Приложение к стенограмме

Доклад Уколова Станислава Сергеевича

**Слайд 1**. Уважаемый председатель диссертационного совета! Уважаемые члены диссертационного совета, оппоненты и присутствующие! Представляю вашему вниманию результаты диссертационной работы «Разработка алгоритмов оптимальной маршрутизации инструмента для САПР управляющих программ машин листовой резки с ЧПУ».

**Слайд 2**. Актуальность работы определяется прежде всего высокой сложностью задачи оптимальной маршрутизации режущего инструмента, которая не только не решена в общем виде, но даже ещё нет её исчерпывающей формулировки. Исследуются только некоторые частные случаи, причём как правило предлагаются эвристики, дающие решения, но чрезвычайно сложно судить об их оптимальности. На практике зачастую используется интерактивный способ назначения маршрута, и вопрос оптимальности вообще не ставится. Поэтому интересно построение точных алгоритмов, причём получение оценок для возможных решений даже важнее собственно получения оптимальных решений. Интересно также получение решений близких к оптимальным за разумное время и с разумными вычислительными ресурсами. Для этого следует расширять классы задач маршрутизации, подлежащих решению, в частности шире использовать непрерывные модели оптимизации, в дополнение к традиционно применяемым чисто дискретным.

**Слайд 3**. Исходя из этих соображений была выработана цель диссертационной работы – разработать алгоритмы решения задачи оптимальной маршрутизации режущего инструмента и методики применения данных алгоритмов в САПР УП для машин фигурной листовой резки с ЧПУ. Для её решения предполагалось решить следующие задачи:

* Разработать точный алгоритм решения обобщённой задачи коммивояжера с ограничениями предшествования (PCGTSP), позволяющий оценить качество решений на основе вычисления нижней оценки
* Разработать эвристики поиска оптимального положения точек врезки в контуры деталей и последовательности обхода контуров в процессе решения задач непрерывной резки (CCP, SCCP)
* Разработать программное обеспечение, реализующие эти 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

**Слайд 4**. Научная новизна заключается в том, что

* Впервые разработан алгоритм ветвей и границ для обобщенной задачи коммивояжера с ограничениями предшествования PCGTSP. Он позволяет строить нижние оценки для решений указанной задачи и способен находить точные решения для задач значительно большей размерности, чем ранее известные алгоритмы, а также решения близкие к оптимальным для задач ещё большей размерности;
* Разработан алгоритм решения задачи непрерывной резки, не использующий механизм дискретизации контуров, путём сочетания двух эвристик для непрерывной и дискретной оптимизации. Кроме задачи непрерывной резки CCP (Continuous Cutting Problem), он оказался способен решать задачу сегментной резки SCCP (Segment Continuous Cutting Problem);
* В обоих алгоритмах учёт ограничений предшествования приводит к уменьшению вычислительной сложности задачи и сокращению времени счёта.
* Модели, использованные в данных алгоритмах, могут обобщаться для их применения для решения ещё более широких классов задач, в частности задачи обобщенной сегментной резки (Generalized SCCP) и даже интегрированной задачи раскроя и маршрутизации (Integrated Nesting and Routing Problem).

**Слайд 5**. Практическая значимость работы заключается в том, что:

1. Разработанные алгоритмы могут применяться для проектирования маршрута инструмента машин листовой резки с ЧПУ в автоматическом режиме, в том числе и при применении нестандартных техник резки;
2. Совместное использование дискретных и непрерывных моделей оптимизации позволяет уменьшить временные и стоимостные параметры маршрута инструмента (в некоторых случаях — до 10%) по сравнению с чисто дискретными моделями;
3. Разработанное программное обеспечение позволяет эффективно решать задачи большой размерности;
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 «T-Flex», а также обеспечивают эффективное тестирование новых оптимизационных алгоритмов.
5. Результаты исследований используются в образовательном процессе ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина».

**Слайд 6**. Основные результаты работы докладывались и обсуждались на международных и всероссийских научных конференциях. Список на экране.

**Слайд 7**. По теме работы осуществлено 18 публикаций, в том числе 9 в рецензируемых научных журналах, определенных ВАК РФ и Аттестационным советом УрФУ, из них 8 публикаций проиндексировано в международных базах данных Web of Science и Scopus. Список на экране.

**Слайд 8**. 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. Первая и последняя задача представляют собой чисто технические проблемы, а вот второй и третий этап – это сложные задачи, не имеющие полного решения на настоящий момент. Ввиду того, что они являются NP-трудными, их точное решение зачастую непрактично, поэтому для их решения широко применяются эвристические и метаэвристические подходы. В диссертационной работе исследовался только третий этап, то есть задача оптимальной маршрутизации режущего инструмента, или коротко говоря задача резки.

**Слайд 9**. При формализации искомый маршрут детали естественным образом представляется в виде кортежа, содержащего информацию двух классов: 1) сведения о выбранных точках врезки и соответствующих им точках выключения резака, соединённых сегментами резки. Это геометрическая информация, в конце концов представляемая в виде набора чисел с плавающей точкой. И 2) порядок вырезания сегментов, который представляет собой некоторую перестановку в терминах комбинаторики, автобиекцию конечного множества сегментов резки. Таким образом, по самому своему устройству задача маршрутизации режущего инструмента содержит как непрерывную, так и дискретную информацию и требует, вообще говоря, использования обеих моделей оптимизации. Кроме того, все элементы кортежа, представляющего маршрут резки, подчиняются некоторым ограничениями, которые тоже должны быть сформулированы и записаны в математической форме.

**Слайд 10**. В качестве целевой функции для оптимизации принято использовать две основных – стоимость резки и её время. Обе они могут быть вычислены, зная элементы кортежа маршрута резки и ряд параметров резки, таких как скорость рабочего и холостого хода, стоимость единицы их длины и операции врезки и время на единичную операцию врезки.

**Слайд 11**. В зависимости от выбора точек врезки и соотношения сегментов резки и контуров деталей, можно выделить несколько классов задач резки. Самый простой случай – задача коммивояжера, когда точка врезки вообще не выбирается. В большинстве исследований изучается обобщённая задача коммивояжера (она исследуется во второй главе диссертации), когда на каждом контуре детали выбирается несколько возможных точек врезки, и её более сложный вариант – задача резки с остановками, где контуры деталей также дискретизируются, но ещё могут вырезаться за несколько подходов, частями. Если же приём дискретизации не используется, то в простейшем случае возникает задача непрерывной резки (предмет третьей главы диссертации) и её усложнённые варианты – задача сегментной и обобщённой сегментной резки, которые позволяют исследовать подходы к задаче прерывистой резки, то есть самой общей формулировке задачи резки, которая всё ещё крайне слабо изучена.

**Слайд 12**. Перейдём к первой из исследуемых в работе задач – обобщённой задаче резки с ограничениями предшествования. Она решается в самой общей постановке, как взвешенный ориентированный граф, его вершины разбиты на кластеры, и на множестве кластеров задан частичный порядок. В задаче маршрутизации режущего инструмента это соответственно точки врезки, контуры и их вложенность. Мы ищем замкнутый маршрут минимального веса, посещающий каждый кластер ровно в одной вершине. В отличие от обобщённой задачи коммивояжера эта задача исследована гораздо хуже.

**Слайд 13**. Здесь приведён пример точного решения задачи PCGTSP, полученной из простой раскройной карты, сгенерированной в САПР «Сириус».

**Слайд 14**. Созданный в диссертационной работе алгоритм состоит из нескольких программных блоков. Основной интерес представляет собой блок построения нижних оценок, в котором в каждом узле дерева поиска исходная задача PCGTSP декомпозируется на две вспомогательных задачи, таким образом, что нижняя оценка оказывается суммой двух оценок, более сложная из вспомогательных задач два раза подвергается релаксации, сначала путём упрощения до асимметричной задачи коммивояжера, а потом её приближенного или при некоторых обстоятельствах даже точного решения. Полученная таким образом полная оценка используется в блоке отсечения ветвей дерева поиска, для этого до начала работы алгоритма однократно запускается недавно разработанная эвристика PCGLNS и сравнение нижней оценки с весом полученного ею решения позволяет отбрасывать ветви дерева поиска, причём эффективность этого процесса может достигать 90% на некоторых экземплярах задач. Наконец, если ветвь не отброшена, она подвергается процедуре ветвления, то есть находятся кластеры, которые достижимы из неё в соответствии с ограничениями предшествования, после чего вся процедура повторяется для новых ветвей.

**Слайд 15**. Здесь представлена процедура декомпозиции задачи PCGTSP для построения оценки снизу. Для префикса полного маршрута порядок посещения уже определён, остальные же кластеры пока могут посещаться в любом порядке с учётом ограничений предшествования. Минимальный путь вдоль кластеров префикса эффективно находится при помощи динамического программирования, а из остальных кластеров строится уменьшенная задача PCGTSP, индуцированная исходной. При этом за счёт использования ограничений предшествования удаётся значительно сократить количество рёбер в ней. Построенная таким образом вспомогательная задача подвергается двухступенчатой релаксации, причём на каждом шаге можно использовать несколько методов, получается нижняя оценка на её решение, которая даёт вклад в полную нижнюю оценку.

**Слайд 16**. В рамках диссертационной работы исследовалось несколько вариантов релаксации вспомогательной задачи, которые совместно дают 9 основных методов расчёта нижней оценки. В специально разработанном численном эксперименте эти методы сравнивались по быстродействию и соотношению даваемых ими оценок. Для окончательной версии алгоритма были отобраны три метода, обозначенные L1, L2 и L3.

**Слайд 17**. Описанные программные блоки собираются в алгоритм ветвей и границ классического дизайна, в котором поиск начинается с тривиального префикса длины 1. Длина рассматриваемых префиксов в ходе работы алгоритма растёт, то есть дерево поиска обходится в ширину, для каждого префикса строится нижняя оценка и принимается решение об отсечении. Для оставшихся префиксов осуществляется процедура ветвления. При достижении алгоритмом префиксов максимальной длины, равной количеству кластеров, эти префиксы суть решения задачи PCGTSP, из которых выбирается оптимальный.

**Слайд 18**. Полученный алгоритм оказался работоспособен, однако его дизайн содержит некоторые недостатки, связанные с тем, что многие вспомогательные задачи оказываются идентичными и требуется строить и решать их один раз. Поэтому хочется переупорядочить ветви дерева поиска, при этом естественным образом возникают сущности, используемые в классической схеме динамического программирования Хелда-Карпа. По этой причине была разработана вторая версия того же алгоритма. В отличие от первой, она допускает параллельное исполнение.

**Слайд 19**. Здесь также происходит наращивание длин префиксов, но все префиксы, дающие одну вспомогательную задачу, обрабатываются одновременно, в отдельном процессе. Таблица динамического программирования состоит из слоёв в соответствии с длинами префиксов. Каждый следующий слой строится на основе предыдущего.

**Слайд 20**. Здесь приведён псевдокод индуктивного построения одного слоя таблицы динамического программирования. Процедуры расчёта нижней границы, отсечения и ветвления работают аналогично первой версии этого алгоритма. Слои таблицы строятся один за другим, пока не будут построены все, после чего обратным просмотром получается решение задачи PCGTSP. Если нам нужен только вес оптимального решения, обратный проход можно опустить, а также резко сократить требования к памяти, храня только два слоя таблицы динамического программирования – текущий и предыдущий.

**Слайд 21**. На этом слайде приведены результаты численных экспериментов, в которых оценивалась производительность обеих версий алгоритма на примерах из открытой библиотеки PCGTSPLIB. В качестве базы сравнения использовался коммерческий решатель Gurobi. В целом, хотя Gurobi демонстрирует в среднем чуть лучшую производительность, предложенные алгоритмы за редким исключением, показывают вполне сопоставимые результаты.

**Слайд 22**. Перейдём к рассмотрению задачи непрерывной резки, в которой не используется приём дискретизации контуров деталей. Контура состоят из отрезков прямых и дуг окружностей. Ограничение предшествования определяется вложенностью контуров. Мы ищем кратчайший путь, посещающий каждый контур. Для этого нужно определить точку врезки на каждом контуре и последовательность их посещения.

**Слайд 23**. Предложенная эвристика решения задачи непрерывной резки состоит из четырёх этапов: сначала мы удаляем часть контуров, чтобы полученное решение автоматически удовлетворяло ограничениям предшествования. Отдельная эвристика способна находить точки врезки в контуры, минимизирую длину холостого хода. На этапе дискретной оптимизации мы ищем последовательность обхода контуров, преследуя ту же цель. Наконец, когда маршрут получен, мы возвращаем удалённые на первом шаге контура и пополняем маршрут точками врезки для них.\

**Слайд 24**. На первом этапе мы удаляем все контуры, внутри которых вложены другие. Задача упрощается, её размерность понижается.

**Слайд 25**. Эвристика поиска позиций точек врезки сводится к многократной релаксации. Начиная со случайных начальных позиций точек на контурах, фиксируем все из них кроме одной и находим для оставшейся положение, минимизирующее общую длину маршрута. Возможны два основных варианта взаимного расположения – когда контур, на котором расположена оптимизируемая точка находится между соседними или по одну сторону от них. В первом случае искомая позиция находится на пересечении контура и отрезка, соединяющего соседние точки врезки. Во втором, позиция находится по правилу «угол падения равен углу отражения». Таким образом находятся позиции всех точек по очереди, после чего процесс повторяется, пока все позиции точек не сойдутся с некоторой заранее заданной точностью. На практике процесс сходится очень быстро, буквально за несколько итераций.

**Слайд 26**. Для поиска порядка обхода контуров можно применить любую эвристику комбинаторной оптимизации, в диссертационной работе использовался метод переменных окрестностей. Снова, мы начинаем со случайного порядка обхода. Мы строим порядки обхода, близкие к исходному, например, при помощи всех попарных перестановок контуров в нём. Для каждого порядка обхода мы находим оптимальные позиции точек врезки, как описано на предыдущем слайде и длину маршрута, выбирая кратчайший. Если более короткого маршрута найти не удалось, значит мы попали в локальный минимум. Попытаемся выпрыгнуть из него, увеличивая размер окрестности, например, при помощи тройных перестановок контуров и т. п.

**Слайд 27**. Когда решение получено, остаётся вернуть удалённые на первом шаге контуры. Для каждого строятся все точки пересечения с полученным маршрутом, из них выбирается последняя и добавляется в маршрут. При этом он сохраняет свою длину и свойство оптимальности.

**Слайд 28**. Численные эксперименты проводились на раскройных картах, содержащих реальные детали. В качестве базы сравнения использовался алгоритм А. Г. Ченцова, который способен находить оптимальные решения для дискретной версии использовавшихся задач маршрутизации инструмента. Видно, что оба алгоритма строят очень похожие маршруты, то есть действительно получаются оптимальные решения задачи непрерывной резки. Можно заметить, что непрерывный алгоритм даёт чуть более короткое решение, причём разница может для отдельных экземпляров задач достигать 10%.