Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Мотоциклетные графы и их свойства

Курсовая работа студента 445 группы
Мальчевского Михаила Андреевича

Научный руководитель К.В.Вяткина

Санкт-Петербург
2011

Содержание

1 Введение

Сначала введем понятие мотоциклетного графа. Пусть дано n мотоциклов на плоскости, каждый из которых имеет начальную позицию и заданную скорость. В момент времени 0 все мотоциклы начинают движение со своих начальных позиций с данными скоростями. Если мотоцикл A наезжает на след, оставленный другим мотоциклом B, то A разбивается и не может продолжать движение. Если два мотоцикла приезжают в одну точку, то оба разбиваются и останавливаются. После того, как все мотоциклы либо разбились, либо уехали на бесконечность, их пути сформировали планарный граф, называемый мотоциклетным.

2 Постановка задачи

Исследование свойств мотоциклетных графов представляется полезной задачей в связи с практическими применениями указанных. Сами по себе, мотоциклетные графы обобщаются на случай трехмерного пространства, где речь идет уже не о движущихся точках, а о прямых, движущихся в пространстве. Решение данной задачи может быть полезно в области трехмерного моделирования, анимации и компьютерных игр. В этих областях возникает необходимость в правильной и быстрой обработке столкновений объемных объектов.

Также решение задачи мотоциклетных графов используется в еще одной задаче вычислительной геометрии. Задаче о прямолинейных скелетах. У этой же задачи сразу возникают любопытные применения в жизни. Например, покрытие крыши черепицей.

В существующих работах по теме мотоциклетных графов основное внимание уделяется построению мотоциклетного графа для заданных мотоциклов (расположений и скоростей). Задача же этой работы – взглянуть на мотоциклетные графы с иной точки зрения, попробовать установить связь с обычными планарными графами. Данное исследование может помочь глубже понять природу мотоциклетных графов и усовершенствовать имеющиеся алгоритмы.

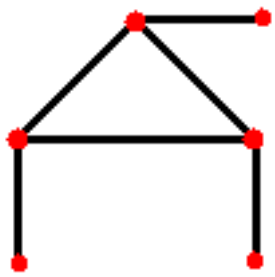
3 Теоретические результаты

3.1 Расстановка мотоциклов в висячих вершинах планарного графа

Рассмотрим следующую задачу: дан плоский граф G, имеющий r выделенных висячих вершин (т.е. степени 1), с общим числом вершин 2r. Можно ли сопоставить каждой из выделенных вершин такое значение скорости, что для r мотоциклов, начинающих движение из выделенных вершин с соответствующими скоростями, G является мотоциклетным графом?

Каждая из невисячих вершин должна получаться столкновением пары мотоциклов (либо один наезжает на след другого, либо оба приезжают к месту столкновения одновременно). Каждое из столкновений дает нам определенное неравенство на скорости мотоциклов. Каким-то образом нужно либо доказать, что данная система всегда имеет решение, либо опровергнуть, приведя контрпример.

Для данной постановки задачи легко можно найти опровергающий рисунок.



3.2 Расстановка мотоциклов во всех вершинах планарного графа

Сформулируем новую задачу таким образом: дан плоский граф; требуется определить, можно ли расставить в некоторых его вершинах мотоциклы и задать их скорости таким образом, чтобы данный граф был мотоциклетным для полученного множества мотоциклов.

Первоначально рассмотрим идею, которая поможет в дальнейшем при решении.

Идея для начального задания скоростей.

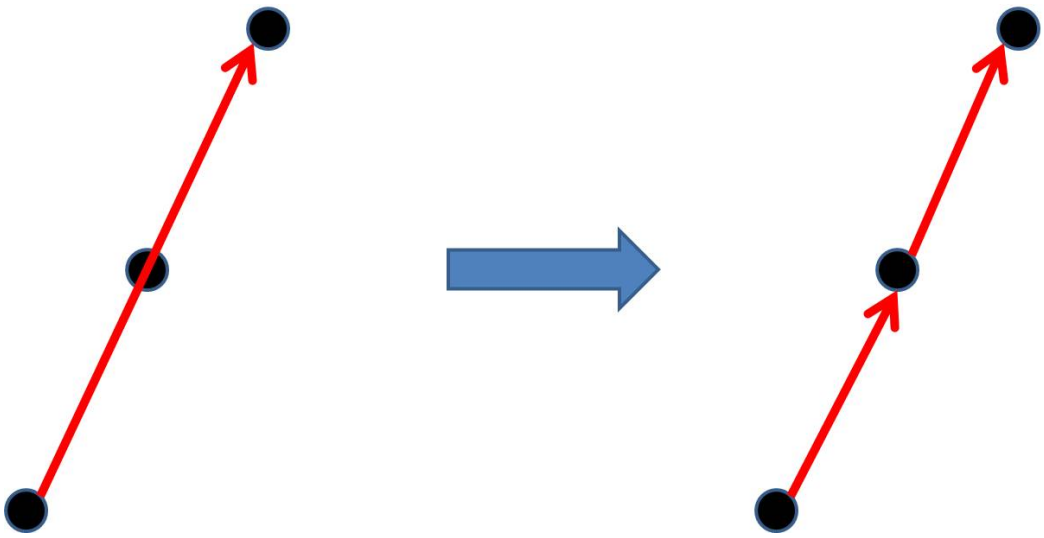
Если задано множество мотоциклов, стоящих в вершинах графа и заданы ребра, по которым им нужно проехать, то как правильно определить скорости, с которыми они должны начинать свое движение, чтобы получился правильный граф. Давайте зафиксируем некоторую единицу времени. И зададим всем мотоциклам скорости, при которых они за эту единицу времени проезжают полностью свое ребро. Давайте более строго докажем, что необходимые условия на скорости будут выполнены (все мотоциклы разобьются в правильных местах).

Устремим время к выбранной единице снизу и посмотрим на ситуацию с мотоциклами. Можно будет увидеть, что нарисован весь граф (кроме бесконечно малой его части), а все мотоциклы еще живы. Из этого также вытекает, что все мотоциклы разобьются там, где нужно.

Приступим теперь к решению самой задачи. Начнем с утверждения.

Утверждение 0. Из каждой вершины может выезжать не более одного мотоцикла. (Очевидно)

Утверждение 1. Если существует какая-нибудь расстановка мотоциклов, решающая задачу, то существует и такая расстановка, при которой каждый мотоцикл проезжает ровно одно ребро.



Док-во.

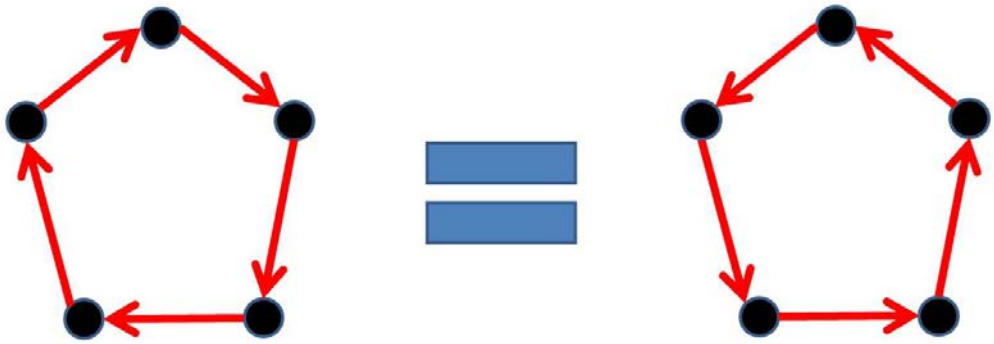
Рассмотрим какой-либо мотоцикл. Пусть он проезжает одно конкретное ребро и едет дальше (по ребру, выходящему из вершины и т.д.). Тогда по остальным ребрам, соединенным с этой вершиной, мотоциклы могут ехать только в направлении «к вершине» (см. Утверждение 0). Значит можно заменить рассматриваемый мотоцикл на два: один, которой проезжает одно ребро и разбивается (о путь второго), и второй, который начинает ехать с того места, где разбивается первый и продолжает бывший путь рассматриваемого вначале мотоцикла.

Итак, мы взяли мотоцикл, проезжающий более одного ребра, и заменили его на два, каждый из которых проезжает меньше ребер, чем данный изначально. Нетрудно видеть, что мы можем продолжать увеличивать количество мотоциклов, пока все они не станут проезжать ровно по одному ребру.
ЧТД.

Добавляем к утверждению первоначальную идею про задание скоростей графам. Теперь мы можем переформулировать начальную задачу в терминах «обычных» графов.

Дан неориентированный граф. Нужно узнать, можно ли ориентировать все его ребра таким образом, чтобы из каждой вершины выходило не более одного ребра.

Утверждение 2. Для каждого цикла исходного графа существует всего два способа ориентации ребер. Один – «по часовой», второй – « против часовой».



Док-во.

Рассмотрим какой-либо цикл исходного графа. Рассмотрим любое ребро, входящее в этот цикл. У этого ребра возможны две ориентации. Зафиксируем одну из этих ориентаций ребра. Возьмем вершину, из которой данное ребро выходит. Также у этой вершины существует и второе ребро, входящее в рассматриваемый цикл. Так как из одной вершины более одного ребра выходить не может, то второе ребро будет входящим в эту вершину.

Таким образом рассматривая ребра цикла по очереди, мы приходим к единственному варианту (при фиксированной ориентации первого ребра). Получаем, что ребра цикла могут иметь одну из двух ориентаций.
ЧТД.

Утверждение 3. Рассмотрим цикл в графе. То, в какую сторону ориентированы его ребра, не влияет на возможность решения первоначальной задачи.

Док-во.

Заметим, что все ребра, принадлежащие вершинам цикла и не входящие в сам цикл, будут направлены однозначно – к вершинам, вне зависимости от ориентации ребер в самом цикле. Становится видно, что мы можем ориентировать наш цикл как угодно, так как ориентация никак не повлияет на остальные ребра в графе.

ЧТД.

Теперь все готово для доказательства главной теоремы, к которой и готовили предыдущие утверждения.

Теорема. Для того, чтобы в графе нельзя было ориентировать ребра нужным образом необходимо и достаточно того, чтобы существовал путь из ребер, соединяющий два цикла (не обязательно разных – может цикл сам с собой)

Док-во.

Достаточность. Пусть существует путь, соединяющий два цикла. Требуется доказать, что тогда правильная ориентация невозможна. Будем доказывать от противного: пусть существует необходимая ориентация ребер этого пути.

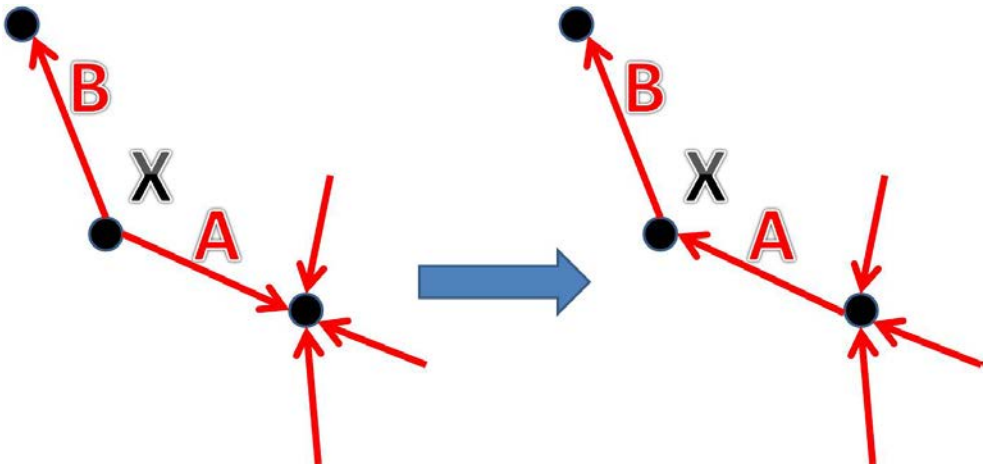
Рассмотрим ребро данного пути, которое выходит из вершины одного из циклов. Его ориентация определяется однозначно – «к циклу». Возьмем следующее ребро на этом пути. Так как из вершины, смежной для первого и второго ребер, уже исходит ребро, то ориентация второго также должна быть «к циклу» (к первому циклу!). С помощью метода математической индукции доказываем, что ориентация всех ребер на данном пути однозначна - «к первому циклу».

Но это означает, что последнее ребро пути должно исходить из какой-либо вершины второго цикла, чего быть не может. Получили противоречие, доказали достаточность.

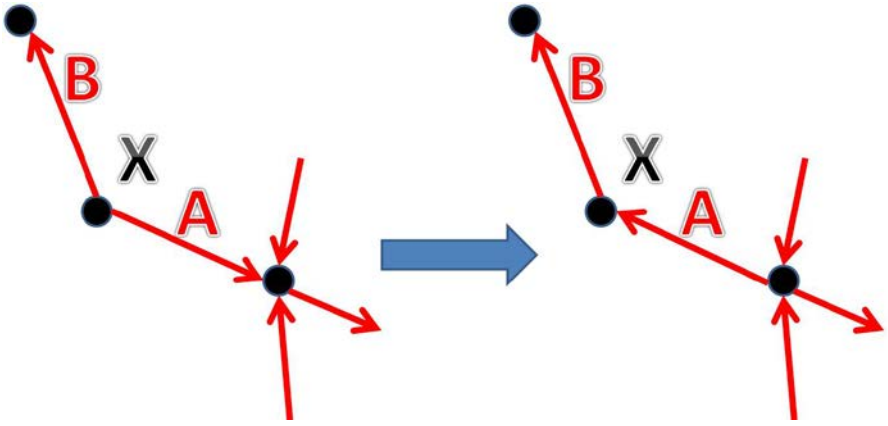
Необходимость. Нужно доказать, что ориентировать ребра нужным образом невозможно только в случае существования пути между циклами.

Рассмотрим процесс ориентации ребер. Предположим, что мы начали каким-либо образом ориентировать ребра и на каком-то из шагов натолкнулись на ситуацию, в которой из одной вершины (назовем ее X) выходят два ребра (то есть, с данной вершиной смежны два ребра, ориентация которых «от вершины»). Назовем эти ребра A и B.

Начнем рассмотрение с ребра A. Рассмотрим вершину, в которую ребро A входит. Переориентируем ребро A. Может возникнуть одна из двух ситуаций:

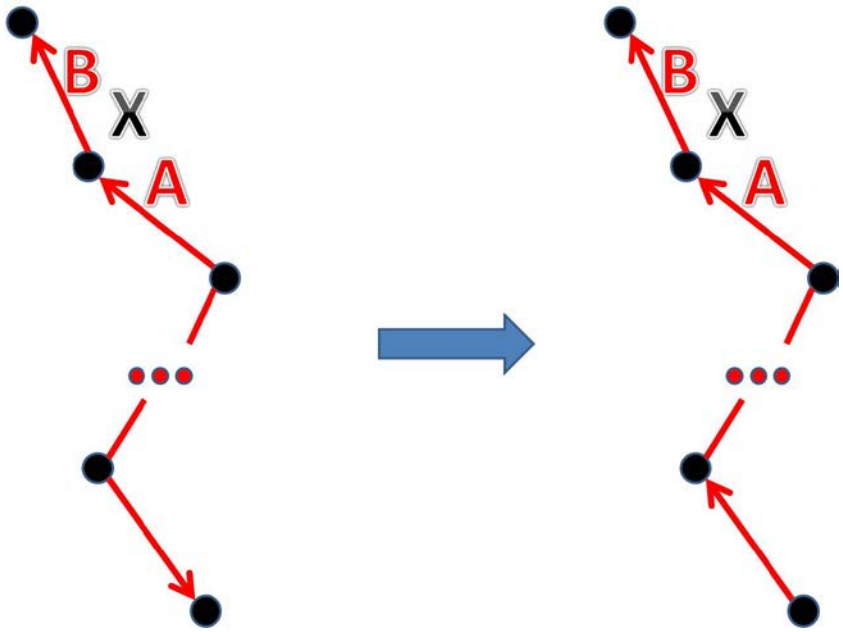


- После переориентации ребра A противоречий не возникло – все остальные ребра, смежные с вершиной, откуда A теперь выходит, входят в эту вершину. Значит, мы смогли разрешить противоречие и можем продолжать работать с ребрами, которые еще не ориентировали.

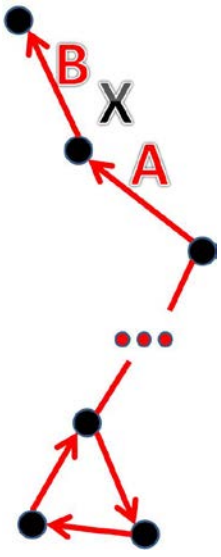


- После переориентации ребра A появилось новое противоречие – одно из ребер, смежных с вершиной, откуда выходит A, тоже выходит оттуда (больше одного противоречия возникнуть не может – если, например, на данном шаге из этой вершины исходят три ребра, то это означает, что и до этого было противоречие). Назовем ребро с противоречием C и перейдем в следующую вершину, повторяя рассуждения.

Продолжаем перемещать противоречие с помощью переориентации ребер от вершины X. С помощью индукции мы приходим к одному из двух различных состояний:



- Мы дошли до висячей вершины, с помощью последней переориентации устранив противоречие. Значит можно вернуться к ориентации оставшихся ребер.



- В какой-то момент мы зациклились – пришли в вершину, в которой уже были. Это означает, что вершина X соединена с каким-то циклом через ребро A. Продолжаем разбираться с проблемой.

Если мы не смогли устранить противоречие в вершине X через ребро A, то попробуем сделать это с помощью второго ребра, исходящего из этой вершины – ребра B.

Единственный вариант, при котором противоречие устранить не удастся – если на обоих концах мы найдем циклы – значит, в случае неустранимого противоречия мы нашли путь, соединяющий два цикла.
ЧТД.

3.3 Алгоритм расстановки мотоциклов

Из доказанной теоремы и предыдущих утверждений можно вывести алгоритм, который расставляет мотоциклы на графе (или говорит, что это невозможно).

1) Сначала нужно разобраться с циклами. С помощью поиска в глубину мы можем находить очередной цикл. С помощью другого поиска в глубину пытаемся ориентировать ребра правильно (выбираем любой из вариантов ориентации цикла, мы уже знаем, что разницы нет). Если попытка не удается, значит это невозможно.

2) После циклов остается некоторый лес (множество деревьев). Каждое дерево можно ориентировать (неоднозначно). Поиск в глубину помогает и здесь.

Получаем требуемый алгоритм ориентации. Так как каждое ребро и каждую вершину мы используем не более, чем константное, число раз, то сложность алгоритма – O(V + E), где V – число вершин, а E – число ребер.

3.4 Алгоритм расстановки минимального числа мотоциклов

Решаем новую задачу: в решение предыдущей задачи нужно расставить минимальное число мотоциклов. Должно быть понятно, что в этом случае мотоциклы могут проезжать более одного ребра.

Хотелось бы как-то модифицировать предыдущий алгоритм, чтобы он выдавал не любое решение, а оптимальное с точки зрения количества мотоциклов. Для этого докажем новые утверждения.

Утверждение 1. Каждая корректная ориентация ребер графа задает определенный, минимальный для этой ориентации, набор мотоциклов, который можно найти за время O(E).

Док-во.
Зафиксируем некоторую корректную ориентацию ребер. Она задает некую расстановку мотоциклов. Как получить из этой расстановки минимальную?

Рассмотрим некоторое ребро, а также соседнее, коллинеарное с ним. Вместо двух мотоциклов, по этим ребрам может проехать всего один. Таким способом можно разобраться со всеми коллинеарными ребрами. Очевидно, что данный набор мотоциклов минимален.

Алгоритм поиска этого минимального набора довольно прост. Будем по порядку рассматривать все ребра, «склеивая» коллинеарные и помечая обработанные. За необходимое время, мы обработаем весь граф.
ЧТД.

Рассмотрим компоненту связности какого-либо цикла. Что можно сказать про нее?

Утверждение 2. Для компоненты связности цикла любая корректная ориентация является оптимальной (после процедуры «сливания коллинеарных ребер»)

Док-во.
Мы помним, что для каждой такой компоненты существует всего два варианта ориентации ребер цикла, и нет вариантов для остальных ребер (они ориентируются однозначно). Значит нужно обратить внимание только на ребра цикла.

Рассмотрим оба варианта для ребер цикла. И в том, и в другом варианте соседние параллельные ребра будут сонаправленными, а значит коллинеарными. Значит оба варианта дадут одинаковый по количеству минимальный набор мотоциклов.
ЧТД.

Остается множество деревьев, с которыми нужно разобраться отдельно. Введем новое понятие.

Определение. Рассмотрим ориентированное дерево. Выберем любую вершину. Если у этой вершины есть исходящее ребро, перейдем по нему в другую вершину. Продолжим переходить из вершины в вершину (из каждой вершины выходит максимум одно ребро). Так как наш граф – дерево, мы обязательно либо придем в висячую вершину, либо попадем в вершину, из которой нет исходящих ребер. Такую вершину дерева будем называть **центром.**

Утверждение 3. У каждого ориентированного дерева существует единственный центр. Существует биекция между ориентацией ребер дерева и вершиной, которая является центром дерева.

Док-во.
Определение центра было конструктивным, поэтому он существует у любого дерева. Докажем единственность.

Пусть существуют два различных центра. Все ребра из компоненты связности первого центра должны быть направлены к нему. Так же и со вторым центром. Получается, каждый из центров не принадлежит компоненте связности другого центра, или они совпадают.

Для каждой ориентации существует единственный центр. Пусть есть неориентированное еще дерево. Выберем вершину, которая будет центром. Ориентируем все ребра по направлению «к ней». Так как наш граф – дерево, получили валидную ориентацию.
ЧТД.

Из последнего утверждения следует, что для того, чтобы найти оптимальную расстановку мотоциклов, можно просто перебрать все вершины, поочередно делая их центрами. Но есть ли более оптимальный способ?

Утверждение 4. Любая ориентация, в которой центр – висячая вершина, является оптимальной.

Док-во.

Будем доказывать от противного. Пусть существует ориентация, в которой центр – висячая вершина, и она не оптимальна.

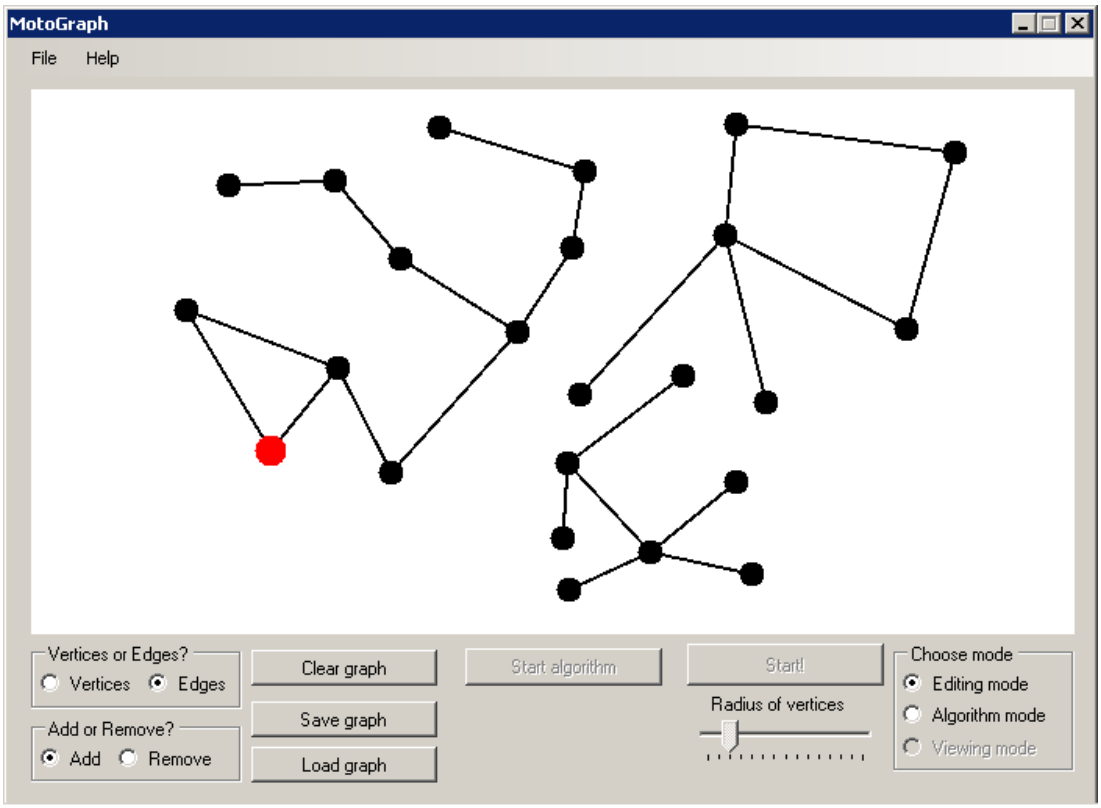
Рассмотрим какую-либо оптимальную ориентацию и подвесим ее за центр. Посчитаем расстояние от центра до висячей вершины. Сделаем следующее: рассмотрим первое ребро на пути от центра к висячей вершине, а также все последующие коллинеарные ему ребра. Все они проезжаются одним мотоциклом. Развернем ориентацию всех этих ребер, получив новую валидную ориентацию дерева, которая также является оптимальной. Итак, мы по данной оптимальной ориентации получили другую оптимальную ориентацию, расстояние от центра которой до висячей вершины строго меньше, чем в данной. С помощью подобных переворотов мы за конечное число шагов сможем переместить центр в висячую вершину, при этом сохранив оптимальность ориентации. Противоречие.
ЧТД.

Теперь, для поиска оптимальной ориентации вместо любой нужно модифицировать алгоритм следующим образом. Шаг для работы с циклами остается неизменным. После обработки всех циклов, снова получаем некоторое количество деревьев. Для каждого дерева с помощью поиска в глубину находим висячую вершину и делаем ее центром. После всех шагов, «сливаем» коллинеарные ребра.

4 Практическая реализация

Также мной была реализована программа, с помощью которой можно тестировать придуманный алгоритм. Она состоит из трех частей.

4.1 Редактирование графа



В данной части программы доступно редактирование графа с помощью мыши. Все действия производятся левой кнопкой.

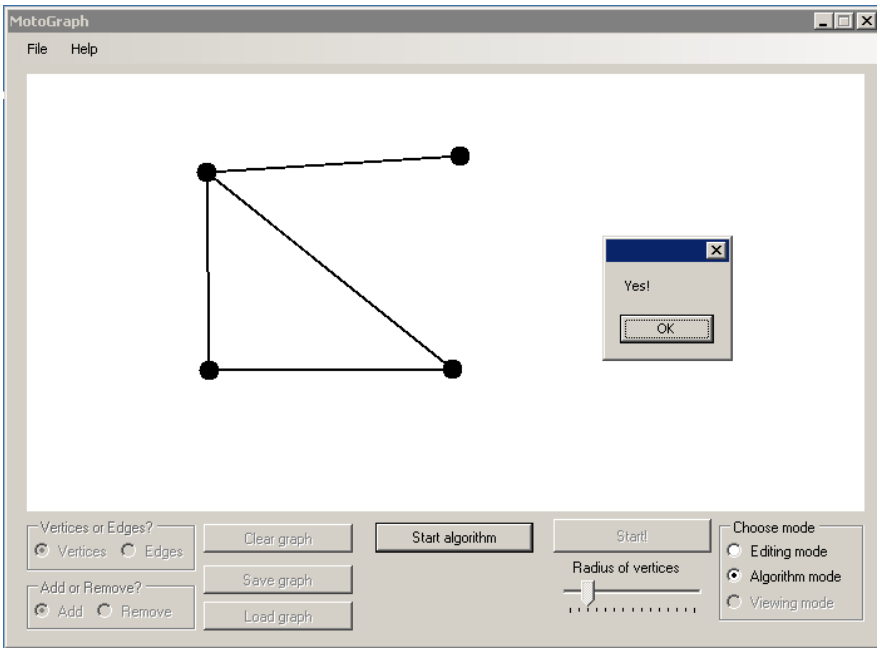
Для начала, нужно выбрать, что вы хотите редактировать – вершины или ребра графа. После чего следует выбрать, хотите ли вы добавлять новые элементы или удалять уже существующие.

Чтобы добавить вершину, просто щелкните мышью на свободном пространстве рабочей области. Для удобства введено ограничение, не позволяющее создавать новые вершины в непосредственной близости от старых. Для удаления вершины, просто щелкните по ней левой кнопкой мыши. Если вы удаляете вершину, к которой присоединены ребра, они удалятся вместе с ней.

Добавление и удаление ребер немногим сложнее. Нужно последовательно щелкнуть на первой вершине, затем на второй. После первого щелчка, выбранная вершина окрашивается красным цветом.

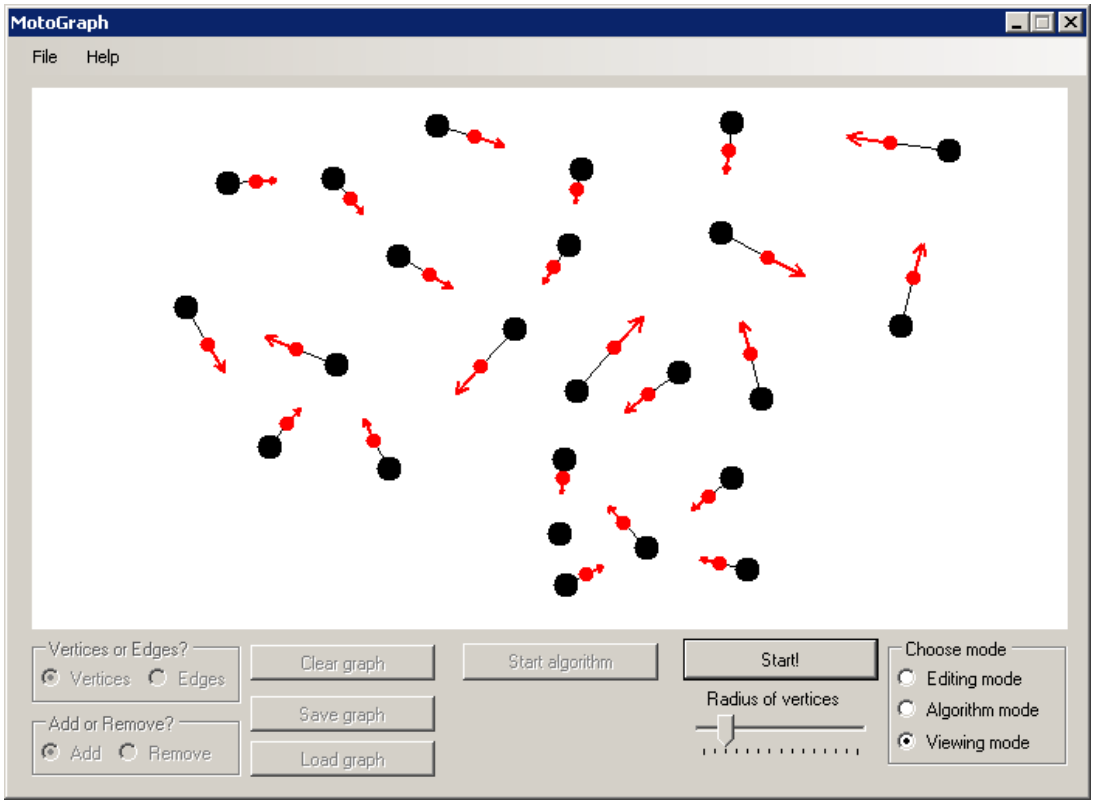
Также любой граф может быть сохранен на диск в формате *.gr и загружен с диска (из файла того же формата). С помощью кнопки «Clear graph» можно очистить рабочую область и начать создание с пустого графа.

4.2 Применение алгоритма



На втором шаге, шаге применения алгоритма, выбор доступных действий невелик. С помощью кнопки «Start algorithm» вы запускаете алгоритм, который проверит, можно ли корректно ориентировать ребра в созданном графе. Если можно, то расставит оптимальным образом мотоциклы. В случае успеха, пользователь увидит окно сообщения с надписью «Yes!», иначе «No…».

4.3 Визуализация мотоциклетного графа



Третий шаг, шаг визуализации мотоциклетного графа, становится доступен только в случае успешного завершения работы алгоритма на предыдущем, втором шаге.

На этом шаге отображается то, как едут мотоциклы. Изначальные вершины графа нарисованы черными кружками. Сами мотоциклы – красные кружки поменьше, со стрелками, показывающими вектор скорости. Мотоциклы оставляют за собой черный след, так что после завершения визуализации можно увидеть граф, который был создан на первом шаге.

Визуализация запускается кнопкой «Start!». Для удобства просмотра, под кнопкой расположен бегунок, регулирующий размер кружков, изображающих вершины, при просмотре. При последующих нажатиях на кнопку «Start!», визуализация повторяется с начального состояния.

5 Заключение

Были исследованы новые свойства мотоциклетных графов, доказаны необходимые утверждения и теоремы. На основе них были построены алгоритмы, позволяющие по обычному графу строить мотоциклетный граф. Эти результаты позволят в дальнейшем исследовать свойства мотоциклетных графов, прямолинейных скелетов и приложений.

На языке C# было реализовано приложение, позволяющее работать с произвольными неориентированными графами, проверять работу алгоритма, визуализировать мотоциклетные графы. Каждая из частей программ написана независимо от других, что позволит добавлять новые необходимые функции, алгоритмы. В дальнейшем, если понадобится, возможен переход на более мощные графические библиотеки для визуализации.

Список литературы

1 D. Eppstein and J. Erickson. Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions. Discrete Comput. Geom., 22:569–592, 1999.

2 Cheng S.-W., Vigneron A.: Motorcycle graphs and straight skeletons. Algorithmica 47, 2 (2007), 159–182