электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.021

Алгоритм обнаружения областей контроля на графе дорог

***Савченко А.С.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научные руководители: Рудаков И.В., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Тассов К.Л., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
irudakov@bmstu.ru

Введение

В современном мире широко применяются системы фото-видеофиксации – сложные комплексы, решающие целый ряд важных задач для обеспечения безопасности движения на дорогах. С их помощью становится возможным фиксировать факт нарушения правил дорожного движения и автоматизировать процесс взыскания штрафов с нарушителей. Также огромную роль играет психологический фактор предупреждения нарушений: водители, обладая информацией о расположении комплекса фото-видеофиксации на участке дороги, с меньшей вероятностью будут нарушать правила.

Такие системы полезны и при решении более важных с точки зрения безопасности задач, к примеру, обнаружение угнанных или разыскиваемых автомобилей по снимкам с камер. Также можно назвать задачу отслеживания транспортных средств подозреваемых. К поиску автомобилей, так или иначе, можно свести большой спектр задач, которые ежедневно возникают перед службами безопасности, начиная угонами, и заканчивая похищениями людей.

К сожалению, на данный момент большая часть операций для поиска конкретного транспортного средства выполняется вручную. Операторам необходимо выбрать камеры, через которые, возможно, мог пройти объект поиска, а затем изучать фотографии за какой-либо временной период.

Данная работа направлена на разработку алгоритма, который разобьёт дорожную сеть на области контроля, основываясь на информации о расположении комплексов фото-видеофиксации.

1. Постановка задачи

Целью работы было повышение скорости и точности поиска транспортных средств операторами. Для решения этой задачи была выбрана стратегия разбиения графа дорог на области контроля – простые циклы в графе, окруженные комплексами фото-видеофиксации. Таким образом, область контроля – это такая зона в графе дорог, которую транспортное средство не может покинуть или въехать в неё, не будучи зафиксированным на камеру комплекса фото-видеофиксации.

Введение областей контроля может значительно повлиять на качественные характеристики поиска автомобилей. Представим ситуацию: поступил запрос на поиск конкретного автомобиля с известным номером, и сказано, что он был замечен в центральной части города в районе полудня.

Операторам необходимо:

1. Определить набор камер, через которые мог бы пройти искомый автомобиль в заданный промежуток времени.
2. Просмотреть записи с выбранных камер.
3. Определить местоположение искомого автомобиля в заданный промежуток времени.
4. Спрогнозировать дальнейшее перемещение автомобиля, основываясь на данных из пункта 3.

Уже на первом шаге возникает вероятность, что оператор ошибется в выборе камер. Дорожная сеть представляет собой граф, в котором может быть очень большое число маршрутов, и человек физически не может просчитать все возможные пути для искомого автомобиля. Таким образом оператор может просматривать данные с камер, которые заведомо не могли фиксировать искомый автомобиль. Это понижает скорость поиска, приходится просматривать лишние записи. Также, есть вероятность, что оператор пропустит те камеры, которые могли бы зафиксировать транспортное средство. Это снижает точность поиска, поскольку некоторые записи могут быть пропущены.

При таком подходе возникает вероятность, что будут просмотрены многие записи, на которых искомый автомобиль не мог быть фиксирован. А записи, на которых автомобиль мог бы быть обнаружен, наоборот могут быть пропущены в силу человеческой нев-

нимательности. Граф дорог – это сложная математическая структура, и человеку физически невозможно построить все возможные варианты перемещения искомого автомобиля.

Если же дать оператору знания об областях контроля, то он может заменить выбор камер на выбор области контроля, и тем самым получить доступ сразу ко всем камерам, окружающим выбранную область. Таким образом, в набор просматриваемых оператором записей попадают те, на которых автомобиль мог быть записан, и не попадают такие записи, на которых его заведомо не может быть.

Также важно отметить, что на данный момент расположение камер в первую очередь решает задачу предотвращения нарушений правил дорожного движения, и, следовательно, комплексы установлены лишь на больших дорогах и проспектах. То есть для построения замкнутых областей контроля может попросту не хватить существующих камер. Здесь возникает вторая задача для алгоритма – это рекомендации по установке камер таким образом, чтобы были образованы области контроля.

Следовательно, требовалось разработать алгоритм, который должен принимать на вход следующие данные:

1. Граф дорог.
2. Координаты перекрестков.
3. Расположение комплексов фото-видеофиксации на графе дорог.
4. Количество недостающих камер для потенциальной области контроля.

Выходными данными являются:

1. Список найденных областей контроля.
2. Количество вершин в каждой области.
3. Количество комплексов фото-видеофиксации в каждой области.
4. Участки дорог, на которые необходимо установить комплекс фото-видеофиксации для организации области контроля (в случае потенциальных областей контроля).

2. Существующие решения

Как было сказано выше, область контроля – это простой цикл на графе дорог. Для этой задачи рассматривалась грубая модель дорожной сети – неориентированный граф (не учитывалось одностороннее движение). Следовательно, основным этапом работы разрабатываемого алгоритма является поиск простых циклов в неориентированном графе.

Цепь в неориентированном графе – это последовательность вершин (конечная или бесконечная) $v_0, v_1, \dots, v_n, \dots$, такая, что для любого i из v_i достижима v_{i+1}, если v_{i+1} существует [1].

Простая цепь – это цепь, все вершины которой, кроме, быть может, первой и последней, попарно различны и все ребра попарно различны [1].

Простую цепь ненулевой длины с совпадающими концами называют *циклом*.

Сегодня алгоритмы расчета простых циклов в графе не являются легкодоступными в учебниках и в Интернете. В ходе работы были изучены три алгоритма:

- Алгоритм Тарьяна.
- Алгоритм Шварцфитера и Лауэра.
- Алгоритм Пейтона.

Все рассмотренные алгоритмы требуют $O(V + E)$ памяти для хранения графа, где V – число вершин, а E – число ребер.

В худшем случае сложность алгоритмов по времени для нахождения циклов в ориентированных графов равна [2]:

- Тарьян - $O(V * E * \mathrm{C})$.
- Шварцфитер и Лауэр - $O(V + E * \mathrm{C})$.
- Пейтон - $O(V^3)$.

Где V – число вершин, E – число ребер, а C – число простых циклов в графе.

Худший показатель достигается на графах с особой структурой, поэтому на реальных задачах алгоритм с более высокой сложностью может превзойти алгоритм с более низкой сложностью [3].

В результате тестирования алгоритмов для поиска простых циклов указанные выше алгоритмы расположены следующим образом (от быстрого к медленному) [2]:

- Шварцфитер и Лауэр.
- Тарьян.
- Пейтон.

Несмотря на то, что алгоритм Шварцфитера и Лауэра обладает примерно одной скоростью с алгоритмом Тарьяна, он также является самым трудным для реализации. Алгоритм Пейтона способен обрабатывать графы с кратными ребрами – такая особенность не требуется в решаемой задаче. Был выбран алгоритм Тарьяна, поскольку он прост в реализации и обладает хорошей скоростью выполнения.

3. Разработка алгоритма

Важно отметить, что рассмотренные алгоритмы применимы лишь к ориентированным графам. В решаемой задаче области контроля строились на неориентированных графах, очевидно, что количество циклов в них может быть намного больше, чем в ориентированных [4]. Следовательно, требовалась доработка алгоритма.

Для того, чтобы алгоритм Тарьяна мог быть применен к неориентированным графам, были написаны следующие функции:

- Инвертирование – операция, в результате которой вершины цикла записываются в другом порядке.
- Нормализация – операция, в результате которой вершины цикла переставляются таким образом, чтобы вершина с наименьшим индексом стояла на первом месте.

Нормализация и инвертирование применяются к каждому найденному циклу и позволяют исключить хранение одинаковых циклов, вершины которых записаны в разном порядке. Если список найденных циклов не содержит нормализованной и инвертированной версии найденного цикла – то этот цикл объявляется новым и сохраняется.

Итак, в качестве основы для разрабатываемого алгоритма был взят доработанный под задачу алгоритм Тарьяна. В результате его работы на графе будет найдено множество замкнутых областей (рис. 2). Далее необходимо определить, какие из этих областей могут быть областями контроля.

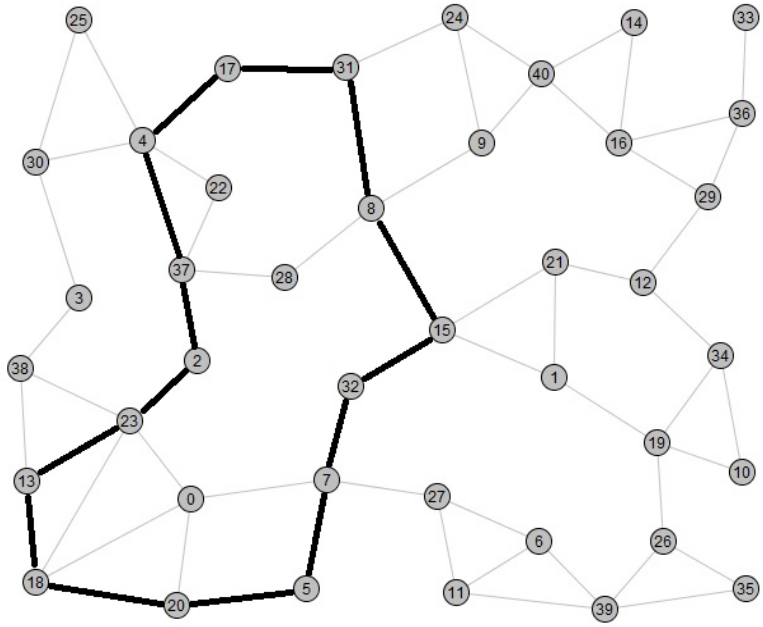

Рис. 1. Простой цикл на неориентированном графе

Было сказано, что область контроля – это замкнутая область, которую окружают комплексы фото-видеофиксации. На графе дорог они смоделированы с помощью взвешенных ребер, значение 1 означает, что на дуге есть комплекс фото-видеофиксации, а 0 – его отсутствие. Необходимо найти такие циклы, которые были бы окружены ребрами с весом 1.

На этом же этапе важно решить задачу добавления новых областей контроля. Пользователь может передать алгоритму число, которое будет означать допустимый недостаток комплексов фото-видеофиксации для каждой замкнутой области. Таким образом, если пользователь введет число 0, он получит все полноценные области контроля, поскольку недостаток камер указан нулевой. Если же он введет 2 – он получит все замкнутые области, которые могли бы стать областями контроля при добавлении на их границы 2 комплексов фото-видеофиксации (т.е. потенциальные области контроля).

Проверка замкнутой области состоит из следующих этапов (для каждой вершины):

1. Найти все вершины, которые доступны из данной за переход по одному ребру. Такие вершины считать соседями.
2. Рассматривать лишь те вершины, которые находятся вне цикла. Данная проверка производится по координатам – решается задача принадлежности точки к многоугольнику. Также исключаются вершины, которые являются вершинами данной замкнутой области.
3. Проверить ребра, соединяющие данную вершину и её соседей из пункта 2. Если на этом ребре нет камеры, то увеличить счетчик недостатка камер.
4. Если счетчик недостатка камер больше, чем допустимое значение, то замкнутая область не является областью контроля, иначе – является.

Для того, чтобы выделить все области контроля, очевидно, необходимо применить алгоритм проверки к каждому найденному простому циклу. Если функция возвращает истину, то данный цикл будет являться областью контроля, иначе – данный цикл не является областью контроля, следовательно, нет необходимости хранить его.

Таким образом, разработанный алгоритм состоит из модификации алгоритма Тарьяна, которая обнаруживает простые циклы в графе дорог, а также в проверке этих циклов, являются ли они областью контроля с допустимым недостатком комплексов фото-видеофиксации. Найденную область контроля можно увидеть на рисунке 2. Как видно, все ребра, окружающие простой цикл, помечены единицей, кроме одной. Можно сделать вы-

вод, что найдена потенциальная область контроля, и если на ребро {15, 1} будет установлена камера, то найденный цикл станет полноценной областью контроля.

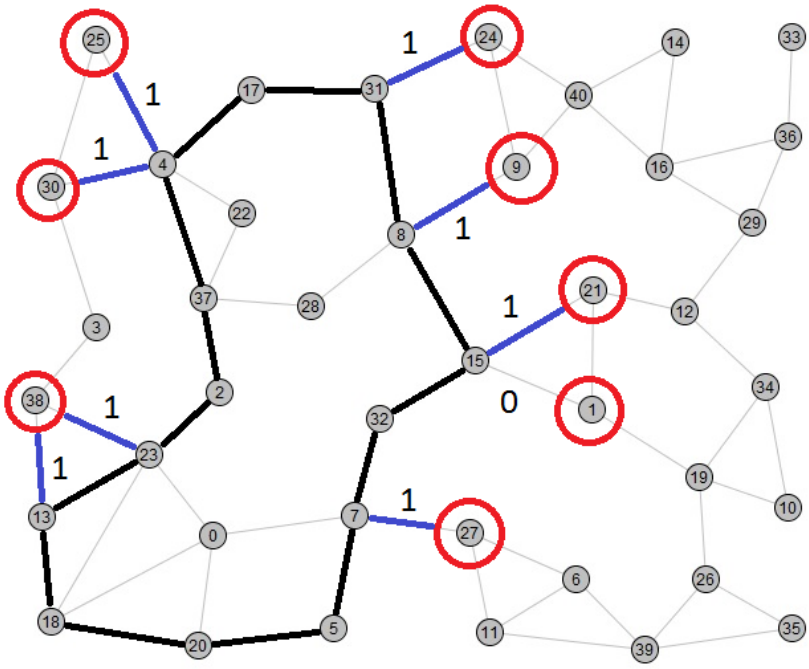

Рис. 2. Найденная потенциальная область контроля

4. Структура разработанного программного обеспечения

Для исследования разработанного алгоритма была создана система, которая его реализует. Разработанная система является сложным и расширяемым программным комплексом, поэтому было необходимо максимально эффективно использовать преимущества объектно-ориентированного подхода. Исходя из этих соображений для разработки был выбран язык Java 7. Язык позволил производить обход графов с достаточной скоростью, и при этом предоставит широкий спектр возможностей для работы с базой данных и для расширения программного комплекса. Для хранения данных была выбрана СУБД Oracle 10g Express Edition.

При разработке системы использовались следующие внешние библиотеки и платформы:

1. JUNG – хранение и обработка графов.
2. Swing – Графический интерфейс пользователя.
3. JUnit – тестирование.

Из рассмотренных библиотек необходимо выделить JUNG. В системе производится интенсивная обработка графов, поэтому необходимо во время исполнения хранить его в структуре данных, которая предоставляла бы широкий спектр функций для обработки

графа и была бы достаточно быстрой. В качестве структуры для хранения графа была выбрана Java Universal Network/Graph Framework (JUNG).

Разработанная система состоит из следующих модулей и компонент (рис. 3):

1. Представление графа (На основе JUNG).
2. Обработчик графа.
3. Графический интерфейс пользователя.
4. Модуль хранения графа дорог.
5. Хранилище графов и найденных областей контроля.

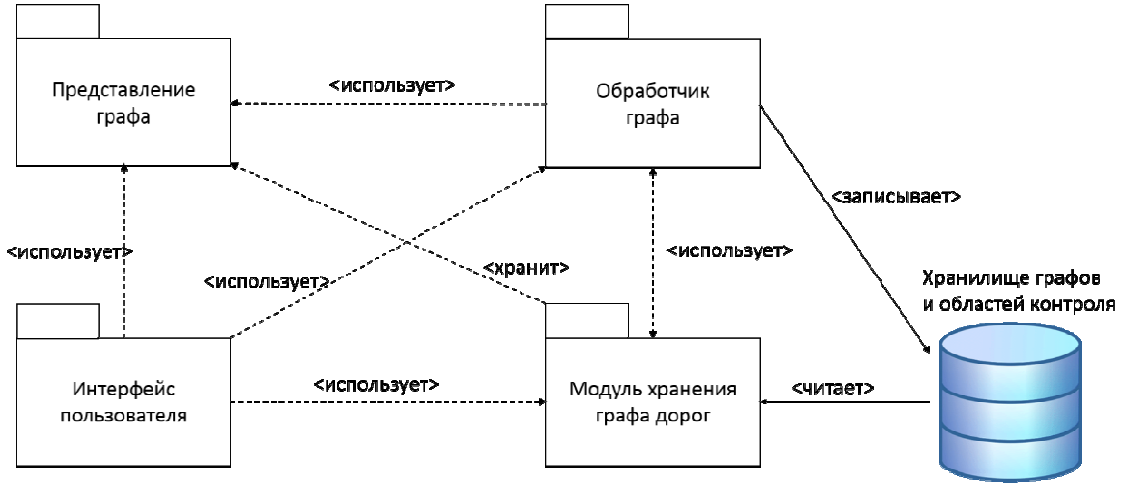

Рис. 3. Структура разработанного ПО

Сгенерированные и обработанные графы система позволяет сохранять в базу данных, и в любой момент загружаться из неё. Помимо самого графа и информации о расположении комплексов фото-видеофиксации, в базу также сохраняются все области контроля, которые были найдены в результате работы алгоритма, таким образом на больших графах не обязательно заново пересчитывать области контроля (расчет может занимать около часа и более), а достаточно их сохранить в базу, и выгрузить при необходимости.

5. Исследование алгоритма

Разработанный программный комплекс требователен к системным ресурсам и быстродействию системы в целом. Это связано с вычислительной сложностью алгоритма Тарьяна, используемого для поиска простых циклов. Этот процесс является самым затратным этапом всего алгоритма выделения областей контроля. Время работы алгоритма сильно зависит от количества вершин и ребер в графе, поскольку он основан на поиске в глубину. Для ускорения работы алгоритма в программе предусмотрен критерий оптими-

зации – ограничение на длину каждого простого цикла. Необходимо сравнить и проанализировать полученные результаты с использованием критерия и без него.

Целью исследования является определение влияния критерия оптимизации на точность алгоритма и на скорость его выполнения.

Определение скорости выполнения алгоритма производится с помощью использования встроенных в программу таймеров. Кроме времени выполнения также важно количество обнаруженных областей контроля. Сводные результаты исследования приведены в таблице. Эксперименты проводились на компьютере со следующими характеристиками:

- Процессор Core i5 2,4GHz;
- 6GB оперативной памяти;
- Операционная система Windows 8.1 x64.

Вершины / дуги	Ограничение на длину циклов	Время работы	Найденные области
68 / 103	-	4 мин 21 сек	6
68 / 103	13	0 мин 19 сек	6
68 / 103	12	0 мин 10 сек	5
90 / 132	-	8 мин 44 с	40
90 / 132	15	1 мин 22 с	28
90 / 132	10	0 мин 2 с	13

Первый из исследуемых графов состоял из 68 вершин и 103 ребер. Такая размерность является крайне высокой для поиска в глубину, на котором основан алгоритм Тарьяна. В результате первого эксперимента было выявлено эталонное время обработки графа, которое в дальнейшем улучшалось вводом критерия оптимизации. В качестве такого критерия рассматривалось ограничение размера простых циклов, поскольку алгоритм Тарьяна является самой затратной частью всего алгоритма, оптимизировать целесообразно именно его. Алгоритм основан на поиске в глубину с рекурсивной реализацией. В таком случае, можно ограничить глубину рекурсии – это и будет ограничение размера цикла. То есть пользователь сможет задать число, которое будет больше или равно числу вершин в любом полученном простом цикле. Таким образом сильно сокращается время выполнения алгоритма, а также отсеиваются заведомо лишние области контроля для пользователя. К примеру, если пользователь хочет найти лишь маленькие области размером в 5 вершин,

ему не придется ждать, пока поиск в глубину в алгоритме Тарьяна будет погружаться в рекурсию тысячи и десятки тысяч раз. Это очень простое и лаконичное решение придает возможность оптимизировать разработанный алгоритм как с точки зрения скорости его выполнения, так и с точки зрения удобства для пользователя.

Итак, было получено время 4 минуты 21 секунда для графа из 68 вершин и 103 ребер. Во второй строчке указано время выполнения алгоритма на том же графе, но с ограничением длины циклов в 13 узлов, оно составляет 19 секунд, это на 4 минуты меньше, чем в случае без критерия. В данном случае критерий оптимизации привел к ускорению алгоритма в 14 раз. Программа не дает никаких рекомендация для выбора значения критерия, это значение должно выбираться исходя из ожидаемого размера максимальной области контроля.

В следующем замере значение критерия уменьшено ещё на единицу – принято значение 12. Время поиска уменьшилось с 19 до 10 секунд, то есть ещё в 2 раза, а по сравнению с оригинальным временем без использования критерия оптимизации, время поиска уменьшилось приблизительно в 28 раз. Но также в таблице приведено количество обнаруженных областей контроля, оно равно 5. С ограничением длины циклов равным 13, было обнаружено 6 областей. Таким образом мы получаем погрешность выходных данных при использовании критерия оптимизации. Далее в ходе исследования будет показана зависимость между размером критерия оптимизации и величиной погрешности выходных данных. В данном случае погрешность составила 12% - это довольно большая погрешность, но следует учитывать малый размер выходных данных.

Был сгенерирован второй граф, он состоял из 90 вершин и 132 ребер. Это ненамного больше, чем первый граф, но тем не менее из 4 строчки таблицы видно, насколько более длительно он обрабатывается. Время выполнения составило 8 минут 44 секунды, что на 52% больше, чем в первом эксперименте. Важно сказать, что количество вершин увеличилось с 68 до 90 (на 24%), количество ребер с 103 до 132 (на 30%), но время выросло в 2 раза. Это объясняется высокой сложностью по времени алгоритма Тарьяна, в основе которого лежит рекурсивный поиск в глубину.

Далее будет продемонстрировано использование критерия оптимизации. Для этого максимальный размер циклов был взят 15 (строка 5). Можно наблюдать увеличение скорости алгоритма приблизительно в 7 раз – 1 минута 22 секунды вместо 8 минут 44 секунд. Но в этой ситуации мы снова сталкиваемся с большой погрешностью, вместо 40 областей, обнаруженных в первом эксперименте, здесь было выделено 28 – это на 32% меньше. Можно увидеть зависимость между скоростью выполнения и погрешностью – чем выше

скорость, тем выше погрешность. Последняя строка в таблице подтверждает это утверждение. Здесь было получено время 2 секунды, это в 262 раза быстрее, чем в первом эксперименте (8 минут 44 секунды). Такой результат компенсируется упущением многих областей контроля – было выявлено 13 вместо 40 в первом эксперименте, разница составляет 31%. Эта разница ощутима, если сравнить покрытие графа областями контроля визуализированном выводе графа в системе.

В эксперименте не были рассмотрены большие графы в силу того, что их обработка без критерия оптимизации может занимать вплоть до нескольких часов.

В таблице наглядно видна зависимость времени работы алгоритма от размера критерия оптимизации. Чем меньше ограничение на длину циклом – тем менее глубоко уходит в рекурсию алгоритм Тарьяна, таким образом получается выигрыш в скорости. Но при этом не будут рассмотрены области больших размеров, и поэтому имеется погрешность в виде областей контроля, упущенных из рассмотрения в силу применения критерия оптимизации.

Заключение

В данной работе был разработан алгоритм обнаружения областей контроля, которые должны увеличить скорость и точность поиска транспортных средств с помощью комплексов фото-видеофиксации. Алгоритм был исследован, был введен критерий оптимизации и исследовано его влияние на точность и скорость работы алгоритма. Также был разработан программный комплекс, состоящий из настольного приложения и базы данных, который позволил провести эксперименты.

На данный момент алгоритм обладает рядом недостатков, например, он работает только с неориентированным графами. Также он не может предоставить разные варианты разбиения графа дорог на области контроля и выбрать оптимальное по заданному критерию, поскольку он находит все области контроля за раз. Нет рекомендаций по выбору значения критерия оптимизации.

Алгоритм и программный комплекс требуют дальнейшей доработки, и будут улучшены в скором времени.

Список литературы

1. Белоусов А.И., Ткачев С.Б. Дискретная математика: учеб. для вузов / под ред. В.С. Зарубина, А.П. Крищенко. 3-е изд. М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. 744 с. (Сер. Математика в техническом университете; Вып XIX).

2. Niographs – A JAVA library of graph algorithms. Available at https://code.google.com/p/niographs, accessed 13.06.2014.
3. Reactorweb. Finding all polygons in an undirected graph. Available at http://blog.reactoweb.com/2012/04/algorithm-101-finding-all-polygons-in-an-undirected-graph, accessed 16.06.2014.
4. Alan Tucker. Applied Combinatorics, Hoboken, New Jersey: Wiley, 2012. 496 p.