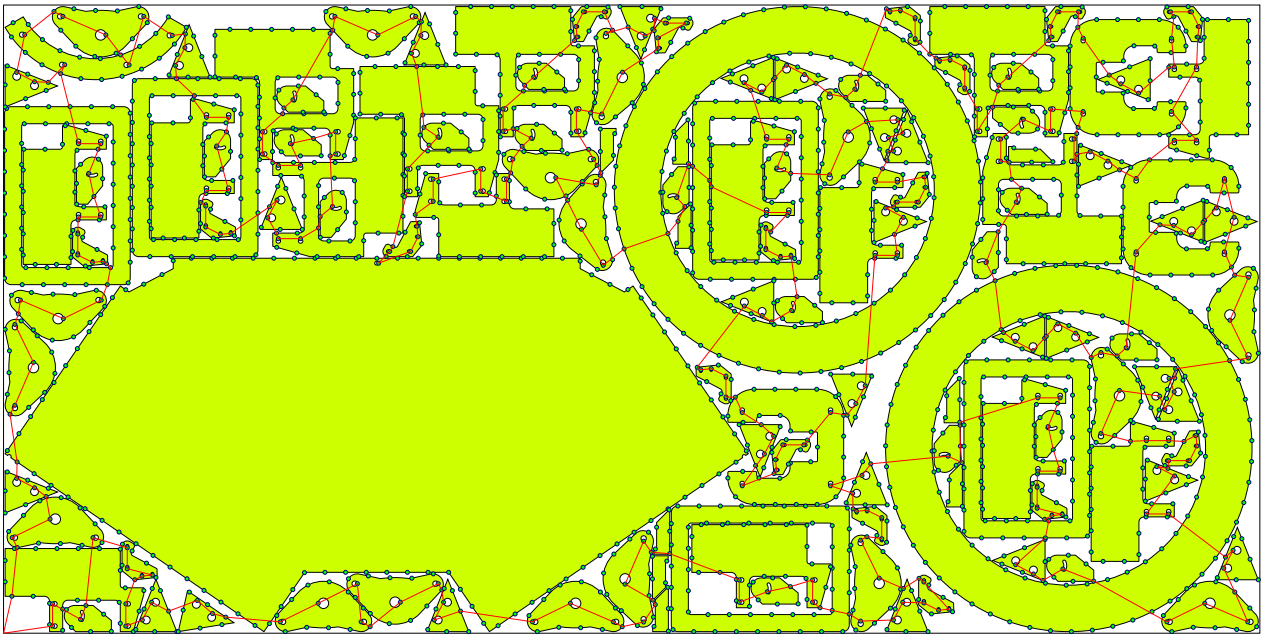
Для каждого раскройного плана строилось 4 разных экземпляра задачи PCGTSP за счёт дискретизации контуров деталей с разным шагом — от 25 мм до 1000 мм (в последнем случае фактически получалась задача TSP). Решения на рис. 2.2. получены при дискретизации с шагом 50 мм. Для каждого экземпляра приведены количество получившихся узлов, нижняя оценка, полученная описанный в данной главе алгоритмом, длина решения, полученного эвристикой PCGLNS и оценка его погрешности. Также приведены длины решения исходной непрерывной задачи CCP алгоритмом, описанным в главе 3 (см. Табл. 3.2 на стр. 76) и на сколько решение дискретной задачи превышает решение соответствующей непрерывной задачи.

2.6. Выводы по Главе 2

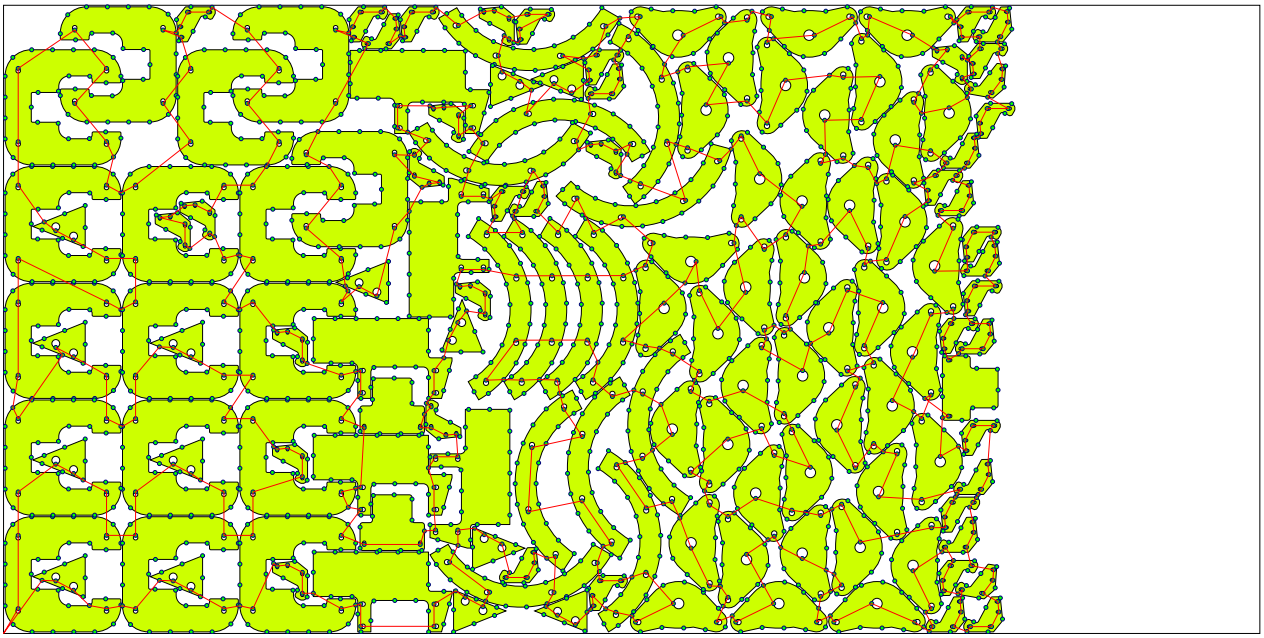
1. В ходе диссертационной работы разработан и реализован первый специализированный алгоритм ветвей и границ для обобщенной задачи коммивояжера с ограничениями предшествования PCGTSP, развивающий идеи классической схемы динамического программирования Хелда и Карпа и подход Салмана к построению нижних оценок.

2. Алгоритм разработан в двух версиях — классической схемы ветвей и границ и динамического программирования — немного отличающихся производительностью. Версия динамического программирования естественным образом допускает параллельное выполнение.

3. Для оценки производительности алгоритмов проведены численные эксперименты, в которых для сравнения использовался передовой коммерче-



а) $m = 424$ кластера



б) $m = 621$ кластер

Рис. 2.2. Решения задач PCGTSP большой размерности

ский солвер Gurobi. Эксперименты продемонстрировали конкурентоспособность предложенных алгоритмов.

4. В качестве направления дальнейших исследований предполагается:

   – разработка более точных нижних оценок.

   – дальнейшая оптимизация и распараллеливание

   – интеграция с системой автоматизированного проектирования «Сириус» [79], предназначенной для оптимизации раскроя листового материала на фигурные заготовки и подготовки управляющих программ для машин листовой резки с ЧПУ.

Глава 3. Применение непрерывно-дискретных оптимизационных моделей маршрутизации в эвристических алгоритмах решения задачи непрерывной резки ССР

3.1. Постановка задачи ССР

Рассмотрим евклидову плоскость $\mathbb{R}\times\mathbb{R}$ и на ней фигуру $\mathcal{B}$ (в большинстве практических случаев – прямоугольник), ограниченную замкнутым контуром. Это – модель листового материала, подлежащего резке. Пусть N попарно непересекающихся плоских контуров $\{C_1, C_2, ...C_N\}$ расположены внутри $\mathcal{B}$, ограничивая n деталей $\{A_1, A_2...A_n\}$. Деталь может быть ограничена одним или несколькими контурами (одним внешним и несколькими отверстиями), так что в общем случае $n \leqslant N$.

Контуры C_i 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. [15].

Далее, внутри $\mathcal{B}$ (как правило, на границе) выберем две точки и обозначим их 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 штук 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

$$\mathfrak{R} = \langle M_0, M_{i_1}, M_{i_2}, \dots M_{i_N}, M_{N+1}\rangle \tag{3.1}$$

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

$$\mathcal{L} = \sum_{j=0}^{N} |M_{i_j} M_{i_{j+1}}| \tag{3.2}$$
$$\mathcal{L} \to \min$$

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

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

Хотя контуры C_i по условию не пересекаются, они могут быть вложены друг в друга: $\widetilde{C}_a \subset \widetilde{C}_b$, где $\widetilde{C}_a$ обозначает 2-мерную фигуру, ограниченную контуром C_a (в более традиционных обозначениях $C_a = \partial\widetilde{C}_a$). 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: $\widetilde{C}_a \subset \widetilde{C}_b \Rightarrow i_a < i_b$, в перестановке $I = (i_1, i_2, ...i_N)$. Таким образом, множество допустимых перестановок ограничено.

3.2. Алгоритмы решения задачи непрерывной резки

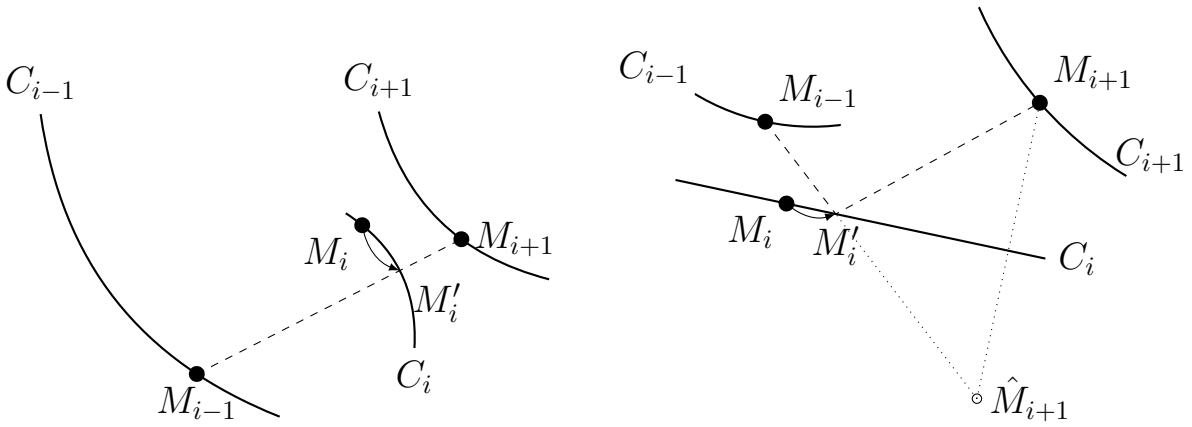
Предлагаемый алгоритм решения задачи непрерывной резки (см. [61; 105]) состоит из нескольких шагов, что хорошо соответствует самой природе решаемой задачи.

3.2.1. О приеме удаления «внешних» контуров для сокращения трудоемкости решения

Для автоматического соблюдения ограничения предшествования, мы начинаем с удаления всех контуров, внутри которых есть вложенные контура, так, чтобы остались только:

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

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



а) На пересечении звена ломаной б) С использованием *принципа Ферма*

Рис. 3.1. Оптимальное положение точки врезки

3.2.2. Поиск траектории перемещения инструмента для случая непрерывной модели описания геометрии контуров

На этом этапе мы полагаем, что последовательность обхода контуров $I = (i_1, i_2, ...i_N)$ задана (фиксирована) и ищем координаты точек врезки $M_i \in C_i$ во все контура, минимизируя полную длину холостого хода (3.2). Для этого, начальные позиции точек врезки выбираются произвольным образом (например, случайно) и затем положение одной (каждой) из точек M_i изменяется, а все остальные остаются неподвижны: $\mathcal{L}(M_i) \to \min$. Большинство слагаемых в целевой функции (3.2) при этом постоянны, так что сама функция упрощается до

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

Несложный геометрический анализ показывает, что если точки M_{i-1} и M_{i+1} расположены по разные стороны сегмента контура C_i, то оптимальное положение точки врезки M_i оказывается на пересечении с этим сегментом: $M_i = M_{i-1}M_{i+1} \cap C_i$ (если, конечно, такое пересечение существует; в противном случае решением будет один из концов сегмента), см. рис. 3.1а.

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. 3.1б.

Поиск позиции точки врезки выполняется для всех N контуров поочерёдно, и весь этот процесс повторяется пока длина холостого пути (3.2) не

стабилизируется с некоторой наперёд заданной точностью δ (на шаге 4), см. Алгоритм 3.1.

Алгоритм 3.1 CCP :: Релаксация позиций точек врезки

1: инициализация $M_i \in C_i, \forall i$ {Случайным образом}
2: $d \leftarrow 0$
3: $d_0 \leftarrow \infty$
4: **while** $|d - d_0| > \delta$ **do**
5: **for all** $i \in \overline{1, N}$ **do**
6: $M_i \leftarrow \arg\min_{M_i \in C_i} |M_{i-1}M_i| + |M_iM_{i+1}|$
7: $d_0 \leftarrow d$
8: $d \leftarrow \mathcal{L}(M_1, M_2, \ldots, M_N)$ {см. (3.2)}
9: **end for**
10: **end while**
11: **return** $\{M_1, M_2, \ldots, M_N\}$

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

3.2.3. Алгоритмы комбинаторной оптимизации построения маршрута

Наиболее вычислительно сложная задача заключается в поиске перестановки $I = (i_1, i_2, ...i_N)$, минимизирующей полную длину холостого хода $\mathcal{L} \to \min$. Фактически, это решение задачи коммивояжёра (Traveling Salesman Problem, TSP), только длина пути вычисляется не аддитивно, а при помощи процесса непрерывной оптимизации, описанного в разделе 3.2.2.

В данной работе для поиска такой перестановки применяется метод переменных окрестностей (Variable Neighborhood Search, VNS [25]), см. Алгоритм 3.2.

На шаге 4 многократно применяется непрерывная оптимизация из предыдущего этапа:

$$\mathcal{L}(I') = \min_{M_1, M_2 \ldots M_N} \mathcal{L}(M_1, M_2 \ldots M_N | I')$$

Окрестности $\mathcal{N}^k(I)$ различного размера конструируются разнообразными способами, например:

Алгоритм 3.2 CCP :: Дискретная оптимизация

Инициализация $I = (i_1, i_2, ...i_N)$ {выбирается случайным образом}
$k \leftarrow 1$
while $k < k_{max}$ **do**
  $I' = \arg\min_{I' \in \mathcal{N}^k(I)} \mathcal{L}(I')$
  if $\mathcal{L}(I') < \mathcal{L}(I)$ **then**
    $I \leftarrow I'$
    $k \leftarrow 1$
  else
    $k \leftarrow k + 1$
  end if
end while
return I

– 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 $I = (i_1, i_2, ...i_N)$ 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 $I = (i_1, i_2, ...i_N)$

– 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)

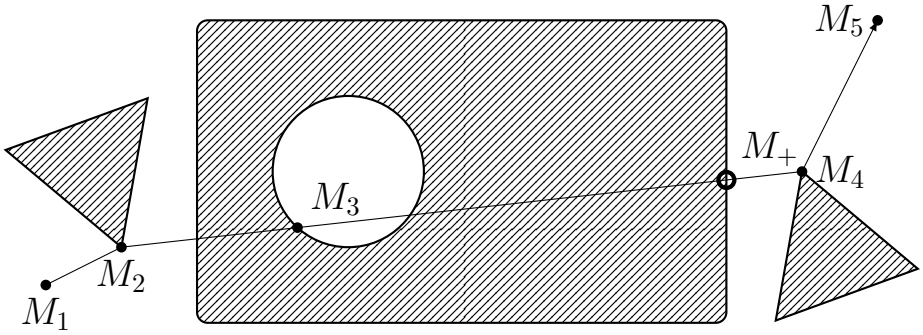


Рис. 3.2. Добавление точек врезки во «внешние» контура M_+

– «вставка» — точное решение оптимизационной задачи для одного блока в составе полной перестановки $I = (i_1, i_2, ...i_N)$

– И ещё порядка десяти других способов генерации «близких» к исходной перестановок

Если размер некоторой окрестности $\mathcal{N}^k(I)$, 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 4) на «первый подходящий» или метод Монте-Карло, но их влияние на скорость и качество получаемого решения задачи непрерывной резки требует дальнейшего исследования.

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

$$\mathfrak{R} = \langle M_0, M_1, M_2, \dots M_{N'}, M_{N'+1} \rangle \tag{3.3}$$

обязательно пересекает все исходные контуры C_i, 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 M_{N+1} всегда выбираются снаружи всех контуров C_i.

Таким образом, для каждого «внешнего» контура C_i, который пока не включён в маршрут резки, мы находим все точки пересечения с полученным маршрутом (3.3), и если таких точек несколько (что как правило и бывает), выбираем из них самую последнюю, то есть посещаемую маршрутом позже всех других, см. рис. 3.2. Сам контур C_i 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, лежащие 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 $O(N)$.

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

Экстремальные свойства получаемого решения

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.

На рис. 3.3 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.

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.

Итак, рассмотрим следующую задачу: пусть контуры C_i 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 C_i мы повторяем следующую операцию: сдвигаем вершину ломаной $M_i \in C_i$ в такое положение, которое минимизирует полную длину ломаной, при этом остальные вершины M_j $(j \neq i)$ неподвижны. Это сводится к несложной геометрической задаче нахождения точки, для которой сумма расстояний до двух других фиксированных точек минимальна.

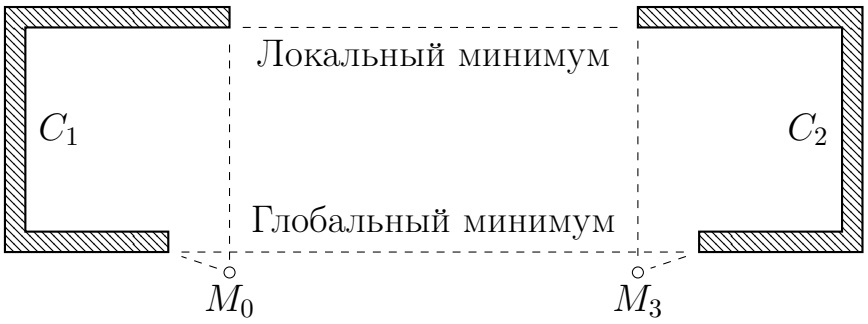


Рис. 3.3. Два маршрута резки, доставляющие локальный и глобальный минимум

В процессе таких пошаговых сдвигов мы получаем последовательность ломаных линий $\{L_k\}$, причём последовательность длин этих ломаных линий по построению монотонно убывает. Обозначим $m = \inf |L_k|$ (точная нижняя граница длин ломаных линий) и пусть L_* – ломаная линия длины m, то есть предельная точка последовательности ломаных линий $\{L_k\}$. 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

Утверждение 3.1. *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.*

Доказательство. Рассмотрим сдвиг *двух* соседних вершин. Обозначим четыре последовательных вершины ломаной L_* за $M_{i-1}, M_i, M_{i+1}, M_{i+2}$.

Пусть точки $M_i \in S_i, M_{i+1} \in S_{i+1}$ лежат на прямолинейных сегментах соответствующих контуров $S_i \subset C_i, S_{i+1} \subset C_{i+1}$.

Докажем, что:

$$\forall M'_i \in S_i, M'_{i+1} \in S_{i+1} : |M_{i-1}M'_iM'_{i+1}M_{i+2}| \geqslant |M_{i-1}M_iM_{i+1}M_{i+2}|$$

Для произвольной вершины $M'_i \in S_i$: $|M_{i-1}M'_iM_{i+1}M_{i+2}|$ минимально, когда $M'_i = M_i$, и аналогично для произвольной вершины $M'_{i+1} \in S_{i+1}$: $|M_{i-1}M_iM'_{i+1}M_{i+2}|$ минимально, когда $M'_{i+1} = M_{i+1}$.

Предположим, что

$$\exists M'_i \in S_i, \exists M'_{i+1} \in S_{i+1} : |M_{i-1}M'_iM'_{i+1}M_{i+2}| < |M_{i-1}M_iM_{i+1}M_{i+2}|$$

Очевидно, что $M'_i \neq M_i, M'_{i+1} \neq M_{i+1}$.

Введём обозначение $M_i(s) = M_i + s \cdot \overrightarrow{M_iM'_i}$, $M_{i+1}(t) = M_{i+1} + t \cdot \overrightarrow{M_{i+1}M'_{i+1}}$ $(s, t \in [0, 1])$, $f(s, t) = |M_{i-1}M_i(s)M_{i+1}(t)M_{i+2}|$.

Но положения вершин M_i and M_{i+1} выбраны так, что:

$$\frac{\partial f(s,t)}{\partial s}\Big|_{(0,0)} \geqslant 0, \frac{\partial f(s,t)}{\partial t}\Big|_{(0,0)} \geqslant 0 \tag{3.4}$$

Если, например, $\partial f(s,t)/\partial s\big|_{(0,0)} = 0$, то это может быть только если точки M_{i-1}, M_i, M_{i+1} лежат на одной прямой (когда M_{i-1} и M_{i+1} по одну сторону от S_i; в противном случае заменяем M_{i-1} на её отражение относительно S_i). Таким образом, если одновременно

$$\frac{\partial f(s,t)}{\partial s}\Big|_{(0,0)} = 0 = \frac{\partial f(s,t)}{\partial t}\Big|_{(0,0)}$$

то значит все четыре точки $M_{i-1}, M_i, M_{i+1}, M_{i+2}$ 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 (3.4) не равна нулю. Обозначим $\varphi(t) = f(t,t)$.

$$\frac{d\varphi}{dt}\Big|_0 = \frac{\partial f(s,t)}{\partial s}\Big|_{(0,0)} + \frac{\partial f(s,t)}{\partial t}\Big|_{(0,0)} > 0$$

Это значит, что $\exists \tau^* \in [0,1]$: $\varphi(\tau^*) > \varphi(0)$. Но по нашему предположению $\varphi(1) < \varphi(0)$, то есть $\varphi(0) < \varphi(\tau^*) > \varphi(1)$.

Теперь заметим, что $\varphi(t)$ представляет собой сумму четырёх слагаемых вида $\sqrt{(a+b\cdot t)^2 + (c+d\cdot t)^2}$, и каждое из этих слагаемых имеет положительную вторую производную, так что и $d^2\varphi(t)/dt^2 \geqslant 0$, а значит функция $\varphi(t)$ является выпуклой на $[0,1]$ и не может принимать во внутренней точке интервала значения большие, чем на его концах.

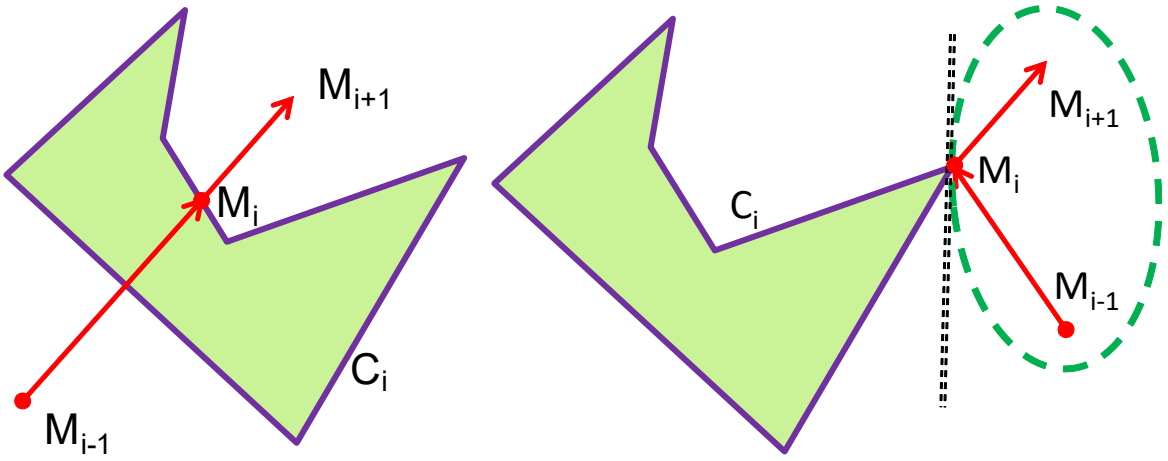
Значит, наше предположение невозможно.

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. □

Глобальный минимум

Перейдём к *достаточному* условию того, что ломаная L_* является глобальным минимумом.



а) Маршрут пересекает контур б) Касательная отделяет контур

Рис. 3.4. Достаточные условия глобального минимума

Пусть $M_i \in C_i$ – вершина ломаной L_*. Соседние вершины M_{i-1} и M_{i+1} находятся вне C_i, поскольку по условию задачи мы исключили вложенные контуры По построению L_*: $\forall M_i' \in C_i : |M_{i-1}M_i|+|M_iM_{i+1}| \leqslant |M_{i-1}M_i'|+|M_i'M_{i+1}|$.

Условие 3.1. Пусть выполняется **одно** из следующих условий (см. рис. 3.4):

1. Сегмент $M_{i-1}M_{i+1}$ пересекает контур C_i, то есть $M_i \in M_{i-1}M_{i+1}$, рис. 3.4а
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, рис. 3.4б

Утверждение 3.2. *Если условие 3.1 выполняется для (всех контуров) 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.*

Доказательство. Используя те же обозначения, рассмотрим четыре соседних вершины $M_{i-1}, M_i, M_{i+1}, M_{i+2} \in L_*$. $M_i \in C_i, M_{i+1} \in C_{i+1}$.

Предположим, что

$$\exists M_i' \in C_i, \exists M_{i+1}' \in C_{i+1} : |M_{i-1}M_i'M_{i+1}'M_{i+2}| < |M_{i-1}M_iM_{i+1}M_{i+2}|$$

Снова, $M_i' \neq M_i, M_{i+1}' \neq M_{i+1}$.

Обозначим $M_i(s) = M_i + s \cdot \overrightarrow{M_iM_i'}$, $M_{i+1}(t) = M_{i+1} + t \cdot \overrightarrow{M_{i+1}M_{i+1}'}$ $(s,t \in [0,1])$, $f(s,t) = |M_{i-1}M_i(s)M_{i+1}(t)M_{i+2}|$.

Но теперь условие 3.1 гарантирует, что $f(s,0) \geqslant f(0,0)$ для $s \in [0,1]$, то есть снова

$$\frac{\partial f(s,t)}{\partial s}\Big|_{(0,0)} \geqslant 0, \frac{\partial f(s,t)}{\partial t}\Big|_{(0,0)} \geqslant 0 \tag{3.5}$$

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.

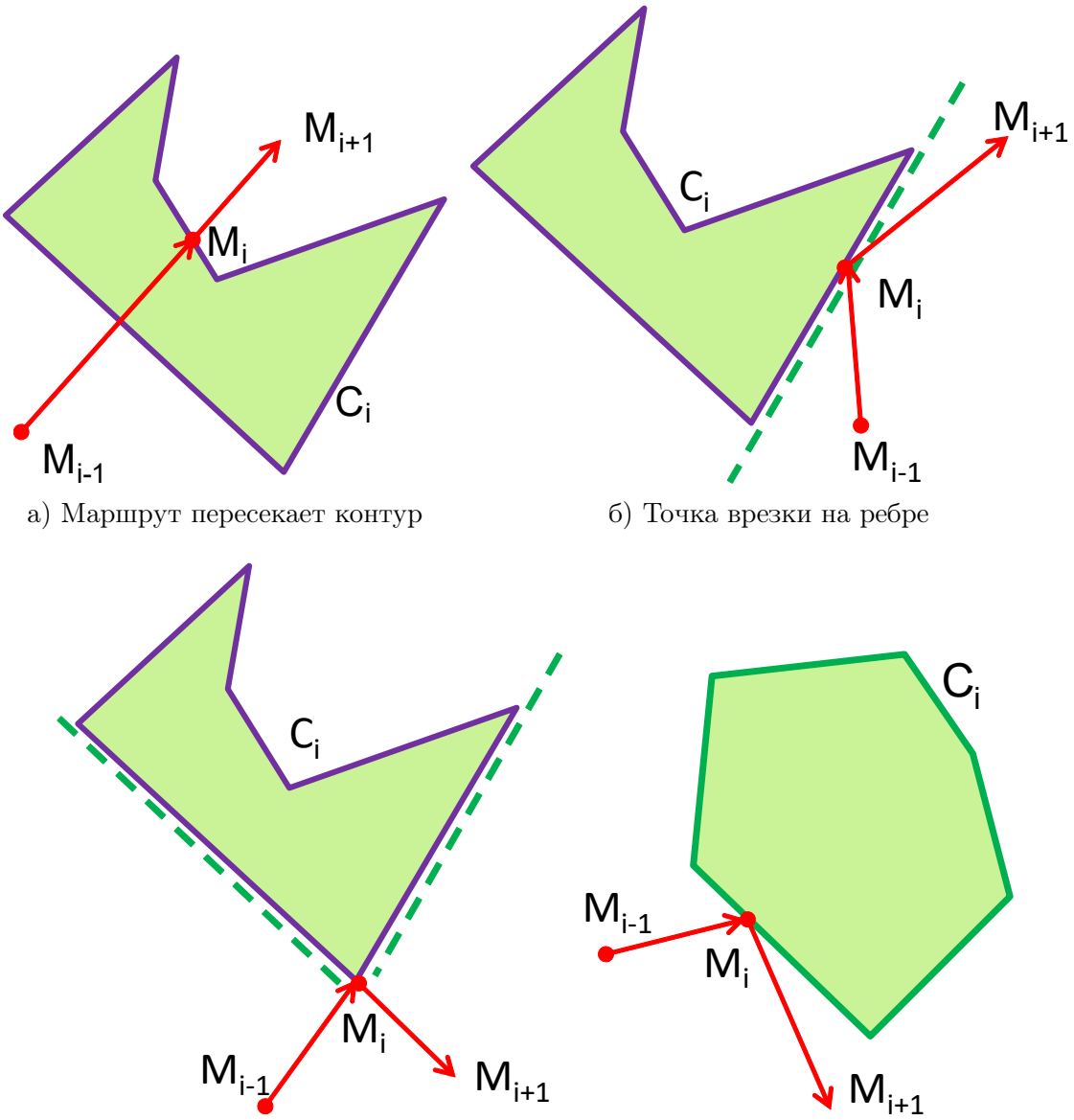
Условие 3.1 легко проверяется программно, однако его можно ещё упростить с тем, чтобы в большинстве практических случаев его можно было проверить просто визуально, буквально «на глаз».

Условие 3.2. Выполняется **любое** из условий (см. рис. 3.5):

1. Сегмент $M_{i-1}M_{i+1}$ пересекает контур C_i: $M_i \in M_{i-1}M_{i+1}$, рис. 3.5а
2. Если 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 (это и есть касательная, которую использует условие 3.1; иначе существовала бы лучшая вершина $M_i' \in C_i$), рис. 3.5б
3. Если 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, рис. 3.5в
4. Если контур C_i ограничивает собой выпуклый многоугольник $\widetilde{C}_i$, рис. 3.5г

3.3. Численные эксперименты

Для оценки качества решений, получаемых описанным эвристическим алгоритмом, использовались несколько раскройных планов, содержащих реальные детали. В качестве базы сравнения использовался алгоритм [8], решающий задачу GTSP и дающий точное решение для количества контуров $N \leqslant 33$.



а) Маршрут пересекает контур

б) Точка врезки на ребре

в) Врезка в вершину контура

г) Выпуклый контур

Рис. 3.5. Ослабленное условие глобального минимума

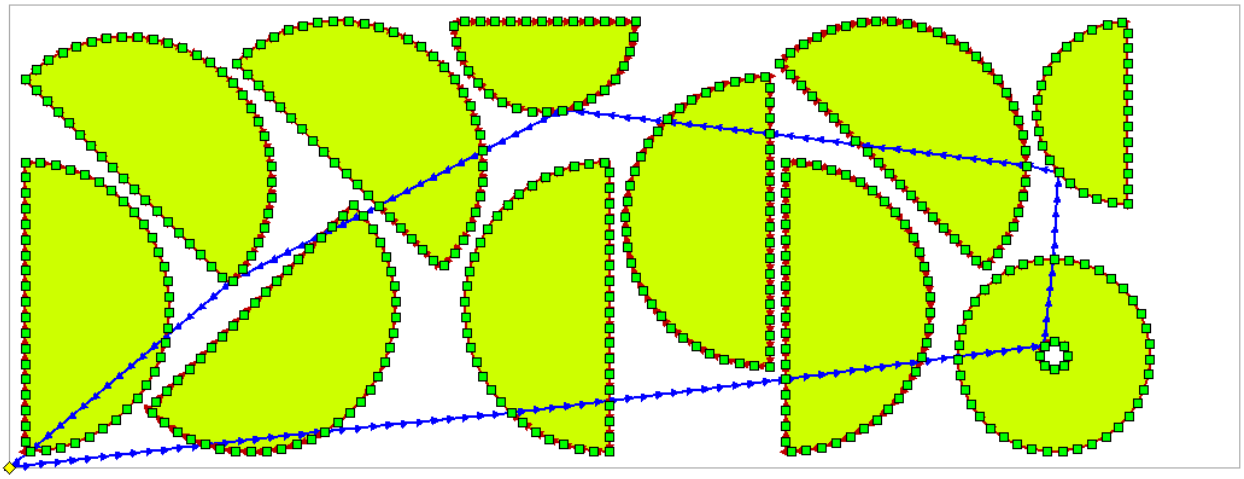
Результаты экспериментов сведены в Табл. 3.1, полученные решения приведены на рис. 3.6, рис. 3.7, рис. 3.8 и рис. 3.9.

Видно, что оба алгоритма дают практически идентичные маршруты резки. Основное отличие вызвано необходимостью дискретизации контуров в ходе сведения задачи непрерывной резки к GTSP. Характерной особенностью решения задачи непрерывной резки является то, что оно содержит много прямолинейных сегментов, разделяемых точками врезки, но фактически лежащих на одной прямой. Аналогичные сегменты решения задачи GTSP имеют небольшие изломы, поскольку возможные координаты точек резки фиксированы заранее и не могут попасть на одну прямую. Поэтому общая длина холостого хода в

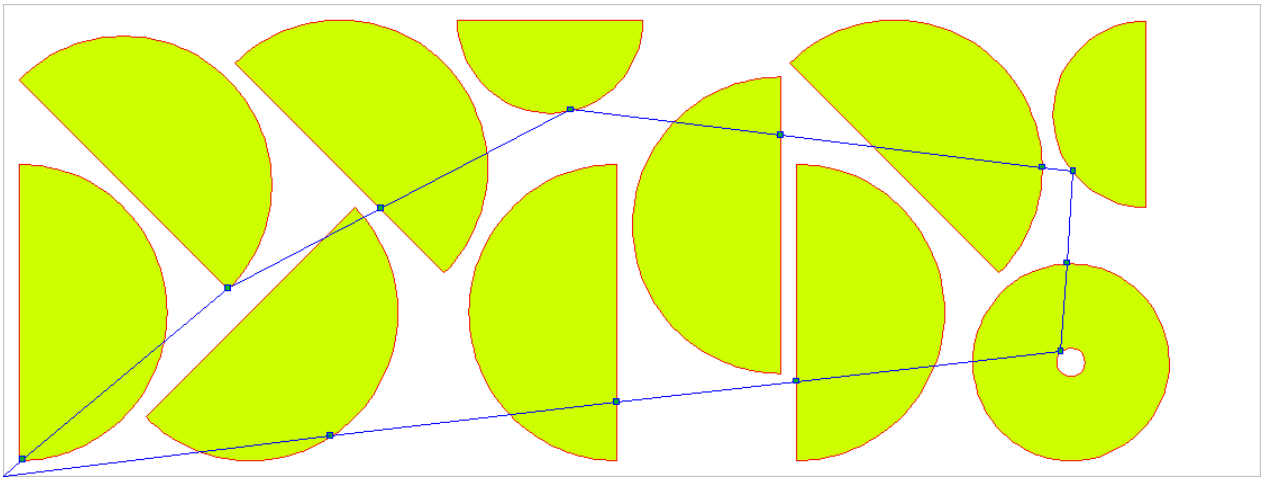
Таблица 3.1

Сравнение качества решений задач CCP и GTSP

Задание	№ 229	№ 464	№ 3211	№ 20205
Кол-во деталей	11	14	17	115
Кол-во контуров	12	21	22	198
Общий периметр, м	24.609	21.717	25.051	143.467
Кол-во точек GTSP	491	429	493	3917
$\mathcal{L}_{GTSP}$, м	7.729	4.743	4.557	26.098
$\mathcal{L}_{CCP}$, м	7.727	4.706	4.536	25.987
См.	Рис. 3.6	Рис. 3.7	Рис. 3.8	Рис. 3.9

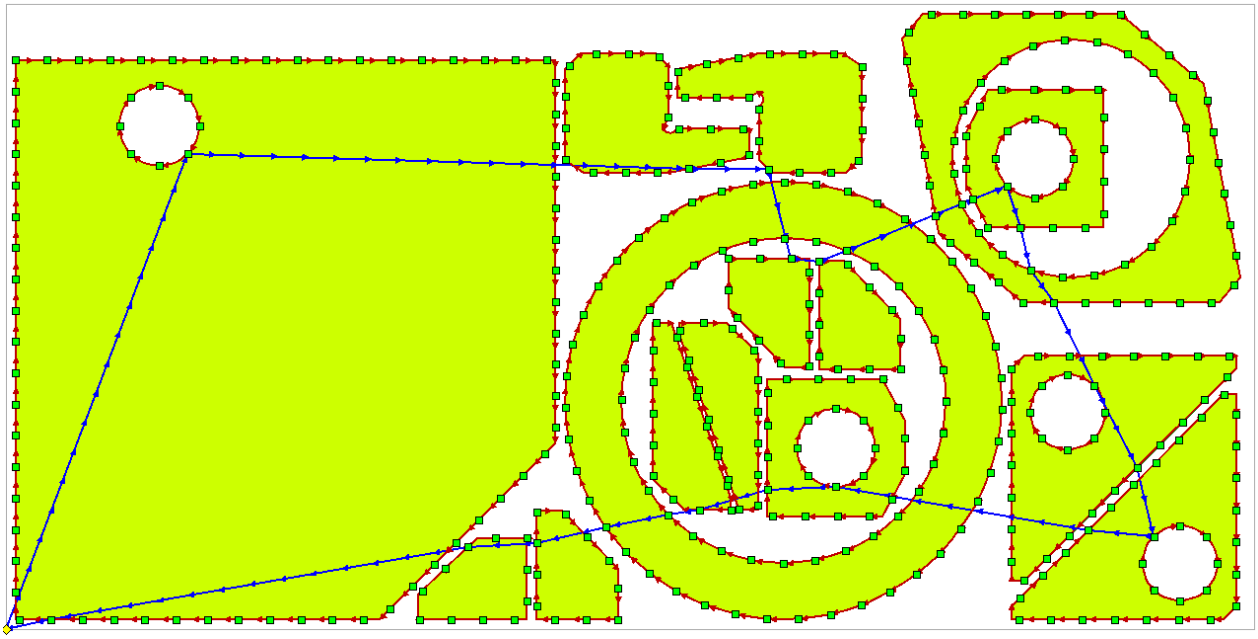


а) Точное решение задачи GTSP

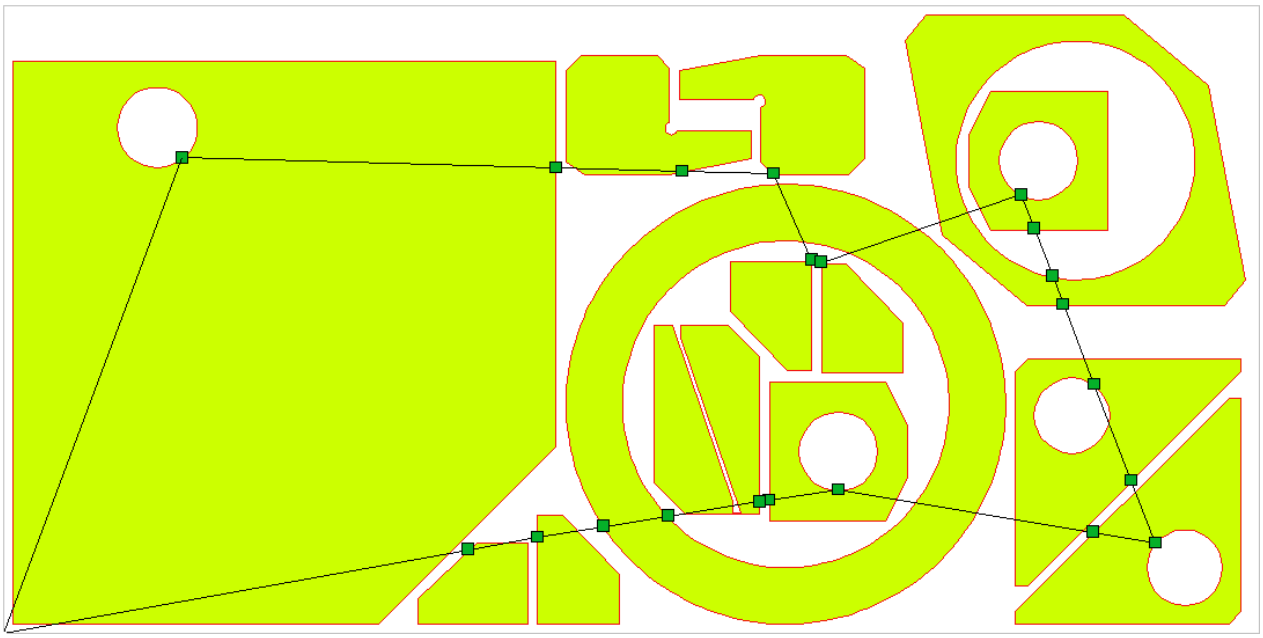


б) Решение задачи CCP

Рис. 3.6. Решения задач резки для задания № 229

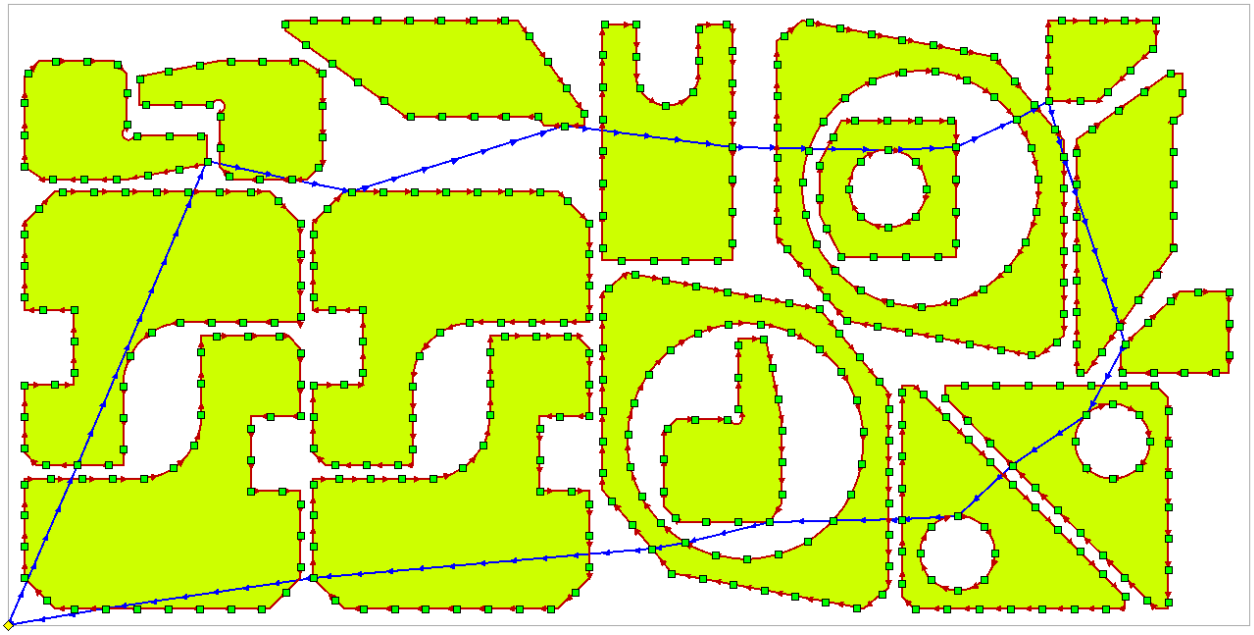


а) Точное решение задачи GTSP

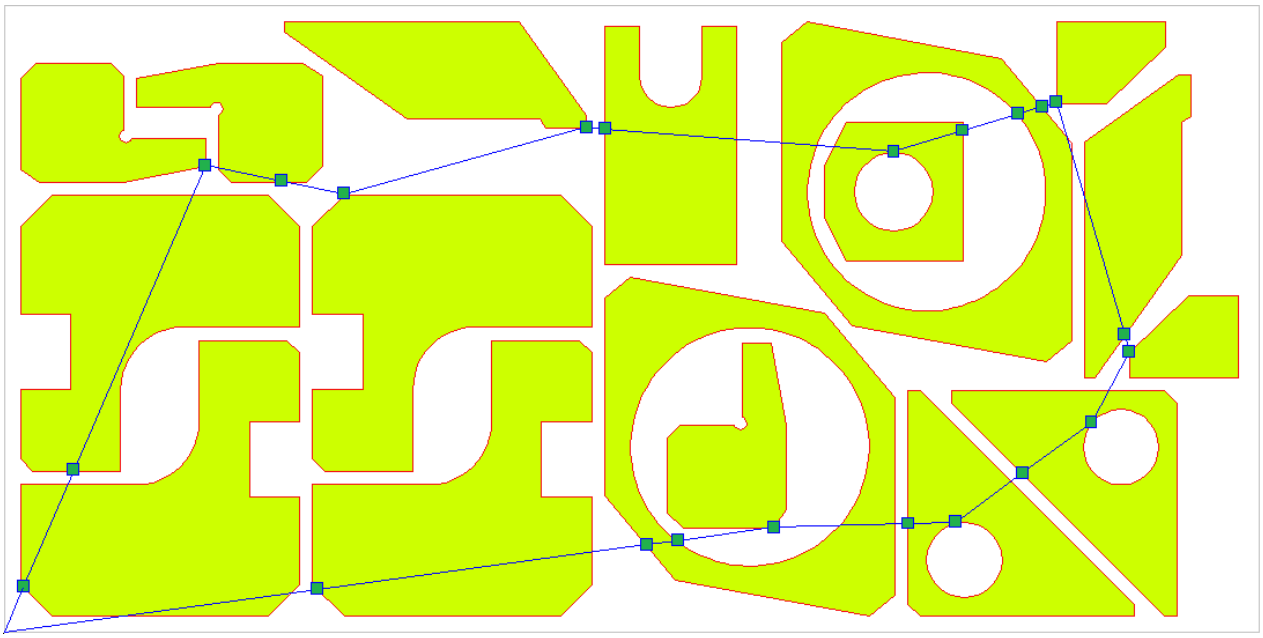


б) Решение задачи CCP

Рис. 3.7. Решения задач резки для задания № 464



а) Точное решение задачи GTSP



б) Решение задачи CCP

Рис. 3.8. Решения задач резки для задания № 3211

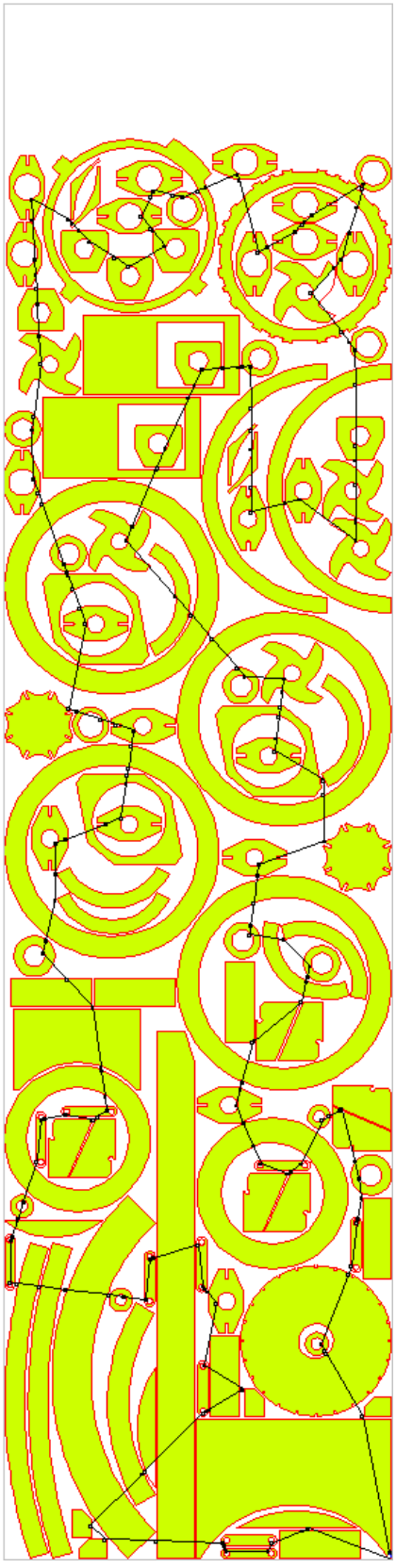


Рис. 3.9. Пример решения задачи ССР большого размера, задание № 20205

общем случае получается чуть больше, чем для «честного» решения задачи непрерывной резки, что и отображено в Табл. 3.1.

Интересно, что Условие 3.1 на стр. 69 соблюдено для всех контуров на рис. 3.7б, то есть изображённое там решение действительно представляет собой глобальный минимум (длины холостого хода). В то же время, более простое Условие 3.2 на стр. 70 соблюдено для *почти* всех контуров, кроме одного (невыпуклой детали сверху по центру). Это довольно редкая с практической точки зрения ситуация, тем не менее, она показывает, что две формулировки достаточного условия глобальности минимума не эквивалентны.

Решение задач большой размерности

Интересно сравнить решения, полученные в Главе 2 (см. рис. 2.2 на стр. 57)

с решениями задачи непрерывной резки для тех же раскройных планов, содержащих сотни контуров.

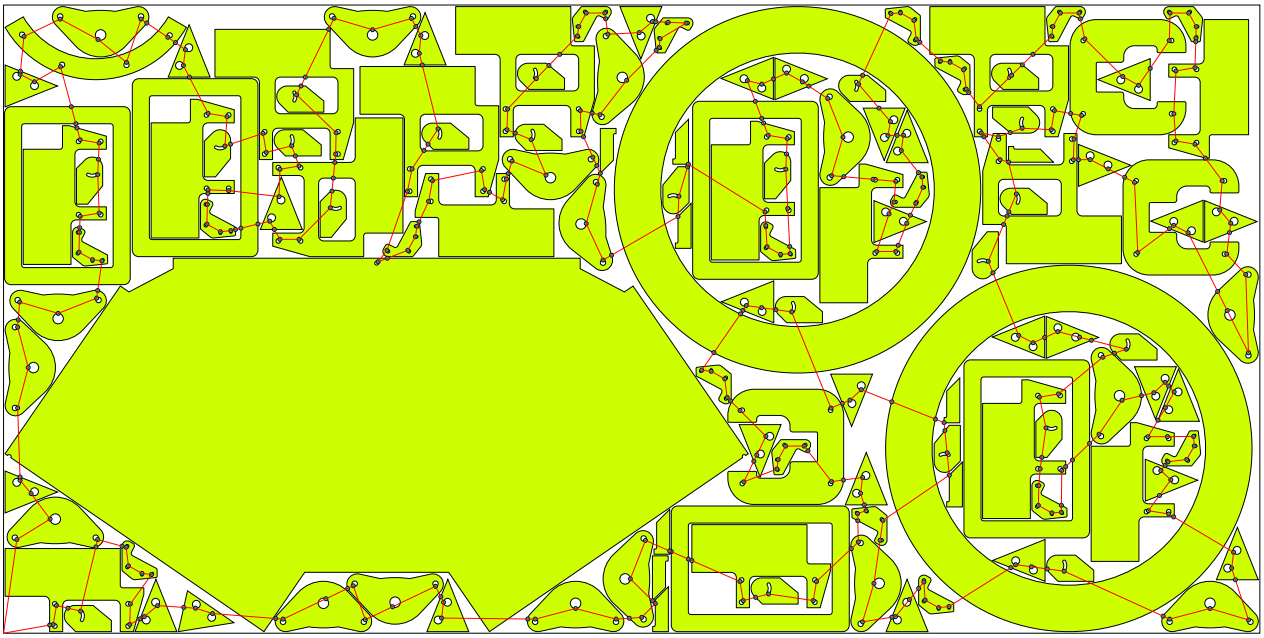
Решения, не использующие механизм дискретизации, приведены на рис. 3.10, численные данные — в табл. 3.2.

Таблица 3.2

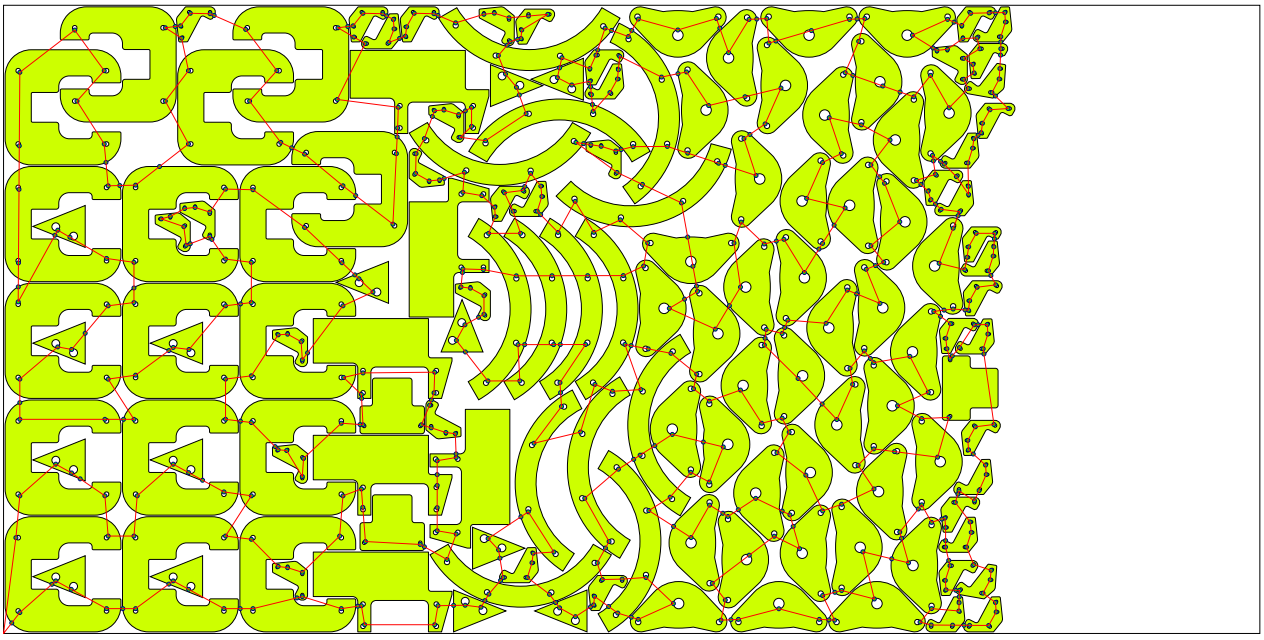
Результаты решения задач CCP большой размерности

Деталей	Контуров	Вложенность	$\mathcal{L}$, мм
125	423	6	22609
140	620	2	32051

Несмотря на то, что решения разных задач нельзя сравнивать напрямую, тем не менее полезно сопоставить их, особенно с учетом того, что для задач такой размерности пока невозможно точно судить об оптимальности решения. Очевидное сходство решений, полученных разными методами позволяет осторожно предполагать, близость обоих решений к оптимальным для своего класса задач. Как и в более простых примерах, решения задачи CCP оказываются чуть лучше, чем решения задачи PCGTSP (подробности в Табл. 2.4 на стр. 56), но разница становится больше и на приведенных примерах достигает $\approx 8\%$, что отчасти объясняется тем, что на рис. 2.2 применялась более грубая дискретизация с шагом 50 мм (ввиду большего размера задачи).



а) 125 деталей; 423 контура



б) 140 деталей; 620 контуров

Рис. 3.10. Решения задач ССР большой размерности

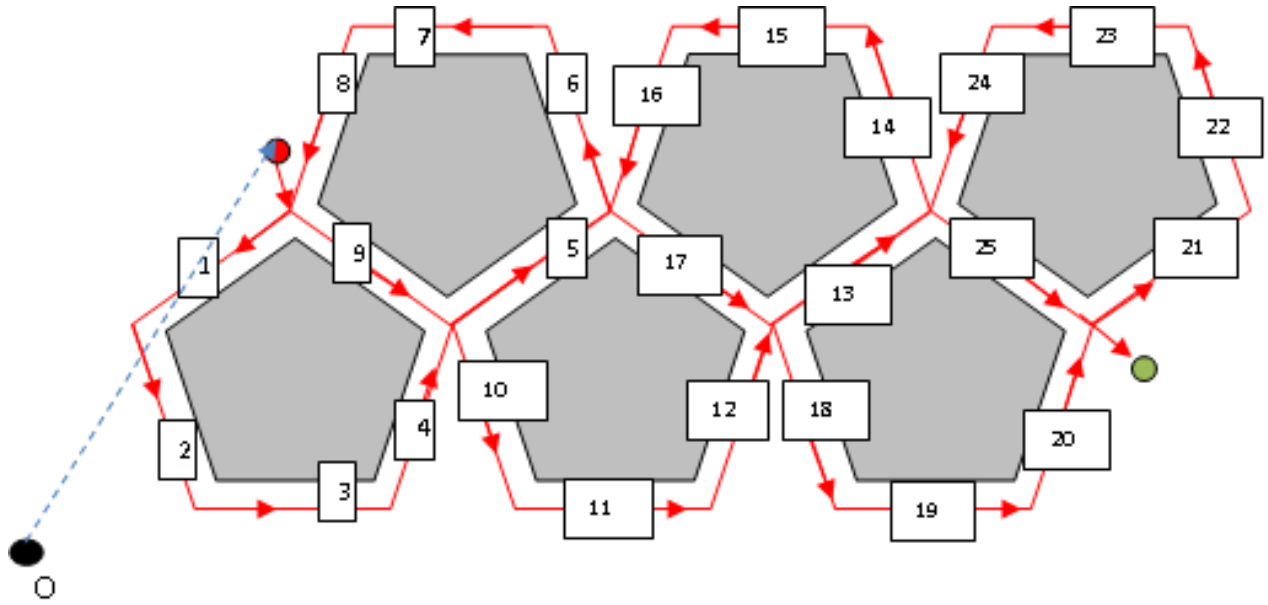


Рис. 3.11. Пример составного сегмента резки, содержащего 6 контуров (деталей)

3.4. Обобщение на задачи сегментной резки SCCP / GSCCP

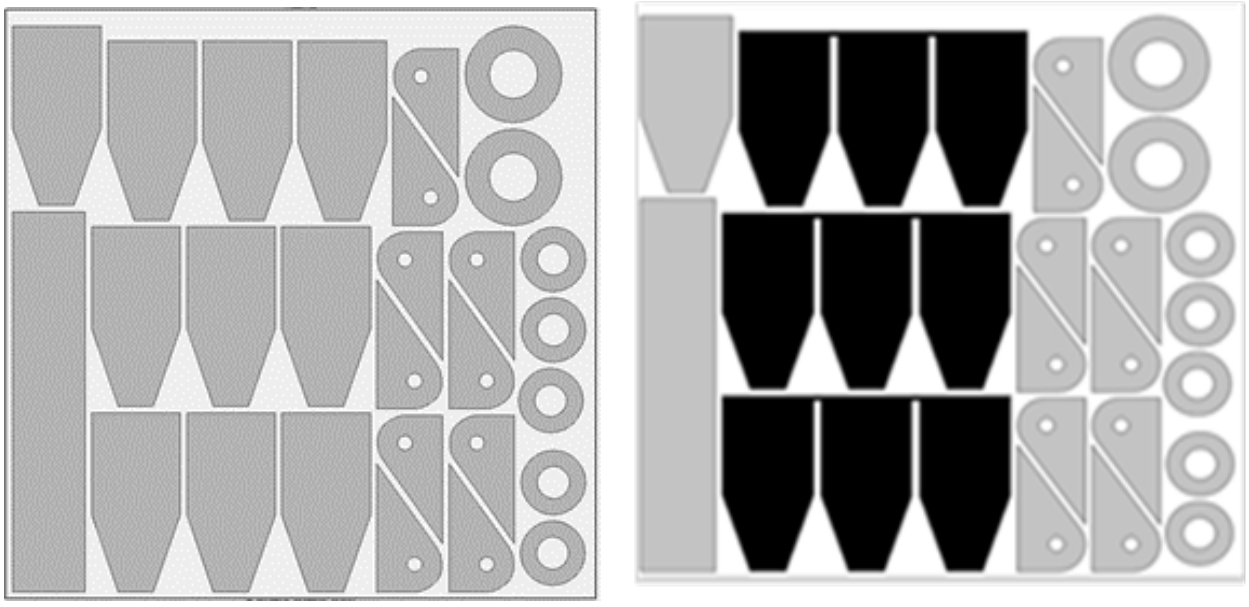
Описанный выше алгоритм разрабатывался и применяется для решения задачи непрерывной резки CCP, однако последняя представляет собой весьма частный случай самой общей формулировки задачи оптимальной маршрутизации режущего инструмента, на данный момент это задача прерывистой резки (Intermittent Cutting Problem, ICP). Она всё ещё слабо исследована и представляет существенный научный интерес как с теоретической точки зрения, так и в смысле практического использования.

Тогда как задача непрерывной резки возникает фактически при использовании так называемой *стандартной техники резки*, существуют и более сложные, прежде всего *мультисегментная* и *мультиконтурная* техники резки, см. раздел 1.1 Пример мультиконтурной резки показан на рис. 3.11.

Для включения этих техник в исследование, понятие контура детали было обобщено [64] и расширено до *сегмента резки*, см. Определение 1.1.

Фактически, пример мультиконтурной резки на рис. 3.11 также может представлять собой один единственный сегмент резки в составе некоторой большой задачи.

Ввиду того, что сегмент резки по определению содержит в себе направление резки (от точки врезки M до точки выключения инструмента M^*), нам потребуется ещё более общее понятие:



а) Стандартная резка, 45 сегментов б) Мульти-контурная резка, 39 сегментов

Рис. 3.12. Ансамбль задач сегментной резки

Определение 3.1. **Базовый сегмент резки** B^S — это часть сегмента резки $S = \overrightarrow{MM^*}$ без участков входа и выхода (lead-in и lead-out).

Базовый сегмент содержит только геометрию (части) контуров, подлежащих резке, и не содержит информации о направлении резки.

При помощи понятия базового сегмента формулируется обобщение задачи непрерывной резки:

Определение 3.2. **Задача непрерывной сегментной резки** (Segment Continuous Cutting Problem, $SCCP$) — это задача резки для фиксированного набора базовых сегментов резки: $SCCP = \left\{B^{S_i}\right\}$.

Описанный в данной главе алгоритм, разработанный для решения задачи непрерывной резки, естественным образом обобщается на решение задачи непрерывной сегментной резки [60; 107].

Далее, для произвольного раскройного плана (то есть фиксированного расположения деталей A_i и контуров C_i на листе $\mathcal{B}$), в общем случае можно сгенерировать целый *ансамбль* наборов базовых сегментов B^{S_i}, отвечающих заданным контурам деталей C_i. Например, для раскроя на рис. 3.12а можно построить задачу $SCCP$ меньшего размера за счёт объединения некоторых контуров в базовые сегменты, как показано на рис. 3.12б. Это наблюдение приводит нас к ещё более общей задаче резки:

Определение 3.3. **Обобщённая задача непрерывной сегментной резки** (Generalized Segment Continuous Cutting Problem, $GSCCP$) — это ансамбль из нескольких задач $SCCP$, полученных из одного раскройного плана: $GSCCP = \{SCCP_i\}$.

Новые классы $SCCP$ и $GSCCP$ значительно расширяют существующую классификацию задач резки для машин термической резки с ЧПУ. Фактически они представляют собой подклассы наиболее общей задачи ICP, состоящие из конечного набора базовых сегментов резки.

$$CCP \subset SCCP \subset GSCCP \subset ICP$$

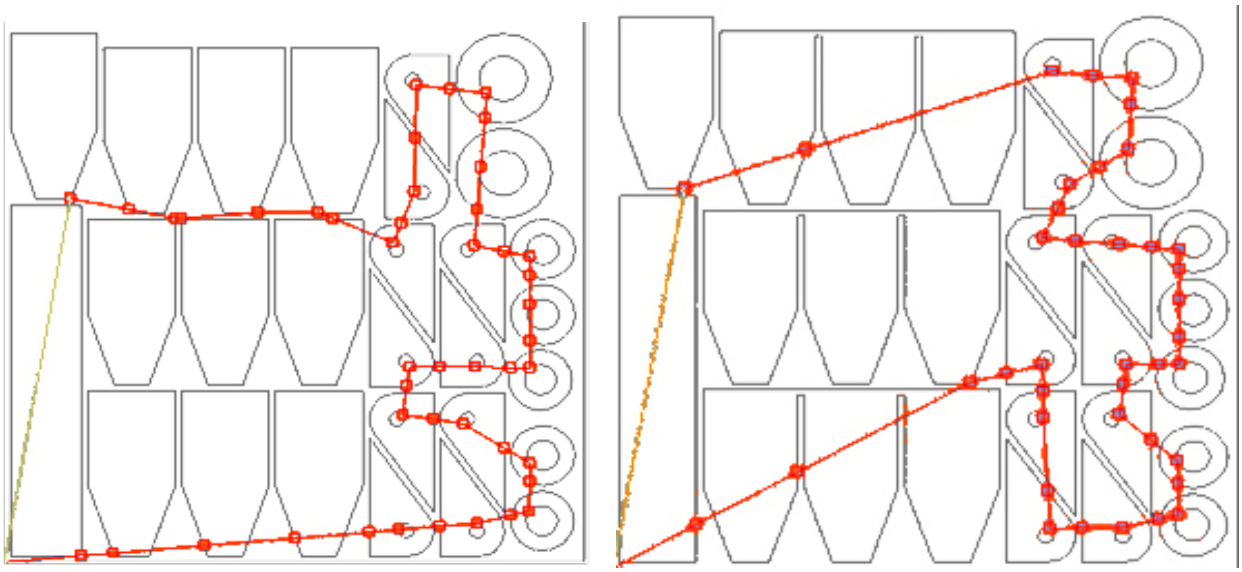
3.4.1. Общая схема решения задачи GSCCP

Считая заданным ансамбль $\{SCCP_i\}$ наборов базовых сегментов $SCCP_i = \left\{B^{S_j}\right\}$, $i \in \overline{1,T}, j \in \overline{1,K_i}$, будем решать задачу $GSCCP$ следующим образом:

- Каждая задача $SCCP_i$ решается независимо одним из существующих алгоритмов, например:
  - Описанный выше эвристический алгоритм решения задачи непрерывной резки, см. Главу 3.
  - Алгоритм ветвей и границ для обобщённой задачи коммивояжера с ограничениями предшествования, см. Главу 2.
  - Алгоритм на основе динамического программирования, дающий точное решение для задач ограниченного размера, см. [8].
  - Итеративный жадный эвристический алгоритм, см. [59]
  - ...

  Для дискретных алгоритмов предварительно проводится дискретизация контуров и построение конечного множества допустимых точек врезки.
- Выбирается лучшее решение, минимизирующее целевую функцию (3.2).

Пример решения задачи $GSCCP$, представленной на рис. 3.12, приведён на рис. 3.13. Использован алгоритм решения задачи непрерывной резки (без



а) Стандартная резка б) Мульти-контурная резка

Рис. 3.13. Решение задачи GSCCP на рис. 3.12

применения дискретизации контуров деталей). Видно, что для разных постановок задач $SCCP$ действительно получаются разные маршруты движения режущего инструмента. Более того, на практике различие оказывается ещё более значительным, поскольку получаемые решения отличаются также количеством точек врезки, а эта операция обычно вносит существенные расходы, как в смысле стоимости резки, так и затрачиваемого времени.

3.5. Выводы по Главе 3

1. Разработана и реализована оригинальная эвристика поиска оптимальных положений точек врезки на контурах деталей.

2. Эта эвристика может сочетаться с алгоритмами дискретной оптимизации для получения полного алгоритма решения задачи непрерывной резки CCP

3. Полученные решения оказываются вполне сравнимы с решениями, получаемыми более традиционными методами, путем сведения задачи CCP к задаче GTSP за счёт дискретизации контуров деталей. Более того, длина холостого хода для решений, даваемых разработанным алгоритмом, систематически оказывается чуть лучше, чем для ранее использовавшихся алгоритмов.

4. Описанный алгоритм может также применяться для решения задач более высокого класса — SCCP и GSCCP (непрерывной сегментной резки), что открывает дорогу к исследованиям в области задачи прерывистой резки (ICP).

5. Перспективными представляются также следующие направления исследований:

   – Использование разработанной эвристики в сочетании с другими алгоритмами дискретной оптимизации

   – Учёт других технологических ограничений термической резки, кроме ограничения предшествования

Глава 4. Методология использования алгоритмов решения задачи оптимальной маршрутизации режущего инструмента в CAD/CAM-системах

Основная сложность интеграции новых алгоритмов оптимальной маршрутизации режущего инструмента для машин листовой термической резки с ЧПУ, включая описанные выше в данной диссертационной работе, является необходимость взаимодействия большого количества программных подсистем, выполняющих разные задачи — геометрическое проектирование деталей и листов, раскрой, собственно резка по полученной раскройной карте, генерация УП, визуализация всех этапов проектирования и т.п. Все эти подсистемы писались и пишутся в разное время, командами разной квалификации, с разными целями, по разным принципам, иногда «на скорую руку», на разных языках программирования, для разных платформ, включая разные операционные системы.

Такое положение дел приводит, среди прочего к тому, что потоки данных между подсистемами устроены хаотически, используют всевозможные форматы данных, от простейших, созданных *ad hoc*, до сложнейших, например DXF (Drawing Interchange / Exchange Format) от Autodesk [16] или XML (eXtensible Markup Language) [17]. Точные спецификации форматов файлов могут быть утеряны (или даже никогда не созданы) или наоборот — оказываются слишком сложными, для того, чтобы быстро написать код, осуществляющий чтение или корректную запись такого файла.

Кроме того, это приводит к необходимости дублирования усилий, когда разным командам приходится писать код для чтения или записи одного и того же формата, зачастую на разных языках программирования.

4.1. Использование открытых форматов файлов данных для взаимодействия подсистем

Хотя задача полной унификации используемых форматов файлов вряд ли может быть *полностью* решена для современных крупных программных систем, тем не менее, в последние десятилетия развития технологий разработки программного обеспечения накоплены подходы, позволяющие значительно снизить остроту проблем, прежде всего за счет продуманного использования

открытых форматов хранения и обмена данными. При этом важны как функциональность формата, то есть то, какие именно данные он содержит, так и организация, то есть представление хранимых данных.

4.1.1. Выбор открытого формата представления геометрической информации

Например, для хранения и обмена геометрической информацией в САПР «Сириус» [104] используется унаследованный двоичный формат DBS [116], см. Приложение В. Он обладает важными достоинствами:

– эффективное хранение больших объемов геометрической информации за счет хранения массивов вещественных чисел в формате IEEE 754 [29] float32;

– 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, его недостатки тоже оказываются значительны, практически блокируя его использование для обмена информацией между программным обеспечением разных команд разработчиков. 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 проводилось исследование возможности замены

формата DBS на другой формат хранения геометрической информации. При этом предъявлялись следующие требования:

- Текстовый формат для упрощения обработки на актуальных языках программирования
- Простота чтения и записи
- Возможность хранения геометрической информации, содержащейся в DBS
- Самодокументируемость: возможность прочитать файл даже без знания спецификации
- Расширяемость: возможность расширить состав хранимой информации прозрачным для существующих программ образом

В результате сравнения нескольких наиболее популярных открытых форматов (JSON [31], YAML [88], XML [17], TOML [82], Tree [83], ...) был выбран формат JavaScript Object Notation (JSON [31]). К его преимуществам можно отнести:

- очень прост: всего ≈ 30 правил в грамматике,
- готовые библиотеки для чтения и записи для практически всех современных языков,
- может формироваться даже без использования специализированных библиотек,
- является стандартом де-факто во многих современных приложениях для обмена данными,
- достаточно выразителен: позволяет представить любую структуру данных,
- расширяем: позволяет добавлять данные, не нарушая существующую структуру.

Формат JSON поддерживает всего четыре элементарных типа данных:

1. число

- как целое
- так и вещественное

2. строка (в двойных кавычках)
3. Булево значение
   - *true*
   - *false*
4. отсутствие значения - *null*

и две структуры данных:

1. массив [ ] произвольных элементов
2. объект { } — неупорядоченный набор пар ключ–значение

К недостаткам формата JSON можно отнести прежде всего отсутствие возможности размещения комментариев, а также поддержки других типов данных, в особенности даты и времени. Однако, для этой цели существуют так называемые *расширения* JSON, которые могут быть при необходимости использованы, см. например [34].

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 4.1. Она содержит прямоугольный лист размером 500×700 и одну деталь в форме кольца диаметрами 200 (внешний) и 100 (внутренний). Контуры представлены в виде набора точек на евклидовой плоскости $\mathbb{R} \times \mathbb{R}$ двумя координатами (x и y), третье число представляет собой кривизну участка контура, для отрезка прямой она равна 0, для дуги в 180° — ± 1, а в общем случае — $tg\frac{\varphi}{4}$, где φ — центральный угол дуги.

```
1 [{
2   "partid": "LIST",
3   "paths": [
4   [
5   [0, 0, 0],
6   [0, 500, 0],
7   [700, 500, 0],
```

```
 8     [700, 0, 0],
 9     [0, 0, 0]]
10   ]},
11   {
12     "partid": "RING",
13     "paths": [
14     [
15     [205, 405, -1],
16     [205, 5, -1],
17     [205, 405, 0]],
18     [
19     [205, 305, 1],
20     [205, 105, 1],
21     [205, 305, 0]]
22   ]}]
```

Листинг 4.1. JSON-файл с геометрией простейшей раскройной карты

4.1.2. Разработка спецификаций JSON

Опыт практического использования формата JSON показал преимущества его использования. Например, разработчики программного обеспечение *RouterManager*, реализующего «жадный» алгоритм [59] и алгоритм на основе динамического программирования схемы Беллмана [8], полностью отказались от использования формата DBS [116] и перешли к использованию формата JSON. Более того, последний стал использоваться и для других файлов данных:

– *Файл задания на резку.* Содержит геометрическую информацию, подобно формату DBS, но в другом представлении — контуры деталей организованы не в плоский список, а дерево для явного представления ограничений предшествования. Здесь же описывается набор допустимых точек врезки M_i, полученных дискретизацией контуров, поскольку все алгоритмы решают задачу GTSP. Кроме того, сюда добавлены некоторые параметры процесса резки, например скорости резки V_{on} и холостого хода V_{off}, координаты начала / конца маршрута и т.п.

– *Файл результатов резки.* Содержит подробное описание маршрута режущего инструмента — активные сегменты, из которых он состоит со

ссылками на исходные контура деталей, сегменты холостого хода, интегральные характеристики, включая длины реза L_{on} и холостого хода L_{off}, время, количество точек врезки и т.п., необходимые для расчета целевых функций (1.2) и (1.3)

Структура этих файлов оказалась уже заметно сложнее структуры файла геометрической информации, особенно с учетом того, что последняя намеренно была разработана максимально упрощенной. Возникла потребность создания формальной спецификации форматов. Для этой цели используется механизм JSON-схем [66], сам основанный на формате JSON и обладающий довольно широкими выразительными возможностями описания семантики используемых структур данных.

В ходе диссертационной работы были разработаны JSON-схемы для всех используемых файлов данных, они приведены в Приложении Г и находятся в свободном доступе в [118].

Эти схемы могут использоваться во многих целях, таких как:

1. Документация на формат файла данных.
2. Автоматическая проверка корректности файлов данных (при помощи специализированных библиотек).
3. Автоматическая или полуавтоматическая генерация программного кода импорта / экспорта файлов данных описываемой структуры.

4.1.3. Разработка конвертеров форматов файлов данных

Для расширения поддержки разработанных форматов файлов данных в ходе диссертационной работы была разработан пакет утилит командной строки [115].

Утилиты написаны на языке JavaScript [32] и его диалекте CoffeeScript [9] и упакованы в исполняемый файл при помощи Webpack [87]. В настоящее время они работают только под операционной системой Microsoft Windows, хотя возможна сборка для большинства современных платформ.

Утилиты обеспечивают конвертацию между множеством форматов файлов, используемых разными подсистемами, включая:

– JSON с геометрической информацией о деталях и раскройной карте

– Двоичный файл геометрии DBS, экспорт и импорт

– Текстовый файл DXF [16], используемый для обмена графической информацией между CAD-системами. Обеспечивается экспорт и импорт.

– Экспорт геометрической информации в YAML [88]

– Запуск автоматического раскроя в САПР «Сириус» с выбором параметров раскроя.

– Запуск автоматической резки в САПР «Сириус» при помощи RoutingManager [8; 59].

– Экспорт задания и импорт результатов в систему фигурного раскроя T-Flex Раскрой [93]

– Экспорт и импорт в форматы файлов системы автоматического раскроя Nesting Factory [51]. Обеспечивается также автоматический запуск раскроя и последующий автоматический импорт его результатов.

– Экспорт в виде графа для поиска Эйлерова цикла [47—49] в рамках решения задачи маршрутизации режущего инструмента.

– Экспорт в виде HTML-страницы, содержащей SVG для визуализации, см. раздел 4.2.

Использование открытых форматов позволяет легко расширять этот список. Вместе с тем, более правильным представляется встраивание экспорта и импорта таких форматов непосредственно в код используемых подсистем, однако это не всегда возможно оперативно.

4.2. Визуализация геометрической информации

На разных этапах как научных исследований, так и технологической подготовки производства, возникает потребность визуализации разнообразной геометрической информации, такой как геометрия деталей и ограничивающих их контуров, положение допустимых точек врезки и выключения инструмента, маршруты, получаемые в ходе решения различных классов задач резки, а также маршруты движения резака, получаемые после обработки постпроцессором и т.п.

Традиционным подходом к визуализации является разработка специализированных графических утилит либо отдельных графических представлений в рамках больших программных систем. Зачастую этот подход является оправданным, но у него есть и свои недостатки — прежде всего усложнение процесса разработки и тестирования программного обеспечения. Кроме того, полученные таким образом визуализации могут использоваться, например, для научных публикаций, только посредством создания растровых копий экранов, или же требуется разработка отдельной функциональности для экспорта визуализации в файл некоторого формата.

Альтернативный подход, зачастую оказывающийся *гораздо* легче в разработке, заключается в том, чтобы визуализация производилась путём экспорта в некоторый графический формат, который уже может широко использоваться как для распространения, так и для собственно просмотра при помощи стандартных утилит. Стандартом де-факто в наше время для этой цели является формат SVG [73], а для его оперативного отображения может использоваться любой современный браузер, впрочем как и множество готовых библиотек для встраивания в разрабатываемое программное обеспечение. Важным достоинством такого подхода является его кросс-платформенность, то есть возможность использовать его на большинстве платформ и операционных систем.

В данной диссертационной работе как раз повсеместно и использовался формат SVG при намеренном отказе от разработки собственных программ визуализации. Например, в Листинге 4.2 приводится простейший SVG-файл, сгенерированный для раскройной карты, представленной на Листинге 4.1.

```
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 4.2. SVG-файл для визуализации раскроя из Листинга 4.1

Можно заметить, что команды SVG практически идеально соответствуют элементам геометрической информации. Фактически, SVG может использоваться даже как альтернативный вариант хранения и обмена графической информацией (вместо DXF, DBS, JSON...), хотя в данной работе этого не происходило, утилиты раздела 4.1.3 только экспортируют в SVG без возможности обратного преобразования.

4.2.1. Настройка параметров визуализации

Значительный объем работы при реализации специализированных утилит для визуализации составляет подбор параметров изображения, таких как цвета объектов, размеры линий и т.п. Как правило требуется предоставить пользователю возможность влиять на такие решения, что дополнительно усложняет и сам программный код и взаимодействие с ним пользователя.

Формат SVG в свою очередь глубоко интегрирован с механизмом каскадных таблиц стилей (Cascading Style Sheets, CSS), также являющимся современным активно развивающимся стандартом. Благодаря этому получается упростить вопросы настройки параметров визуализации. Фактически, речь идёт о редактировании одного или нескольких текстовых файлов, а возможности CSS очень велики [3] и в общем превосходят потребности CAD/CAM-систем:

- Настройка цветов изображений.
- Настройка толщин, цветов и вида линий.
- Настройка заливки фигур, включая сплошную, градиентную и штриховую, что необходимо для CAD-систем.
- Широкие возможности геометрических трансформаций всего изображения или групп объектов, включая сдвиг, поворот, отражение и масштабирование.
- Построение теней
- Возможность анимации, как постоянно действующей, так и интерактивной

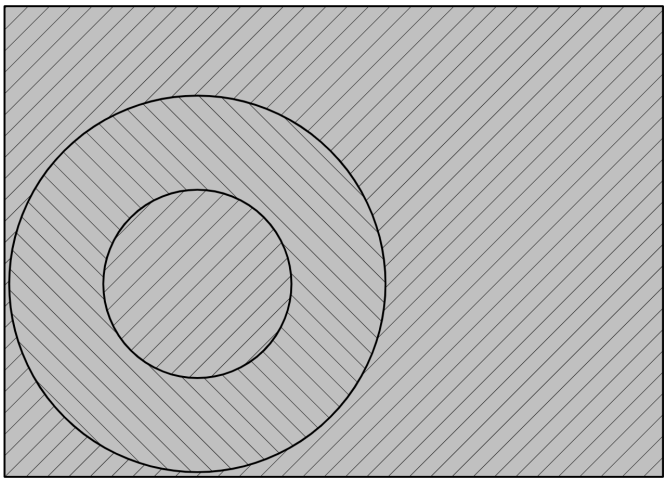


Рис. 4.1. Визуализация раскроя из Листинга 4.1

– И многое другое...

В ходе данной диссертационной работы были разработаны несколько схем оформления получаемых SVG-файлов, интегрированные в утилиты раздела 4.1.3. Один из вариантов оформления SVG-файла из Листинга 4.2 приведён на рис. 4.1.

4.2.2. Организация пользовательского интерфейса

Когда визуализация представлена на мониторе, у пользователя зачастую возникает потребность взаимодействия с изображением — как целиком, например, изменить масштаб, повернуть или подвинуть его на экране, так и с отдельными его элементами, например, навести на него курсор мыши и получить дополнительную информацию, удалить, сдвинуть или изменить его и т.п. Удобная организация графического пользовательского интерфейса (GUI) традиционно является одной из сложных задач программирования. Кроме того, именно в этой области чаще всего возникают сложности с переносимостью программного кода между разными платформами и операционными системами.

Использование SVG и здесь позволяет использовать огромный опыт, накопленный в области организации пользовательского интерфейса за десятилетия развития Web-технологий. Объекты SVG широко поддерживают управление при помощи языка JavaScript [32], также являющегося главным (а до недавнего времени — единственным) языком программирования, исполняемым в браузерах.

Программы на языке JavaScript способны реагировать на огромное количество событий, включая сюда

- нажатие пользователем клавиш на клавиатуре
- движение курсора мыши
- нажатие кнопок мыши
- события таймера
- загрузку страницы и завершение ее просмотра
- начало и конец анимации объектов

и производить с объектами страницы произвольные действия:

- прятать их и восстанавливать
- создавать новые объекты и удалять их
- двигать, увеличивать или уменьшать
- изменять цвета, шрифты, способы заливки
- изменять текст

и многое другое. Благодаря этому возможна организация сколь угодно сложных сценариев взаимодействия с пользователем.

При этом нет необходимости разрабатывать все это самостоятельно, целиком, «с нуля». В мире создано и регулярно создается огромное количество библиотек и фреймворков, реализующих разные аспекты пользовательского интерфейса на высоком уровне абстракции. Например, утилиты раздела 4.1.3 для организации масштабирования и прокрутки встраивают в экспортируемый SVG-файл обращение к внешней библиотеке с открытым кодом svg-pan-zoom [33].

Для визуализации решения задачи PCGTSP (см. Главу 2), которая требует объединения информации, полученной из нескольких источников (раскройная карта; координаты допустимых точек врезки; индексы вершин оптимального маршрута), была разработана специализированная утилита [114].

Для разработки использовался стек технологий, сходный с описанным в разделе 4.1.3: язык JavaScript [32] и его диалект LiveScript [46], система сборки rollup.js [69]. Утилита изначально являлась кросс-платформенной. Сначала она разрабатывалась как утилита командной строки, но опыт использования в данном исследовании показал, что это неудобно, и она была трансформирована в Web-приложение Single Page Application (SPA), которое может запускаться в браузере, как локально, так и будучи размещенным на произвольном хостинге.

Пример созданного ею изображения оптимального решения задачи PCGTSP для 34 кластеров приведен на рис. 2.1, стр. 38.

4.3. Выводы по Главе 4

1. Использование открытых форматов файлов позволяет резко упростить и ускорить организацию взаимодействия подсистем и в частности, встраивание алгоритмов оптимальной маршрутизации режущего инструмента для машин листовой резки с ЧПУ в существующие CAD/CAM-системы.

2. Предложенные схемы информационного обмена позволяют интегрировать в одну подсистему автоматического проектирования маршрутов резки существующие алгоритмы оптимизации, включая алгоритмы, использующие схему непрерывной оптимизации.

3. Использование открытых форматов позволяет привлекать наработки сообщества для реализации значительной части функциональности, что также полезно как для ускорения разработки, так и повышения качества разрабатываемых программных систем.

4. Использование механизмов визуализации на основе SVG вполне имеет право на существование в рамках как научных исследований, так и практического применения в ходе проектирования управляющих программ для машин листовой резки с ЧПУ в промышленном производстве.

5. Полный перевод всех подсистем на открытые форматы представляется нереалистичным, однако движение в этом направлении определенно следует продолжать при каждой возможности.

Заключение

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

1. Разработан алгоритм ветвей и границ для точного решения обобщённой задачи коммивояжёра с ограничениями предшествования. Он может быть реализован в классической схеме, а также в парадигме динамического программирования, при этом он допускает распараллеливание и демонстрирует лучшую производительность.

2. Предложенный алгоритм способен находить точные решения для задач большего размера, чем известные алгоритмы. В проведённых экспериментах было найдено решение для задачи со 151 кластером.

3. Данный алгоритм также решает важную задачу оценки качества решений, в том числе полученных другими алгоритмами.

4. Разработана схемы эффективного учёта ограничений предшествования для задач маршрутизации как в дискретной, так и в непрерывной схеме оптимизации.

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

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

7. Разработанные алгоритмы могут использоваться для решения задач сегментной непрерывной резки (SCCP и GSCCP) тем самым открывая подход к решению общей задачи прерывистой резки (ICP)

8. Разработаны форматы данных и алгоритмические схемы для обмена геометрической и маршрутной информацией и визуализации для использования в CAD/CAM-системах, а также алгоритмы преобразования формата файлов, что позволило интегрировать разработанное ПО с САПР «Сириус» и T-Flex CAD.

9. Предложенные схемы информационного обмена позволяют интегрировать в одну подсистему автоматического проектирования маршрутов резки существующие алгоритмы оптимизации, включая алгоритмы, использующие схему непрерывной оптимизации.

10. 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. Разработка методов получения нижних оценок для частичных подзадач GTSP; например, за счёт более точного учёта расстояний между узлами, а не только между кластерами.

2. Разработка метаэвристических алгоритмов дискретной оптимизации в задачах непрерывной резки (CCP, SCCP) и оценка производительности и качества получаемых алгоритмов.

3. Учет технологических требований термической резки для разработанных в данной диссертационной работе алгоритмов.

4. Интеграция разработанных алгоритмов с отечественными САПР для проектирования УП машин листовой резки

Список основных сокращений

ДП	Динамическое программирование (Dynamic Programming, DP)
Орграф	Ориентированный граф (Directed graph)
ПО	Программное обеспечение
САПР	Система автоматизированного проектирования
УП	Управляющая программа (для ЧПУ)
ЧПУ	Числовое программное управление
ALNS	Adaptive Large Neighborhood Search
AP	Assignment Problem (Задача о назначениях)
ATSP	Asymmetric Traveling Salesman Problem (Асимметричная задача коммивояжера)
BnB	Branch-and-Bound (Алгоритм ветвей и границ)
CAD	Computer Aided Design (Система автоматизированного проектирования, САПР)
CAM	Computer Aided Manufacturing (Система автоматизации технологической подготовки производства)
CCP	Continuous Cutting Problem (Задача непрерывной резки)
CNC	Computer Numerical Control (Числовое программное управление, ЧПУ)
DP	Dynamic Programming (Динамическое программирование, ДП)
GSCCP	Generalized Segment Continuous Cutting Problem (Обобщенная задача непрерывной сегментной резки)
GTSP	Generalized Traveling Salesman Problem (Обобщенная задача коммивояжера)

ICP Intermittent Cutting Problem (Задача прерывистой резки)

JSON JavaScript Object Notation (Открытый формат сериализации данных)

LB Lower bound (Нижняя оценка)

MILP Mixed-Integer Programming Problem (Смешанное целочисленное линейное программирование)

MIP Mixed Integer Programming (Смешанное целочисленное программирование)

MSAP Minimal Spanning Arbouring Problem (Задача поиска минимального остовного дерева)

PCGLNS Precedence Constrained GTSP Large Neighborhood Solver

PCGTSP Precedence Constrained Generalized Traveling Salesman Problem (Обобщенная задача коммивояжера с ограничениями предшествования)

PTAS Polynomial-Time Approximation Scheme (Приближённая схема полиномиального времени)

SCCP Segment Continuous Cutting Problem (Задача непрерывной сегментной резки)

SVG Scalable Vector Graphics (Язык разметки масштабируемой векторной графики)

TPP Touring Polygon Problem (Задача обхода многоугольников)

TSP Traveling Salesman Problem (Задача коммивояжера)

UB Upper bound (Верхняя оценка)

VNS Variable Neighborhood Search (Метод переменных окрестностей)

Список литературы

1. *Balas E.* Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs: A Computational Study / E. Balas, N. Simonetti // INFORMS J. on Computing. — Institute for Operations Research, the Management Sciences (INFORMS), Linthicum, Maryland, USA, 2001. — Т. 13, № 1. — С. 56—75. — URL: http://dx.doi.org/10.1287/ijoc.13.1.56.9748.
2. *Arkin E. M.* Approximation algorithms for the geometric covering salesman problem / E. M. Arkin, R. Hassin // Discrete Applied Mathematics. — 1994. — Т. 55, № 3. — С. 197—218.
3. Cascading Style Sheets. — URL: https://www.w3.org/Style/CSS/.
4. *Castelino K.* Toolpath optimization for minimizing airtime during machining / K. Castelino, R. D'Souza, P. K. Wright // Journal of Manufacturing Systems. — 2003. — Т. 22, № 3. — С. 173—180. — URL: http://www.sciencedirect.com/science/article/pii/S0278612503900185.
5. *Chentsov A. G.* A Discrete–Continuous Routing Problem with Precedence Constraints / A. G. Chentsov, A. A. Chentsov // Proceedings of the Steklov Institute of Mathematics. — 2018. — Т. 300, № 1. — С. 56—71.
6. *Chentsov A. G.* An exact algorithm with linear complexity for a problem of visiting megalopolises / A. G. Chentsov, M. Y. Khachai, D. M. Khachai // Proceedings of the Steklov Institute of Mathematics. — 2016. — Т. 295, № 1. — С. 38—46. — URL: https://doi.org/10.1134/S0081543816090054.
7. *Chentsov A.* Linear time algorithm for Precedence Constrained Asymmetric Generalized Traveling Salesman Problem / A. Chentsov, M. Khachay, D. Khachay // IFAC-PapersOnLine. — 2016. — Т. 49, № 12. — С. 651—655. — URL: http://www.sciencedirect.com/science/article/pii/S2405896316310485 ; 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.
8. *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.

9. CoffeeScript: a little language that compiles into JavaScript. — URL: https://coffeescript.org/.

10. *Dewil R.* A Critical Review of Multi-hole Drilling Path Optimization / R. Dewil, İ. Küçükoğlu, C. Luteyn, D. Cattrysse // Archives of Computational Methods in Engineering. — 2019. — Т. 26, № 2. — С. 449—459. — URL: https://doi.org/10.1007/s11831-018-9251-x.

11. *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.

12. *Dewil R.* Construction heuristics for generating tool paths for laser cutters / R. Dewil, P. Vansteenwegen, D. Cattrysse // International Journal of Production Research. — 2014. — Т. 52, № 20. — С. 5965—5984.

13. *Dewil R.* Sheet Metal Laser Cutting Tool Path Generation: Dealing with Overlooked Problem Aspects. Т. 639 / R. Dewil, P. Vansteenwegen, D. Cattrysse. — Trans Tech Publications Ltd, 2015.

14. *Dewil R.* An improvement heuristic framework for the laser cutting tool path problem / R. Dewil, P. Vansteenwegen, D. Cattrysse, M. Laguna, T. Vossen // International Journal of Production Research. — 2015. — Т. 53, № 6. — С. 1761—1776.

15. *Dror M.* Touring a sequence of polygons / M. Dror, A. Efrat, A. Lubiw, J. S. Mitchell // Proceedings of the thirty-fifth annual ACM symposium on Theory of computing. — ACM. 2003. — С. 473—482.

16. DXF Specifications. — 2012. — URL: http://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf.

17. Extensible Markup Language (XML). — URL: https://www.w3.org/XML/.

18. *Ezzat A.* A bare-bones ant colony optimization algorithm that performs competitively on the sequential ordering problem / A. Ezzat, A. M. Abdelbar, D. C. Wunsch // Memetic Computing. — 2014. — Т. 6, № 1. — С. 19—29.

19. *Feremans C.* The geometric generalized minimum spanning tree problem with grid clustering / C. Feremans, A. Grigoriev, R. Sitters // 4OR. — 2006. — Т. 4, № 4. — С. 319—329. — URL: https://doi.org/10.1007/s10288-006-0012-6.

20. *Fischetti M.* A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem / M. Fischetti, J. J. S. González, P. Toth // Operations Research. — 1997. — Т. 45, № 3. — С. 378—394.

21. *Frolovsky V. D.* Constructing the Shortest Closed Tour on a Set of Line Segments Using Ant Colony approach / V. D. Frolovsky, N. D. Ganelina. —.

22. *Gendreau M.* Handbook of metaheuristics. Т. 2 / M. Gendreau, J.-Y. Potvin [и др.]. — Springer, 2010.

23. *Ghilas V.* An adaptive large neighborhood search heuristic for the Pickup and Delivery Problem with Time Windows and Scheduled Lines / V. Ghilas, E. Demir, T. Van Woensel // Computers & Operations Research. — 2016. — Т. 72. — С. 12—30.

24. *Gutin G.* A Memetic Algorithm for the Generalized Traveling Salesman Problem / G. Gutin, D. Karapetyan // Natural Computing. — 2010. — Т. 9, № 1. — С. 47—60.

25. *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.

26. *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`.

27. *Helsgaun K.* Solving the equality Generalized Traveling Salesman Problem using the Lin–Kernighan–Helsgaun Algorithm / K. Helsgaun // Mathematical Programming Computation. — 2015. — Т. 7, № 3. — С. 269—287.

28. *Hoeft J.* Heuristics for the plate-cutting traveling salesman problem / J. Hoeft, U. S. Palekar // IIE Transactions. — 1997. — Т. 29, № 9. — С. 719—731.

29. IEEE Standard for Floating-Point Arithmetic. — 2019. — URL: `https://ieeexplore.ieee.org/document/8766229`.

30. *Imahori S.* Generation of cutter paths for hard material in wire EDM / S. Imahori, M. Kushiya, T. Nakashima, K. Sugihara // Journal of Materials Processing Technology. — 2008. — Т. 206, № 1. — С. 453—461.

31. Introducing JSON. — URL: `https://www.json.org/`.

32. JavaScript - MDN Web Docs - Mozilla. — URL: https://developer.mozilla.org/ru/docs/Web/JavaScript.

33. 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.

34. JSON for Humans, superset of JSON that aims to alleviate some of the limitations of JSON by expanding its syntax to include some productions from ECMAScript 5.1. — URL: https://json5.org/.

35. *Kandasamy V. A.* Effective location of micro joints and generation of tool path using heuristic and genetic approach for cutting sheet metal parts / V. A. Kandasamy, S. Udhayakumar // International Journal of Material Forming. — 2020. — Т. 13, № 2. — С. 317—329.

36. *Karapetyan D.* Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem / D. Karapetyan, G. Gutin // European Journal of Operational Research. — 2012. — Т. 219, № 2. — С. 234—251. — URL: https://www.sciencedirect.com/science/article/pii/S0377221712000288.

37. *Karapetyan D.* Lin–Kernighan heuristic adaptations for the generalized traveling salesman problem / D. Karapetyan, G. Gutin // European Journal of Operational Research. — 2011. — Т. 208, № 3. — С. 221—232.

38. *Khachai M. Y.* Approximation Schemes for the Generalized Traveling Salesman Problem / M. Y. Khachai, E. D. Neznakhina // Proceedings of the Steklov Institute of Mathematics. — 2017. — Т. 299, № 1. — С. 97—105. — URL: https://doi.org/10.1134/S0081543817090127.

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

40. *Khachay M.* Complexity and approximability of the Euclidean Generalized Traveling Salesman Problem in grid clusters / M. Khachay, K. Neznakhina // Annals of Mathematics and Artificial Intelligence. — 2020. — Т. 88, № 1. — С. 53—69.

41. *Khachay M.* Towards Tractability of the Euclidean Generalized Traveling Salesman Problem in Grid Clusters Defined by a Grid of Bounded Height / M. Khachay, K. Neznakhina // Optimization Problems and Their Applications. Т. 871 / под ред. A. Eremeev, M. Khachay, Y. Kochetov, P. Pardalos. — Cham : Springer International Publishing, 2018. — С. 68—77. — (Communications in Computer and Information Science). — URL: https://doi.org/10.1007/978-3-319-93800-4%7B%5C_%7D6.

42. *Khachay M.* Problem-Specific Branch-and-Bound Algorithms for the Precedence Constrained Generalized Traveling Salesman Problem / M. Khachay, S. Ukolov, A. Petunin // Optimization and Applications. Т. 13078 / под ред. N. Olenev [и др.]. — Cham, Switzerland : Springer Nature Switzerland AG, 2021. — С. 136—148. — (Lecture Notes in Computer Science).

43. *Kudriavtsev A.* PCGLNS: adaptive heuristic solver for the Precedence Constrained GTSP / A. Kudriavtsev, M. Khachay. — 2020. — URL: https://github.com/AndreiKud/PCGLNS/.

44. *Laporte G.* Computational Evaluation Of A Transformation Procedure For The Symmetric Generalized Traveling Salesman Problem / G. Laporte, F. Semet // INFOR: Information Systems and Operational Research. — 1999. — Т. 37, № 2. — С. 114—120.

45. *Li J.* Automatic generation of auxiliary cutting paths based on sheet material semantic information / J. Li, H. Zhu, T. Zhang, L. He, Y. Guan, H. Zhang // International Journal of Advanced Manufacturing Technology. — 2020. — Т. 106, № 9. — С. 3787—3797.

46. LiveScript – a language which compiles to JavaScript. — URL: https://livescript.net/.

47. *Makarovskikh T. A.* Mathematical models and routing algorithms for economical cutting tool paths / T. A. Makarovskikh, A. V. Panyukov, E. A. Savitskiy // International Journal of Production Research. — 2018. — Т. 56, № 3. — С. 1171—1188.

48. *Makarovskikh T.* Software Development for Cutting Tool Routing Problems / T. Makarovskikh, A. Panyukov, E. Savitsky // Procedia Manufacturing. — 2019. — Т. 29. — С. 567—574.

49. *Makarovskikh T. A.* Software for Constructing A-chains with Ordered Enclosing for a Plane Connected 4-regular Graph / T. A. Makarovskikh, A. V. Panyukov // IFAC-PapersOnLine. — 2019. — Т. 52, № 13. — С. 2650—2655.

50. *Morin T. L.* Branch-And-Bound Strategies for Dynamic Programming / T. L. Morin, R. E. Marsten // Operations Research. — 1976. — Т. 24, № 4. — С. 611—627. — URL: http://www.jstor.org/stable/169764.

51. Nesting Factory: Automatic Nesting – Making Feasible. — URL: http://algomate.com/.

52. NetworkX: Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. — URL: https://networkx.org/.

53. *Noon C. E.* An Efficient Transformation Of The Generalized Traveling Salesman Problem / C. E. Noon, J. C. Bean // INFOR: Information Systems and Operational Research. — 1993. — Т. 31, № 1. — С. 39—44.

54. NumPy: library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. — URL: https://numpy.org/.

55. Official home of the Python Programming Language. — URL: https://www.python.org/.

56. *Panteleev A. V.* Metaheuristic Algorithms for Searching Global Extremum / A. V. Panteleev // MAI, Moscow. — 2009.

57. *Papadimitriou C.* Euclidean TSP is NP-complete / C. Papadimitriou // Theoret. Comput. Sci. — 1977. — Т. 4, вып. 3. — С. 237—244.

58. *Petunin A. A.* Elements of dynamic programming in local improvement constructions for heuristic solutions of routing problems with constraints / A. A. Petunin, A. A. Chentsov, A. G. Chentsov, P. A. Chentsov // Automation and Remote Control. — 2017. — Т. 78, № 4. — С. 666—681.

59. *Petunin A. A.* About routing in the sheet cutting / A. A. Petunin, A. G. Chentsov, P. A. Chentsov // Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software. — 2017. — Т. 10, № 3. — С. 25—39.

60. *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. 020005-1—020005-7.

61. *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.

62. *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. 060002-1—060002-8.

63. *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. 012041-1—012041-5.

64. *Petunin A.* General Model of Tool Path Problem for the CNC Sheet Cutting Machines / A. Petunin // IFAC-PapersOnLine. — 2019. — Т. 52, № 13. — С. 2662—2667.

65. *Petunin A. A.* Optimization Models of Tool Path Problem for CNC Sheet Metal Cutting Machines / A. A. Petunin, C. Stylios // IFAC-PapersOnLine. — 2016. — Т. 49, № 12. — С. 23—28.

66. *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.

67. *Punnen (auth.) A. P.* The Traveling Salesman Problem and Its Variations / A. P. Punnen (auth.), G. Gutin, A. P. Punnen (eds.) — 1-е изд. — Springer US, 2002. — (Combinatorial Optimization 12).

68. *Reihaneh M.* An Efficient Hybrid Ant Colony System for the Generalized Traveling Salesman Problem / M. Reihaneh, D. Karapetyan // Algorithmic Operations Research. — 2012. — Т. 7, № 1. — С. 22—29. — URL: https://www.erudit.org/en/journals/aor/2012-v7-n1-aor7_1/aor7_1art03.

69. Rollup: a module bundler for JavaScript. — URL: https://rollupjs.org/guide/en/.

70. *Salman R.* An Industrially Validated CMM Inspection Process with Sequence Constraints / R. Salman, J. S. Carlson, F. Ekstedt, D. Spensieri, J. Torstensson, R. Söderberg // Procedia CIRP. — 2016. — Т. 44. — С. 138—143. — URL: http://www.sciencedirect.com/science/article/pii/S2212827116004182 ; 6th CIRP Conference on Assembly Technologies and Systems (CATS).

71. *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.

72. *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.

73. Scalable Vector Graphics. — URL: https://www.w3.org/Graphics/SVG/.

74. SciPy: Python-based ecosystem of open-source software for mathematics, science, and engineering. — URL: https://www.scipy.org/.

75. *Sherif S. U.* Sequential optimization approach for nesting and cutting sequence in laser cutting / S. U. Sherif, N. Jawahar, M. Balamurali // Journal of Manufacturing Systems. — 2014. — Т. 33, № 4. — С. 624—638.

76. *Smith S. L.* GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem / S. L. Smith, F. Imeson // Computers & Operations Research. — 2017. — Т. 87. — С. 1—19.

77. *Srivastava S.* Generalized Traveling Salesman Problem through n sets of nodes / S. Srivastava, S. Kumar, R. Garg, P. Sen // CORS journal. — 1969. — Т. 7, вып. 2, № 2. — С. 97—101.

78. *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.

79. *Tavaeva A.* A Cost Minimizing at Laser Cutting of Sheet Parts on CNC Machines / A. Tavaeva, A. Petunin, S. Ukolov, V. Krotov // Mathematical Optimization Theory and Operations Research. — Cham, Switzerland : Springer, 2019. — С. 422—437.

80. The Gurobi Optimizer, commercial optimization solver for linear programming (LP), quadratic programming (QP), quadratically constrained programming (QCP), mixed integer linear programming (MILP), mixed-integer quadratic programming (MIQP), and mixed-integer quadratically constrained programming (MIQCP). — URL: `https://www.gurobi.com/`.

81. The Gurobi Python Interface. — URL: `https://pypi.org/project/gurobipy/`.

82. TOML: Tom's Obvious, Minimal Language. — URL: `https://github.com/toml-lang/toml`.

83. Tree: Simple fast compact user-readable binary-safe extensible structural format. — URL: `https://github.com/nin-jin/tree.d/`.

84. *Vicencio K.* Multi-goal path planning based on the generalized Traveling Salesman Problem with neighborhoods / K. Vicencio, B. Davis, I. Gentilini // 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. — IEEE. — С. 14—18.

85. *Vijay Anand K.* Heuristic and genetic approach for nesting of two-dimensional rectangular shaped parts with common cutting edge concept for laser cutting and profile blanking processes / K. Vijay Anand, A. Ramesh Babu // Computers & Industrial Engineering. — 2015. — Т. 80. — С. 111—124.

86. *Wäscher G.* An improved typology of cutting and packing problems / G. Wäscher, H. Haußner, H. Schumann // European Journal of Operational Research. — 2007. — Т. 183, № 3. — С. 1109—1130.

87. Webpack: an open-source JavaScript module bundler. — URL: `https://webpack.js.org/`.

88. YAML: YAML Ain't Markup Language™. — URL: `https://yaml.org/`.

89. *Ye J.* An Optimized Algorithm of Numerical Cutting-Path Control in Garment Manufacturing. T. 796 / J. Ye, Z. G. Chen. — Trans Tech Publications Ltd, 2013.

90. *Yu W.* A route planning strategy for the automatic garment cutter based on genetic algorithm / W. Yu, L. Lu // 2014 IEEE Congress on Evolutionary Computation (CEC). — IEEE. — С. 6—11.

91. *Yuan Y.* A branch-and-cut algorithm for the generalized traveling salesman problem with time windows / Y. Yuan, D. Cattaruzza, M. Ogier, F. Semet // European Journal of Operational Research. — 2020. — Т. 286, № 3. — С. 849—866. — URL: https://www.sciencedirect.com/science/article/pii/S0377221720303581.

92. *Yun Y.* Hybrid genetic algorithm approach for precedence-constrained sequencing problem / Y. Yun, H. Chung, C. Moon // Computers & Industrial Engineering. — 2013. — Т. 65, № 1. — С. 137—147.

93. *Бабичев С.* Оптимизация раскроя средствами T-FLEX / С. Бабичев // САПР и графика. — 2018. — № 9. — С. 50—53.

94. *Бурылов А. В.* Автоматическое и интерактивное формирование маршрута режущего инструмента в программном комплексе раскроя ITAS NESTING / А. В. Бурылов, Р. Т. Мурзакаев, В. С. Приступов // Приволжский научный вестник. — 2016. — 1 (53).

95. *Верхотуров М. А.* О задаче построения пути режущего инструмента с учетом термических воздействий при раскрое плоского материала / М. А. Верхотуров, Г. Н. Верхотурова, М. И. Айбулатов, Д. Р. Зарипов // Перспективные информационные технологии: сборник трудов междунар. научно-технической конференции. — 2020. — С. 346—351.

96. *Верхотуров М. А.* Раскрой листовых материалов на фигурные заготовки: оптимизация пути режущего инструмента на основе приминения группировки контуров / М. А. Верхотуров, П. Ю. Тарасенко, А. Р. Тарасснко // Альманах современной науки и образования. — 2008. — № 1. — С. 36—39.

97. *Верхотуров М. А.* Математическое обеспечение задачи оптимизации пути режущего инструмента при плоском фигурном раскрое на основе цепной резки / М. А. Верхотуров, П. Ю. Тарасенко // Вестник Уфимского

государственного авиационного технического университета. — 2008. — Т. 10, № 2.

98. *Канторович Л. В.* Рациональный раскрой промышленных материалов / Л. В. Канторович, В. А. Залгаллер. — Новосибирск : Наука, 1971. — 290 с.

99. *Мурзакаев Р. Т.* Применение метаэвристических алгоритмов для минимизации длины холостого хода режущего инструмента / Р. Т. Мурзакаев, В. С. Шилов, А. В. Бурылов // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. — 2015. — № 14.

100. *Мурзакаев Р. Т.* Построение маршрута режущего инструмента на основе алгоритма «Всемирного потопа» / Р. Т. Мурзакаев, В. С. Приступов // In the World of Scientific Discoveries/V Mire Nauchnykh Otkrytiy. — 2015. — Т. 69.

101. *Мухачева Э. А.* Рациональный раскрой промышленных материалов. Применение АСУ / Э. А. Мухачева. — М. : Машиностроение, 1984. — 176 с.

102. *Мухачева Э. А.* Модели и методы расчета раскроя-упаковки геометрических объектов / Э. А. Мухачева, М. А. Верхотуров, В. В. Мартынов. — Уфа : УГАТУ, 1998. — 217 с.

103. *Петунин А. А.* Две задачи маршрутизации режущего инструмента для машин фигурной листовой резки с ЧПУ / А. А. Петунин // Intelligent Technologies for Information Processing and Management (ITIPM'2014). — 2014. — С. 215—220.

104. *Петунин А. А.* САПР «Сириус» – оптимизация раскроя и резки листовых материалов в единичном производстве / А. А. Петунин, В. И. Кротов, С. С. Уколов, В. В. Видяпин // САПР и графика. — 1999. — № 10. — С. 42.

105. *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.

106. *Петунин А. А.* Эффективная маршрутизация робота/беспилотного летательного аппарата в задачах с условиями предшествования / А. А. Петунин, М. Ю. Хачай, С. С. Уколов // XIV Всероссийская мультиконференция по проблемам управления (МКПУ-2021). Т. 1. — Издательство Южного федерального университета, 2021. — С. 202—205.

107. *Петунин А. А.* Алгоритмы оптимальной маршрутизации для систем автоматизированного проектирования управляющих программ машин листовой резки с ЧПУ / А. А. Петунин, П. А. Ченцов, Е. Г. Полищук, С. С. Уколов, В. В. Мартынов // Актуальные проблемы прикладной математики и механики. — Институт математики и механики УрО РАН им. Н.Н. Красовского, 2020. — С. 58—59.

108. *Петунин А. А.* Методологические и теоретические основы автоматизации проектирования раскроя листовых материалов на машинах с числовым программным управлением : дис. ... д-ра техн. наук : 05.13.12 / Петунин Александр Александрович. — Уфимский государственный авиационно-технический университет, 2010.

109. *Петунин А. А.* О классификации техник фигурной листовой резки для машин с ЧПУ и одной задаче маршрутизации инструмента / А. А. Петунин, В. И. Кротов // Материаловедение. Машиностроение. Энергетика. — 2015. — С. 466—475.

110. *Петунин А. А.* Оптимальная маршрутизация инструмента машин фигурной листовой резки с числовым программным управлением. Математические модели и алгоритмы / А. А. Петунин, А. Г. Ченцов, П. А. Ченцов. — Издательство Уральского университета, 2020. — 247 с.

111. *Пушкарева Г. В.* Применение гибридного генетического алгоритма для оптимизации маршрутов / Г. В. Пушкарева // Автометрия. — 2006. — Т. 42, № 2. — С. 68—79.

112. *Таваева А. Ф.* Разработка инвариантного модуля генерации управляющих программ для машин лазерной резки. Вопросы интеграции с CAD/CAM системами / А. Ф. Таваева, Е. Н. Шипачева, П. А. Ченцов, А. А. Петунин, С. С. Уколов, А. П. Халявка // Актуальные проблемы прикладной математики и механики. — Институт математики и механики УрО РАН им. Н.Н. Красовского, 2020. — С. 70—71.

113. *Таваева А. Ф.* Разработка методик расчета временных и стоимостных параметров процесса резки в системах автоматизированного проектирования управляющих программ для машин листовой лазерной резки с ЧПУ : дис. ... канд. техн. наук : 05.13.12 / Таваева Анастасия Фидагилевна. — Уральский федеральный университет имени первого Президента России Б.Н. Ельцина, 2020.

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

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

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

117. *Уколов С. С.* Алгоритм ветвей и границ для обобщённой задачи коммивояжера с ограничениями предшествования / С. С. Уколов, М. Ю. Хачай. — 2021. — URL: `https://github.com/ukoloff/PCGTSP-BnB`.

118. *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/`.

119. *Файзрахманов Р. А.* Формирование энергоэкономичного маршрута режущего инструмента станков гидроабразивной и лазерной резки с ЧПУ / Р. А. Файзрахманов, Р. Т. Мурзакаев, А. В. Бурылов, В. С. Шилов // Электротехника. — 2015. — № 11. — С. 32—36.

120. *Фроловский В. Д.* Автоматизация проектирования управляющих программ тепловой резки металла на оборудовании с ЧПУ / В. Д. Фроловский // Информационные технологии в проектировании и производстве. — 2005. — № 4. — С. 63—66.

121. Центр коллективного пользования ИММ УрО РАН «Суперкомпьютерный центр ИММ УрО РАН». — URL: `https://parallel.uran.ru/`.

Список иллюстраций

Список таблиц

Приложение А. Документы о внедрении результатов диссертационного исследования

УТВЕРЖДАЮ

Директор по образовательной деятельности

_____ С. Г. Князев

«__02__» _декабря_ 2021 г.

АКТ

о внедрении результатов диссертационной работы Уколов С.С.

«Разработка алгоритмов оптимальной маршрутизации инструмента для САПР управляющих программ машин листовой резки с ЧПУ»

в учебном процессе.

Материалы научных и теоретических исследований, изложенных в диссертационной работе Уколов Станислава Сергеевича, используются в учебном процессе ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» при выполнении практических работ на кафедре информационных технологий и автоматизация проектирования по дисциплинам «Автоматизация проектирования раскройно-заготовительного производства», «Автоматизация конструкторского и технологического проектирования» при подготовке бакалавров по направлениям 09.03.02 «Информационные системы и технологии» и 15.03.01 «Машиностроение».

Заведующий кафедрой информационных технологий и автоматизации проектирования, директор школы базового инженерного образования — Д. В. Куреннов

Приложение Б. Основные формулы геометрии дуг на комплексной плоскости

Поскольку современное оборудование листовой фигурной резки с ЧПУ использует для задания маршрута резки два геометрических примитива — отрезки прямых и дуги окружностей, возникает вопрос удобного представления этих сущностей для аналитического исследования и программной обработки. Вооружившись тем фактом, что дробно-линейное преобразование комплексной плоскости переводит прямые в окружности и наоборот, можно использовать единое представление обоих геометрических примитивов, к тому же значительно уменьшив обращение к трансцендентным функциям.

Рассмотрим круговую дугу $\smile AMZ$ на рис. Б.1, имея в виду, что она может выражаться в отрезок прямой. Зафиксируем начальную точку A и конечную точку Z, а кривизну дуги выразим в виде параметра *bulge* по формуле:

$$\beta = \operatorname{tg} \frac{\measuredangle ACZ}{4} \tag{Б.1}$$

Знак параметра β выберем положительным для дуг, идущих против часовой стрелки ($\beta < 0$ на рис. Б.1). Для отрезка прямой $\beta = 0$, для половины круга $\beta = \pm 1$, а для часто встречающегося на практике случая дуги в четверть круга $\beta = \pm(\sqrt{2} - 1) \approx \pm 0.41421\ldots$

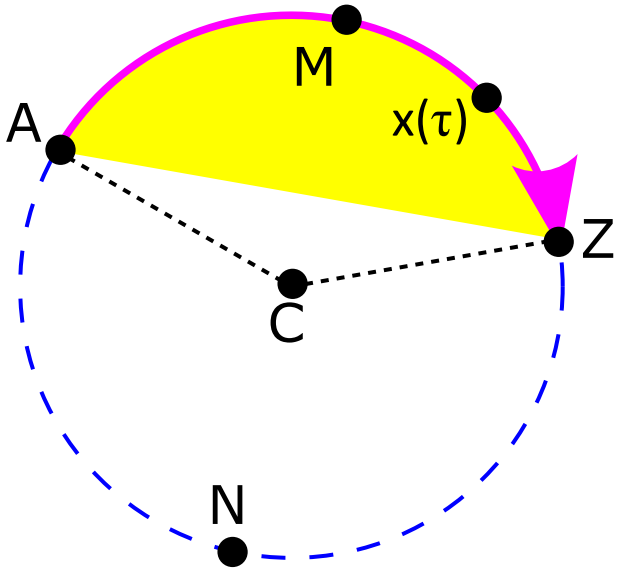


Рис. Б.1. Основные элементы круговой дуги

В этих обозначениях формула для положения произвольной точки $x(\tau)$ дуги, параметризуемой положением $\tau \in [-1, +1]$, может быть просто подобрана из общих соображений и приобретает вид:

$$x(\tau) = \frac{A+Z}{2} + \frac{Z-A}{2}\frac{\tau - i\cdot\beta}{1 - i\cdot\beta\tau} \tag{Б.2}$$

Очевидно, $x(-1) = A$, $x(+1) = Z$. Несложно также найти середину дуги M («зенит», $\tau = 0$)

$$M = \frac{A+Z}{2} - \frac{Z-A}{2}i\beta \equiv \frac{1+i\beta}{2}A + \frac{1-i\beta}{2}Z$$

и «надир» N ($\tau = \infty$):

$$N = \frac{A+Z}{2} - \frac{Z-A}{2}\frac{1}{i\beta} \equiv \frac{1+i\beta}{2i\beta}A - \frac{1-i\beta}{2i\beta}Z$$

Заметим, что «анти-дуга» $\smile ANZ$ опирается на те же точки A и Z, но с другим значением кривизны

$$\beta^- = -\frac{1}{\beta}$$

Далее уже легко найти центр:

$$C = \frac{(1+i\beta)^2}{4i\beta}A - \frac{(1-i\beta)^2}{4i\beta}Z$$

и радиус

$$R = \left|\beta + \frac{1}{\beta}\right| \frac{|A-Z|}{4}$$

Периметр и площадь

Длина (периметр) дуги $\smile AMZ$:

$$P_\smile = \left(1+\beta^2\right)\frac{\operatorname{arctg}\beta}{\beta}\,|A-Z| \tag{Б.3}$$

Для дуг, близких к отрезку ($\beta \to 0$):

$$P_\smile = \left(1 + \frac{2}{3}\beta^2 - \frac{2}{15}\beta^4 + \frac{2}{35}\beta^6 - \frac{2}{63}\beta^8 + O(\beta^{10})\right)|A-Z|$$

Площадь кругового сегмента AMZ:

$$S_\smile = \frac{\left(1-\beta^2\right)\beta - \left(1+\beta^2\right)^2\operatorname{arctg}\beta}{8\beta^2}|A-Z|^2 \tag{Б.4}$$

В пределе, когда дуга вырождается в отрезок,

$$S_{\smile} = \left(-\frac{1}{3}\beta - \frac{1}{15}\beta^3 + \frac{1}{105}\beta^5 - \frac{1}{315}\beta^7 + \frac{1}{693}\beta^9 + O(\beta^{11})\right)|A - Z|^2$$

Площадь треугольника $\triangle AOZ$ (где O — начало координат)

$$S_{\triangle} = \frac{\operatorname{im} A \cdot \overline{Z}}{2} \equiv \frac{\operatorname{re} Z \cdot \operatorname{im} A - \operatorname{im} Z \cdot \operatorname{re} A}{2} \tag{Б.5}$$

Знак в формулах для элемента площади (Б.4)–(Б.5) выбран в соответствии с соглашениями об ориентации контуров в DBS-файле, см. Приложение В.

Комбинируя формулы (Б.3)–(Б.5) для контура, состоящего из отрезков прямых и дуг окружностей, получаем соответственно периметр контура

$$P_{\circ} = \sum P_{\smile}$$

и площадь *замкнутого* контура

$$S_{\circ} = \sum S_{\smile} + S_{\triangle}$$

Разбиение дуги

Произвольная точка $x(\tau)$ делит дугу $\smile AMZ$ на две, вершины которых естественно вычислять по формуле (Б.2), а значения кривизны определяются по формуле

$$\begin{cases} \beta_-(\tau) = \operatorname{tg} \frac{\arg(1+i\beta)(1+i\beta\tau)}{2} \\ \beta_+(\tau) = \operatorname{tg} \frac{\arg(1+i\beta)(1-i\beta\tau)}{2} \end{cases},$$

причём функция $\operatorname{tg}(1/2 \arg z)$ может вычисляться без использования трансцендентных функций:

$$\operatorname{tg} \frac{\arg z}{2} = \begin{cases} \frac{|z| - \operatorname{re} z}{\operatorname{im} z}, & \operatorname{re} z < 0 \\ \frac{\operatorname{im} z}{|z| + \operatorname{re} z}, & \operatorname{re} z \geqslant 0 \end{cases} \tag{Б.6}$$

Кривизны поддуг $\beta_{\pm}(\tau)$ удовлетворяют формуле тангенса суммы:

$$\beta = \frac{\beta_-(\tau) + \beta_+(\tau)}{1 - \beta_-(\tau) \cdot \beta_+(\tau)}$$

В частном случае разбиения дуги пополам (в «зените») $\tau = 0$:

$$\beta_{1/2} = \frac{\beta}{1 + \sqrt{1 + \beta^2}}$$

Построение дуги по трем точкам

Иногда нужно найти дугу по ее концам и точке x, через которую она проходит. Кривизна такой дуги однозначно определяется углом $\measuredangle AxZ$:

$$\beta(x) = \operatorname{tg} \frac{\arg \overline{x - A} \cdot (Z - x)}{2},$$

и функция $\operatorname{tg}(1/2 \arg z)$ по-прежнему может вычисляться по формуле (Б.6).

Параметр же τ, задающий положение точки x на этой дуге, однозначно зависит от отношения $|A - x| : |Z - x|$:

$$\tau(x) = \frac{|A - x| - |Z - x|}{|A - x| + |Z - x|}$$

Скорость движения по дуге

Скорость $v(\tau)$ движения точки $x(\tau)$ по дуге при изменении τ легко находится как производная:

$$v(\tau) = \frac{\partial x}{\partial \tau} = \frac{Z - A}{2} \frac{1 + \beta^2}{(1 - i\beta\tau)^2}$$

Отсюда

$$|v(\tau)| = \frac{|Z - A|}{2} \frac{1 + \beta^2}{1 + \beta^2\tau^2},$$

и скорость движения точки $x(\tau)$ по дуге оказывается неравномерной, наибольшей в середине и минимальной по краям:

$$\frac{|v(0)|}{|v(\pm 1)|} = 1 + \beta^2,$$

и

$$\lim_{\beta \to \pm\infty} \frac{|v(0)|}{|v(\pm 1)|} = \infty,$$

что делает построение больших дуг непосредственно по формуле (Б.2) непрактичным. Разумеется, несложно найти другой параметр $\widetilde{\tau}$, такой, что $v(\widetilde{\tau}) = const$, но тогда в зависимость $\tau(\widetilde{\tau})$ будет входить тригонометрическая функция. Можно подобрать более простое преобразование, дающее *почти* равномерное движение:

$$\tau(\hat{\tau}) = \frac{\hat{\tau}}{1 + \frac{\sqrt{9 + 8\beta^2} - 3}{4} (1 - \hat{\tau}^2)} \qquad \text{(Б.7)}$$

и параметр $\hat{\tau} \in [-1, +1]$.

Преобразование (Б.7) подобрано таким образом, что $|v(0)| = |v(\pm 1)|$, можно показать, что для всех остальных $\hat{\tau} \in (-1, +1) \setminus \{0\}$

$$1 < \frac{|v(\hat{\tau})|}{|v(0)|} < \frac{1+\sqrt{2}}{2} \approx 1.2071\ldots$$

Приложение В. Описание формата файлов DBS

DBS — унаследованный двоичный формат файла обмена информацией в CAD-системах. В данной диссертационной работе используется в основном представление геометрической информации о плоских деталях.

DBS-файл состоит из произвольного количества DBS-записей. Каждая запись имеет тип, определяющий какие данные в ней содержатся. Основные типы записей сведены в табл. В.1.

Таблица В.1

Основные виды DBS-записей

Тип записи	Назначение
1	Геометрия одного контура детали
2	Копия контура (с геометрическим преобразованием)
4	Последовательность обработки
5	Текст
7	Плоскость
8	Объединение нескольких контуров в деталь
9	Фаска
10	Револьверная головка
11	Токарный инструмент
12	Инструментальный блок
13	Приспособление
21	Фрезерный инструмент
25	Маркировка
26	Наименование детали
27	Площадь и периметр детали
28	Примечание

Для каждой записи указывается длина, это позволяет пропускать неизвестные или неинтересные типы записей, читая только необходимые в данный момент. Благодаря этому формат DBS легко расширяем, возможно создание новых типов записей без необходимости изменять существующий код импорта / экспорта.

Служебная информация в полях записей систематически дублируется, что усложняет задачу корректной записи DBS-файла.

Заголовок записи

Все записи (кроме последней) начинаются со стандартного заголовка, содержащего длину, тип записи и служебную информацию.

Смещение	Тип	Имя	Описание
+0	int16	$size$	Размер записи (в 4-байтных словах)
+2	int16	id'	Не используется или копия поля id
+4	int16	$size'$	Копия поля $size$
+6	int16		
+8	int16	$type$	Тип записи (1, 2, 4, 5, ...)
+10	int16		
+12	int16	id	Номер записи
+14	int16		
+16	?		Данные записи (зависит от типа $type$)

Полный размер записи в байтах вычисляется по формуле $4 \cdot (size + 1)$, что для концевой записи дало бы 0, а для всех остальных записей дает осмысленное значение. Теоретически возможны записи с любым неотрицательным значением поля $size$, но на практике $size \geqslant 4$.

Семантика поля id запутана. В первом приближении это номер записи, что часто и бывает. Однако, номера не обязаны идти подряд и даже возрастать, допустимы любые неотрицательные значения. Зачастую, например, контура деталей нумеруются с 1, а сами детали со 100, тогда id записей могут идти вперемешку.

У разных объектов (например, контура и детали) id не могут совпадать, однако записи разного типа (например, 8, 26, 27, 28), описывающие одну деталь, обязаны иметь одинаковый id.

Допустимы ссылки вперед, когда например, сначала описывается деталь с $id = 1$, в которую включен контур с $id = 2$, который размещается в файле после содержащей его детали. Однако рекомендуется при записи файла таких ситуаций избегать и ссылаться в каждой записи только на уже описанные (находящиеся ближе к началу файла) записи.

Далее приведены описания только DBS-записей, используемых для хранения геометрической информации о плоских деталях, ограниченных контурами, состоящими из отрезков прямых и круговых дуг.

Копия контура (тип записи 2)

Данная запись часто называется «копией геометрии», потому что ее задача — взять контур (последовательность точек) из записи типа 1 и создать его копию путем поворота / отражения и смещения. Таким образом достигается значительное сокращение объема файла, поскольку типичная раскройная карта содержит множество копий одной и той же детали, отличающихся друг от друга только положением на листе.

Смещение	Тип	Имя	Описание
+0	dbs		Стандартный заголовок
+16	int16	*subtype*	Подтип записи (1, 2, 3)
+18	int16		
+20	int16	*text*	Признак наличия текста, связанного с контуром (0, 1)
+22	int16		
+24	int16	*autoseq*	Признак наличия автопоследовательности (0, 1)
+26	int16		
+28	int16	*part*	*id* детали
+30	int16		
+32	int16	*original*	*id* исходного контура
+34	int16		
+36	int16	*rev*	Признак реверса (-1, 0, 1, 2)
+38	int16		
+40	float32	$\cos_{xx}$	Матрица поворота / отражения
+44	float32	$\sin_{xx}$	
+48	float32	$\cos_{yx}$	
+52	float32	$\sin_{yx}$	
+56	float32	Δ_x	Смещение по горизонтали
+60	float32	Δ_y	Смещение по вертикали

Поле *part* — отсылка к группе записей (все с одинаковым *id*), описывающей деталь, в которую входит указанный контур. Для первого (внешнего) контура детали $part = -id$ детали, для всех остальных $part = id$. Тем самым дублируется информация из записи 8.

Поле *original* содержит *id* записи типа 1, которая копируется. В записи типа 1 поля *id* и *original* совпадают, в записи типа 2 всегда различаются.

Если в поле $rev \neq 0$, значит точки контура надо обходить в обратном порядке — от последней к первой. При этом следует изменить знак всех полей *bulge* в записи 1.

Геометрическое преобразование всех точек задается формулой:

$$\begin{cases} x' = \cos_{xx} \cdot x + \cos_{yx} \cdot y + \Delta_x \\ y' = \sin_{xx} \cdot x + \sin_{yx} \cdot y + \Delta_y \end{cases} \tag{B.1}$$

Матрица поворота зачастую бывает единичной, то есть копия получается из оригинала простым сдвигом. Смещение для записей типа 1 как правило является нулевым, а для записей 2 — никогда.

Геометрия контура (тип записи 1)

Данная запись часто называется «геометрия», так как содержит собственно список точек контура. Она содержит также всю информацию из записи 2, но здесь геометрическое преобразование как правило тривиально (ни сдвига, ни поворота, ни отражения).

Смещение	Тип	Имя	Описание
+0	rec2		Запись типа 2 (копия геометрии)
+64	point	$point_1$	Первая точка контура
+72	point	$point_2$	Вторая точка контура
+84	point	$point_3$	Третья точка контура
+96	point	$point_i$	...

Количество точек в записи явно не указано, но легко вычисляется исходя из размера записи (поле *size* стандартного заголовка).

Точка контура

Смещение	Тип	Имя	Описание
+0	float32	x	X-координата
+4	float32	y	Y-координата
+8	float32	$bulge$	Выпуклость участка контура
+12	point		Следующая точка контура

Поля x и y — двумерные координаты точки в виде чисел с плавающей точкой IEEE-754 (в миллиметрах).

Поле $bulge$ — кривизна участка контура от данной точки до следующей, определяемая как

$$bulge = \operatorname{tg} \frac{\varphi}{4},$$

где φ — угол дуги (или 0 для отрезка прямой). Подробнее про кривизну контура — см. Приложение Б. Параметр $bulge$ последней точки никогда не используется и всегда игнорируется, обычно там записывается 0.0. Знак $bulge$ определяется направлением обхода дуги, положительным считается обход против часовой стрелки, что немного противоречит знаку обхода контура.

По соглашению, материал детали при движении по контуру остаётся справа. Таким образом, внешние контура обходятся по часовой стрелке, внутренние — против.

В файле нет никакого признака того, что контур замкнут, хотя в некоторых случаях это важно. Определить замкнутость можно только сравнением координат первой и последней точки контура. Обычно это сравнение делается с некоторой точностью (например до $1 \cdot 10^{-3}$), но зачастую имеет смысл простое побитовое сравнение.

Связка контуров в деталь (тип записи 8)

Это основная запись о детали, Содержит список входящих в неё контуров. Эти же сведения продублированы в их поле $part$. Как правило эта запись сопровождается другими записями с тем же id (чаще всего 26 — наименование и 27 — площадь и периметр).

Смещение	Тип	Имя	Описание
+0	dbs		Стандартный заголовок
+16	int16	id_1	*id* первого контура
+18	int16		
+20	int16	id_2	*id* второго контура
+22	int16		
+24			...

Количество контуров также никак не указано, но вычисляется по размеру записи.

Обозначение детали (тип записи 26)

Здесь указывается обозначение детали (как правило совпадает с именем файла). Эта запись всегда идёт в паре с записью типа 8 (контура, относящиеся к детали) с тем же самым *id*.

Смещение	Тип	Имя	Описание
+0	dbs		Стандартный заголовок
+16	char[8]	*partid*	Обозначение детали

По историческим причинам обозначение детали хранится в странном формате:

- Строка дополняется до 8 символов пробелами справа
- В каждой паре символы переставляются

Таким образом деталь с *partid* «CIRCLE1» в поле *partid* будет иметь 8 символов «ICCREL 1».

Во время разработки формата DBS стандарта Unicode еще не существовало, поэтому в обозначении детали рекомендуется использовать только печатные ASCII-символы.

Измерения детали (тип записи 27)

Смещение	Тип	Имя	Описание
+0	dbs		Стандартный заголовок
+16	float32	*area*	Площадь детали, дм^2
+20	float32	*perimeter*	Периметр детали, дм

Несмотря на то, что размеры детали указываются в миллиметрах, площадь и периметр выражаются в дециметрах.

Для того, чтобы эта запись имела смысл, в DBS-файле должна присутствовать запись типа 8 (контура, относящиеся к детали) с тем же самым id.

Концевая запись

Последняя запись в файле имеет особый формат и показывает, что продолжение чтения невозможно.

Смещение	Тип	Имя	Описание
+0	int16	eof	-1
+2	int16	eof'	-1

Достаточно было бы первого бита, но принято помещать в конец DBS-файла 32 единичных бита.

Приложение Г. JSON-схемы

Ниже приводятся формальные спецификации [118] разработанных и использованных в диссертационной работе форматов файлов, см. раздел 4.1

Г.1. Сведения о геометрии деталей и раскроя

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://ukoloff.github.io/dbs.js/json-schema/dbs.json",
  "$ref": "#/$defs/dbs",
  "$defs": {
    "point": {
      "title": "Point on the plain",
      "description": "X, Y and bulge (i.e. tan(angle / 4) for an arc)",
      "type": "array",
      "items": {
        "type": "number"
      },
      "minItems": 2,
      "maxItems": 3,
      "default": 0.0
    },
    "path": {
      "title": "2D contour",
      "description": "Polyline consisting of line segments and arcs",
      "type": "array",
      "items": {
        "$ref": "#/$defs/point"
      },
      "minItems": 2
    },
    "part": {
      "title": "2D part",
      "description": "Collection of plain contours",
      "properties": {
        "partid": {
          "type": "string"
        },
        "paths": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/path"
          }
        },
        "area": {
          "type": "number"
```

```
41          },
42          "perimeter": {
43            "type": "number"
44          }
45        },
46        "required": [
47          "partid",
48          "paths"
49        ]
50      },
51      "dbs": {
52        "title": "DBS file",
53        "description": "Collection of 2D parts",
54        "type": "array",
55        "items": {
56          "$ref": "#/$defs/part"
57        },
58        "minItems": 1
59      }
60    }
61  }
```

Листинг Г.1. Файл геометрии деталей и раскроя

Г.2. Задание на резку

```
1  {
2    "$schema": "https://json-schema.org/draft/2020-12/schema",
3    "$id": "https://ukoloff.github.io/dbs.js/json-schema/rm-task.json",
4    "$ref": "#/$defs/task",
5    "$comment": "Generated by https://app.quicktype.io/",
6    "$defs": {
7      "point": {
8        "title": "Pierce poitnt",
9        "description": "Feasible pierse point to cut-in",
10       "type": "object",
11       "additionalProperties": false,
12       "properties": {
13         "Index": {
14           "type": "integer"
15         },
16         "ZeroBasedIndex": {
17           "type": "integer"
18         },
19         "GlobalIndex": {
20           "type": "integer"
21         },
22         "ZeroBasedGlobalIndex": {
23           "type": "integer"
24         },
25         "X": {
```

```
26            "type": "number"
27          },
28          "Y": {
29            "type": "number"
30          },
31          "UseCircuitAndFinishCutPoint": {
32            "type": "boolean"
33          }
34        },
35        "required": [
36          "GlobalIndex",
37          "Index",
38          "UseCircuitAndFinishCutPoint",
39          "X",
40          "Y",
41          "ZeroBasedGlobalIndex",
42          "ZeroBasedIndex"
43        ]
44      },
45      "contour": {
46        "title": "2D contour",
47        "description": "Sequence of 2D points with nested contours allowed",
48        "type": "object",
49        "additionalProperties": false,
50        "properties": {
51          "Index": {
52            "type": "integer"
53          },
54          "ZeroBasedIndex": {
55            "type": "integer"
56          },
57          "Points": {
58            "type": "array",
59            "items": {
60              "$ref": "#/$defs/point"
61            }
62          },
63          "NestedContours": {
64            "$ref": "#/$defs/contours"
65          }
66        },
67        "required": [
68          "Index",
69          "Points",
70          "ZeroBasedIndex"
71        ]
72      },
73      "contours": {
74        "title": "Several contours",
75        "type": "array",
76        "items": {
```

```
77          "$ref": "#/$defs/contour"
78        },
79        "minItems": 1
80      },
81      "params": {
82        "title": "Configuration for RouteManager",
83        "type": "object",
84        "additionalProperties": false,
85        "properties": {
86          "StartX": {
87            "type": "number"
88          },
89          "StartY": {
90            "type": "number"
91          },
92          "TerminalMotion": {
93            "type": "boolean"
94          },
95          "FinishX": {
96            "type": "number"
97          },
98          "FinishY": {
99            "type": "number"
100         },
101         "SheetMinX": {
102           "type": "number"
103         },
104         "SheetMaxX": {
105           "type": "number"
106         },
107         "SheetMinY": {
108           "type": "number"
109         },
110         "SheetMaxY": {
111           "type": "number"
112         },
113         "ToolIdlingSpeed": {
114           "type": "number"
115         },
116         "ToolCutSpeed": {
117           "type": "number"
118         },
119         "PiercingDuration": {
120           "type": "number"
121         }
122       },
123       "required": [
124         "FinishX",
125         "FinishY",
126         "PiercingDuration",
127         "SheetMaxX",
```

```
128           "SheetMaxY",
129           "SheetMinX",
130           "SheetMinY",
131           "StartX",
132           "StartY",
133           "TerminalMotion",
134           "ToolCutSpeed",
135           "ToolIdlingSpeed"
136         ]
137       },
138       "task": {
139         "title": "Task for RouteManager",
140         "description": "JSON version of RDF file, containing all data for running RouteMa
141         "type": "object",
142         "additionalProperties": false,
143         "properties": {
144           "TaskData": {
145             "$ref": "#/$defs/params"
146           },
147           "Contours": {
148             "$ref": "#/$defs/contours"
149           }
150         },
151         "required": [
152           "Contours",
153           "TaskData"
154         ]
155       }
156     }
157   }
```

Листинг Г.2. Задание на резку

Г.3. Результат резки

```
1  {
2    "$schema": "https://json-schema.org/draft/2020-12/schema",
3    "$id": "https://ukoloff.github.io/dbs.js/json-schema/rm-result.json",
4    "$ref": "#/$defs/result",
5    "$comment": "Generated by https://app.quicktype.io/",
6    "$defs": {
7      "mode": {
8        "title": "Mode of tool motion",
9        "type": "string",
10       "enum": [
11         "air",
12         "cut"
13       ]
14     },
15     "point": {
16       "title": "2D point",
```

```
17        "description": "X, Y and angle (for arc)",
18        "type": "array",
19        "items": {
20          "type": "number"
21        },
22        "minItems": 3,
23        "maxItems": 3
24      },
25      "points": {
26        "title": "2D contour",
27        "description": "Sequence of 2D points",
28        "type": "array",
29        "items": {
30          "$ref": "#/$defs/point"
31        },
32        "minItems": 2
33      },
34      "path": {
35        "title": "Part contour",
36        "description": "Single 2D contour of initial Task",
37        "type": "object",
38        "additionalProperties": false,
39        "properties": {
40          "part.id": {
41            "type": "integer"
42          },
43          "id": {
44            "type": "integer"
45          },
46          "points": {
47            "$ref": "#/$defs/points"
48          }
49        },
50        "required": [
51          "id",
52          "part.id",
53          "points"
54        ]
55      },
56      "job": {
57        "title": "Task solved",
58        "description": "Some information on the Task solved by RouteManager",
59        "type": "object",
60        "additionalProperties": false,
61        "properties": {
62          "paths.count": {
63            "type": "integer"
64          },
65          "parts.count": {
66            "type": "integer"
67          },
```

```
68          "total.points.count": {
69            "type": "integer"
70          },
71          "total.address.pairs.count": {
72            "type": "integer"
73          },
74          "list": {
75            "$ref": "#/$defs/points"
76          },
77          "paths": {
78            "type": "array",
79            "items": {
80              "$ref": "#/$defs/path"
81            }
82          }
83        },
84        "required": [
85          "list",
86          "parts.count",
87          "paths",
88          "paths.count",
89          "total.address.pairs.count",
90          "total.points.count"
91        ]
92      },
93      "toolpath": {
94        "title": "Tool path",
95        "description": "Sequence of points visited by tool during cutting process",
96        "type": "object",
97        "additionalProperties": false,
98        "properties": {
99          "point.id": {
100           "type": "integer"
101         },
102         "mode": {
103           "$ref": "#/$defs/mode"
104         },
105         "point": {
106           "$ref": "#/$defs/point"
107         },
108         "part.id": {
109           "type": "integer"
110         },
111         "id": {
112           "type": "integer"
113         }
114       },
115       "required": [
116         "mode",
117         "point",
118         "point.id"
```

```
119         ]
120       },
121       "results": {
122         "type": "object",
123         "additionalProperties": false,
124         "properties": {
125           "result": {
126             "type": "number"
127           },
128           "route.length.no.contours": {
129             "type": "number"
130           },
131           "route.length": {
132             "type": "number"
133           },
134           "count.time": {
135             "type": "string",
136             "format": "time"
137           },
138           "toolpath": {
139             "type": "array",
140             "items": {
141               "$ref": "#/$defs/toolpath"
142             }
143           }
144         },
145         "required": [
146           "count.time",
147           "result",
148           "route.length",
149           "route.length.no.contours",
150           "toolpath"
151         ],
152         "title": "Results"
153       },
154       "result": {
155         "title": "Solution obtained by RouteManager",
156         "type": "object",
157         "additionalProperties": false,
158         "properties": {
159           "job": {
160             "$ref": "#/$defs/job"
161           },
162           "results": {
163             "$ref": "#/$defs/results"
164           }
165         },
166         "required": [
167           "job",
168           "results"
169         ]
```

```
170         }
171     }
172 }
```

Листинг Г.3. Решение задачи резки