МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра Автоматизированных Систем Управления   
 (полное название кафедры)

Ермошенко Павел Андреевич

(И., О., фамилия студента – автора работы)

Разработка программного обеспечения для оптимизации маршрута обхода

(полное название темы магистерской диссертации)

геометрических объектов на плоскости на основе метода колонии муравьев

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**

постановка задачи

##### Руководитель

Фроловский

Владимир Дмитриевич  
(фамилия, И., О.)

д.т.н., профессор

(уч. степень, уч. звание)

Новосибирск, 2015 г.

# 1. Введение

В современном мире автоматизация получает все более и боллее широкое распространение. Автоматизация является привлекательным вариантом инвестиций (как денег, так и времени) ввиду хороших перспектив уменьшения затрат и упрощения производства (путем передачи работы, выполняемой человеком, машине).

Одним из самых очевидных направлений автоматизации является оптимизация производственных процессов. Сокращение затраченных ресурсов (временных или денежных) имеет прямой эффект на любое производство и всегда позволяет извлечь большую прибыль.

Оптимизация пройденного пути, будь то для транспорта или части механизма является одним из вариантов снизить затраты. В данной работе будет рассматриваться задача оптимизации пути обхода геометрических фигур на плоскости. Результат, полученный в процессе решения задачи может использоваться, например, для оптимизации маршруты некоего режущего устройства при создании определенной детали. Предприятия, занимающиеся производством партий деталей по заказу могут оптимизировать путь режущего устройства всего раз для одной схемы, чтобы получить определенный выйгрыш в сохраненных ресурсах (энергии, времени и запасе прочности деталей машин) при созданни каждой детали.

В данной работе, для решения задачи обхода будет использоваться алгоритм итеративной оптимизации “муравьиной колонии”, который основывается на поведении муравьев в реальной жизни. Результаты, полученные в процессе применения могут не являться лучшими доступными. Но учитывая, что полный перебор всех решений почти невозможен для данной задачи, “достаточно хорошие” результаты очень часто предпочтительны.

# 2. Описание предметной области

## 2.1. Задача коммивояжёра

Данная задача формулируется следующим образом: дан граф, в состав которого входят некое количество точек и ребра между ними. Нужно найти такой путь через этот граф, который бы затрагивал все точки хотя бы однажды и был самым коротким.

Это является самой простой формулировкой задачи. Название задачи исходит из того, что этот граф можно считать некоторым набором городов, соединенных дорогами (ребрами), а проходящего по этому графу – торговцем (коммивояжером). Находя самый короткий путь торговец уменьшает свои затраты на “доставку” своего товара.

У этой задачи есть множество вариаций. Представим некоторые из них:

* Замкнутая задача. Нужно найти такой путь, который не только проходит по всем городам, но и возвращается в исходный.
* Добавление эвристического “веса” ребрам графа. Тогда задача сводится к поиску маршрута, который проходит через ребра суммарный вес которых является самым низким.
* Добавление направлений к ребрам (ассиметричная задача). В такой вариации вес прохождения ребра в одну сторону может отличаться от прохождения в обратную.
* Метрическая вариация. В этой задаче для всех весов (длин) ребер выполняется неравенство треугольника Cij <= Cik + Cjk (C – длина, j,i,k – точки графа). Иными словами “обходной” путь к точке всегда является длиннее прямого пути.
* Полностью соединенный граф. В этой вариации нет четко заданных ребер графа (дорог), вместо этого проход может быть осуществлен из каждой точки в любую другую. Вес ребер в таком случае может быть вычислен из расстояния между точками.

Решением задачи, очевидно, является упорядоченная последовательность точек графа. При этом, стоит отметить что в этой последовательности некоторые города могут оказаться дважды (задача не имеет верхней границы посещения городов).

Данная задача является NP-трудной (Ричард Мэннинг Карп, 1972), поэтому полный перебор не является оптимальной стратегией в ее случае. На данный момент (2015 год), самая большая задача, решенная точным методом насчитывает 85900 точек (2006 год), но эффективного алгоритма для общего случая до сих пор не существует.

Существуют несколько отдельных алгоритмов решения задачи коммивояжера. Среди них метод ветвей и границ (точный) и алгоритм Христофайдса (названный в честь Никоса Христофайдса, приближенный), но их эффективность зависит от особенностей графа и варьируется от задачи к задаче.

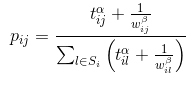
Одним из самых перспективных вариантов решения является представление задачи в виде задачи дискретной оптимизации и применения к ней соответствующих алгоритмов. В рамках этой работы мы рассмотрим алгоритм оптимизации “колонии муравьев”.

## 2.2. Алгоритм муравьиной колонии

Этот алгоритм был предложен в 1992 году Марко Дориго и он относится к алгоритмам дискретной оптимизации.

В рамках алгоритма, в дополнение к обычным объектам задачи коммивояжера, рассматривается система из некоторого количества муравьев (агентов системы). Каждый такой агент снабжен памятью пройденных им точек (в зависимости от данных в этой памяти, он может изменить решение о следующей точке). В каждой итерации алгоритма каждый из агентов делает выбор следующей точки своего маршрута. Этот выбор основывается на следующих факторах: случайная составляющая (добавление энтропии позволяет избежать схождения к первому найденному маршруту и дает возможность поиска альтернатив), количество феромона (оставленное на агентами на предыдущих итерациях) и эвристический показатель (в самом простом случае – длина ребра). В алгоритме также присутствует испарение феромона со временем (по той же причине что и случайная составляющая).

Пусть муравей находится в узле http://latex.codecogs.com/gif.latex?i, а узел http://latex.codecogs.com/gif.latex?j– это один из узлов, доступных для перехода: http://latex.codecogs.com/gif.latex?j%20%5Cin%20S_i. Обозначим вес ребра, соединяющего узлы http://latex.codecogs.com/gif.latex?iи http://latex.codecogs.com/gif.latex?j, как http://latex.codecogs.com/gif.latex?w_%7Bij%7D, а интенсивность феромона на нем – как http://latex.codecogs.com/gif.latex?t_%7Bij%7D. Тогда вероятность перехода муравья из http://latex.codecogs.com/gif.latex?i в http://latex.codecogs.com/gif.latex?j будет равна:



где http://latex.codecogs.com/gif.latex?%5Calpha и http://latex.codecogs.com/gif.latex?%5Cbeta– это регулируемые параметры, определяющие важность составляющих (веса ребра и уровня феромонов) при выборе пути. Выбор правильного соотношения параметров является предметом исследований, и в общем случае производится на основании опыта. После того, как муравей успешно проходит маршрут, он оставляет на всех пройденных ребрах след, обратно пропорциональный длине пройденного пути:

http://latex.codecogs.com/gif.latex?%5CDelta%20_%7Bij%7D%20=%5Cleft%5C%7B%5Cbegin%7Barray%7D%7Bc%7D%20%7B%5Cfrac%7Bk%7D%7BL%7D%20,%28ij%29%5Cin%20P%7D%20%5C%5C%20%7B0,%28ij%29%5Cnotin%20P%7D%20%5Cend%7Barray%7D%5Cright.

где http://latex.codecogs.com/gif.latex?L– длина пути, а http://latex.codecogs.com/gif.latex?k– регулируемый параметр. Кроме этого, следы феромона испаряются, то есть интенсивность феромона на всех ребрах уменьшается на каждой итерации алгоритма. Таким образом, в конце каждой итерации необходимо обновить значения интенсивностей:

http://latex.codecogs.com/gif.latex?t_%7Bij%7D%20=%5Cleft%281-e%5Cright%29%5Ccdot%20t_%7Bij%7D%20+%5CDelta%20t_%7Bij%7D

Название алгоритма обусловлено тем, что он моделирует поведение муравьев в реальной жизни. Используя этот способ (заложенный в них эволюцией), муравьи всегда могут отыскать короткий путь от гнезда к источнику пропитания. Это и является наглядным примером группового интеллекта (интеллекта стаи).

## 2.3. Задача обхода фигур на плоскости

Рассматриваемая в этой работе задача в упрощенном варианте (включающем в себя только фигуры, состоящие из прямых), сводится к задаче коммивояжёра с дополнительными условиями. Эти дополнительные условия включают в себя добавление обязательных к прохождению отрезков (отрезок считается пройденным если были пройдены его начало и конец один за другим в любом порядке), описание маршрута перехода между фигурами, а также, возможно, добавление точки возврата режущего инструмента (его исходного положения). Так как маршрут эффективность режущего инструмента не зависит от направления его движения, рассматриваемая задача является симметричной (с точки зрения задачи коммивояжера). Также, задача является метрической, потому что эвристическим параметром является расстояние между точками.

# 3. Предлагаемое решение

Самым большим отличием рассматриваемой задачи от задачи коммивояжера является наличие обязательных к прохождению ребер. Эти ребра являются контурами представленных на входном чертеже фигур и их наличие обеспечивает сам процесс получения нужной формы режущим инструментом. В данном случае предлагается изменение алгоритма выбора муравьями следующей точки прохода маршрута. Маршрут может считаться завершенным только если все обязательные отрезки были пройдены. Отрезок считается пройденным, если его точки записаны в маршрут одна за другой в любом порядке.

Второй отличительной особенностью является необходимость перемещаться между несколькими фигурами во время резки, так как чертеж может состоять из нескольких форм. В данном случае предлагается расставить на ребрах фигур дополнительные точки, которые будут использованы в процессе решения. Предлагается использовать случайный выбор места расстановки точек, а их количество будет зависеть от желаемой производительности алгоритма.

Последней проблемой является выбор самих параметров алгоритма оптимизации (количество муравьев, итераций и дополнительных точек). Этот выбор, как и в большинстве других случаев должен производиться на основе опыта. Предпочтительно будет реализовать как автоматический, так и ручной выбор данных параметров.

# 4. Программная реализация

Результатом данной работы является программный продукт, позволяющий создавать оптимальные маршруты обхода геометрических фигур на плоскости. Такие маршруты в последствие могут быть использованы в качестве инструкций для автоматического механизма резки.

В данном случае логично будет использовать архитектуру из двух частей. Вычислительная часть должна отвечать за реализацию самого алгоритма и получение результата. На вход она должна принимать полное описание задачи а на выходе получать полное описание результата. Интерфейс же должен позволять пользователю задать параметры задачи в интуитивном виде. После создания задачи интерфейс должен передать вычислительной части все входные данные для продолжения работы.

Архитектура из двух частей позволит полученному программному комплексу эффективно взаимодействовать с другими автоматизированными продуктами. Таким образом, возможно создать специальный переходник для этого комплекса, позволяющий использовать выходные данные на входе, например, программного обеспечения режущего инструмента. То же самое справедливо и для “переходника” на входе: возможно создать программное обеспечение, получающее входные данные задачи из, например, CAD файлов. В рамках данной работы переходники рассматриваться не будут.

## 4.1. Вычислительная часть

Для вычислительной части программы основной характеристикой является скорость выполнения алгоритма (так как используется алгоритм итеративной оптимизации и количество итераций часто влияет на качество полученного результата). Язык C++ был изобретен в 1979 году на основе языка C. Развитие этого языка продолжается и по сей день, последним стандартом является C++14. Благодаря высокому развитию инструментов (компиляторов и линковщиков) и возможности создания “родного” машинного кода, этот язык является отличным кандидатом для задач, требующих большой производительности. C++ считается одним из самых подходящих языков для “научного” программирования (выполнения математических задач).

Выполнение поставленной задачи требует наличия некоторых примитив и технологий:

* Точки на плоскости – наборы из двух координат.
* Функции расчета расстояния между точками.
* Генератора случайных чисел – реализация случайности в муравьином алгоритме.
* Ввод/вывод - возможность записи файлов на диск.
* Многопоточность – полное использование ресурсов вычислительной системы, параллелизация алгоритма муравьиной колонии.
* Механизмы анализа производительности. Позволят реализовать оптимальный выбор численных параметров алгоритма (количество муравьев, количество итераций и дополнительных точек).

Все эти технологии либо доступны в стандартной библиотеке C++, либо легко реализуемы с ее помощью. В случае возникновения проблем с реализацией каких-либо частей, предлагается использовать известную библиотеку Qt, включающую в себя множество дополнительных функций. На выходе получится бинарный исполняемый файл, который будет запускаться на платформе, под которую произведена компиляция (использование C++ позволит создать исполняемые файлы как под Windows так и под Linux).

## 4.2. Интерфейс пользователя

При архитектуре, состоящей из двух частей, интерфейс не играет большой роли на конечный результат, а вместо этого служит одним из вариантов задания параметров задачи. Для интерфейса важными характеристиками являются возможность запуска под большим количеством платформ (в случае предложенной архитектуры возможен сценарий создания задачи на одном терминале и расчет на другом) и простота понимания.

Стандарт HTML (HyperText Markup Language – язык разметки гипертекста) является основой Интернета. Со временем стандарт развивался и в данный момент Web-технологии, которые начинались как способ передачи статичных страниц с графикой и текстом, стали способом передачи целых сетевых приложений по сети с целью их запуска на компьютере пользователя. Веб-платформа является самой распространенной платформой запуска в мире.

При построении интерфейса нужно реализовать следующие примитивы:

* Область для создания геометрических фигур на плоскости – главная деталь интерфейса, позволяющая задать форму вырезки. Для этой цели возможно использовать элемент canvas (“полотно”).
* Система сохранения и загрузки файлов. Данная система должна позволить сохранить файл, который полностью описывает задачу, выбранную через интерфейс. Очевидно, что файл должен быть совместим с вычислительной частью.
* Дополнительные механизмы выбора параметров задачи. Они включают в себя такие параметры, как: количество муравьев, количество итераций (“ручной” выбор параметров), нужно ли вернуться в исходную точку после завершения и другие.
* Механизм просмотра выходного результата вычислительной части. Реализует возможность просмотра результата оптимизации.

HTML вместе с JavaScript является отличной технологией создания интерфейсов и все вышеперечисленные функции удастся реализовать с минимальным использованием сторонних библиотек. На выходе должна получиться веб-страница формата HTML (которую возможно “запустить” на любой платформе, имеющей стандартные браузеры последних версий), позволяющая без использования удаленных серверов задать все параметры задачи и сохранить их в файл, который потом будет использован как входные данные для вычислительной части.

# Список использованных источников

1. Ant Colony Optimization (Marco Dorigo, Thomas Stützle, 2004): книга, печатное издание.
2. Ant Colony Optimization (Utkarsh Jaiswal, Shweta Aggarwal): <http://www.ijser.org/researchpaper/Ant_Colony_Optimization.pdf> [Электронный ресурс] Дата обращения: 01.02.15.
3. Scheduling of flexible manufacturing systems: an ant colony optimization approach (R Kumar, M K Tiwari and R Shankar): <http://www.et.byu.edu/~ered/ME482/Paper_Topics/TomAnt_ColonyOptimization.pdf> [Электронный ресурс] Дата обращения: 01.02.15.
4. Ant Colony Optimization routing algorithm for selecting Multiple Feasible Paths for Packet Switched Networks (Meenakshi R Patel, Babita Kubde): <http://ijcsi.org/papers/IJCSI-9-2-3-271-273.pdf> [Электронный ресурс] Дата обращения: 02.02.15.
5. An Ant Colony Optimization Algorithm for maximizing the lifetime of Heterogeneous Wireless Sensor Network (Saburi Uday Parvatkar and Deipali V. Gore): <http://inpressco.com/wp-content/uploads/2014/06/Paper1792116-2118.pdf> [Электронный ресурс] Дата обращения: 02.02.15.
6. Ant colony optimization with re-initialization (Matej Ciba, Ivan Sekaj): <http://article.sciencepublishinggroup.com/pdf/10.11648.j.acis.20130103.14.pdf> [Электронный ресурс] Дата обращения: 09.02.15.
7. Applying Ant Colony Optimization Algorithms to Solve the Traveling Salesman Problem: <http://www.codeproject.com/Articles/644067/Applying-Ant-Colony-Optimization-Algorithms-to-Sol> [Электронный ресурс] Дата обращения: 09.02.15.
8. HTML Web API, Canvas Tutorial: <https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial?redirectlocale=en-US&redirectslug=Canvas_tutorial> [Электронный ресурс] Дата обращения: 10.02.15.