Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова
Факультет информатики и вычислительной техники

Заметки по информатике и математике

Сборник научных статей

Выпуск 7

Ярославль
ЯрГУ
2015

УДК 004:51(082)
ББК В1я43+397я43
З26

Рекомендовано
Редакционно-издательским советом университета
в качестве научного издания. План 2015 года

Заметки по информатике и математике : сборник
З26 научных статей / под ред. А. Н. Морозова ; Яросл. гос. ун-т им. П. Г. Демидова. — Ярославль : ЯрГУ, 2015. — Вып. 7. — 124 с.

ISBN 978-5-8397-1072-6

В сборник включены научные статьи студентов и аспирантов факультета ИВТ, посвященные важным направлениям развития математики, информатики и вычислительной техники.

УДК 004:51(082)
ББК В1я43+397я43

ISBN 978-5-8397-1072-6

УДК 004.422

Разработка информационного портала о культурных достопримечательностях Новой Москвы

С. В. Аверкиев

Ярославский государственный университет им. П. Г. Демидова
E-mail: sergey.averkiev.w@gmail.com

В статье рассматривается процесс разработки информационного портала о культурных достопримечательностях Новой Москвы. Описана архитектура приложения, решения, используемые для реализации его функционала, взаимодействие с внешним сервером.

Ключевые слова: Ruby on Rails, CoffeeScript, информационная система.

Новая Москва — это регион, расположенный на юго-западе Москвы. В этом месте сосредоточены разнообразные интересные места: от музеев и выставочных залов до катков и спортивных площадок. Для привлечения внимания людей к этому региону департаментом культуры г. Москвы было решено создать веб-портал, посвященный местным достопримечательностям.

Информация должна храниться в виде трех сущностей.

1. Объект — это отдельно взятое место, представляющее потенциальный интерес для посещения. Объекты характеризуются описанием, информацией о местоположении, интересными фактами для пробуждения интереса.

2. Маршрут — это набор объектов, объединенных по тематике или местоположению. Маршруты дают возможность пользователю взглянуть на набор объектов, чтобы при изучении одного объекта он мог найти другие — похожие и расположенные поблизости.

3. Событие — мероприятие связанное с объектом, служит для привлечения внимания к объектам, для вовлечения пользователя в изучение их истории путем некоего интерактива.

В процессе разработки портала было необходимо сформулировать техническое задание, реализовать взаимодействие с двумя точками получения данных, реализовать поддержку мобильных устройств, разработать модуль «Избранное» с возможностью построения собственных маршрутов.

Для написания приложения был выбран язык Ruby с использованием фреймворка Ruby on Rails. Фреймворк Ruby on Rails используется для написания приложений с архитектурой «модель — представление — контроллер». Модель отвечает за хранение и обработку данных. Видом являются шаблоны страниц. Контроллер служит для обработки запросов клиента. Он может выполнить некоторые операции над данными и отправить ответ браузеру. Для создания визуальной части сайта используется HAML, SASS и CoffeeScript.

Приложение способно получать данные из двух точек получения данных. Объекты и маршруты расположены на внешнем сервере. События расположены локально и управляются Ruby on Rails.

Точки для получения данных по данным сущностям предоставляют сторонние разработчики. У них уже было разработано ядро для хранения данных и обработки поисковых запросов. JSON-структуры для получения данных были предоставлены сторонними разработчиками. Необходимо реализовать взаимодействие с этим ядром, расположенным на внешнем сервере. Информацию об объектах и маршрутах приложение получает, отправляя к внешнему серверу AJAX-запросы и получая ответы в виде JSON.

Для хранения и обработки запросов для поиска событий используются возможности самого приложения Ruby on Rails. С этой целью была разработана база данных, содержащая таблицу событий. Данные о событиях отображаются указанием в шаблонах ссылок на соответствующие данные. При компиляции HAML-шаблона Ruby on Rails подставляет в указанные места данные. Приложение предоставляет REST-интерфейс для получения данных о событиях.

Главным назначением CoffeeScript является отображение полученных данных на страницах приложения. Для этого используется библиотека KnockoutJs. При загрузке любой из страниц со-

здается модель Knockout, содержащая данные, которые необходимо отобразить на странице. Данные для моделей Knockout передаются из внешнего сервера или из локального хранилища браузера. После того как модель была создана, в HAML-шаблонах указываются привязки с помощью атрибута data-bind. В качестве значения атрибута передается строка в следующем формате: «название_привязки: значение». В качестве названий привязок могут использоваться text для отображения текста, html для отображения HTML-документа, onclick для выполнения функции при нажатии на элемент. В качестве значения могут передаваться поля модели и статические значения или функции. На страницах приложения для отображения данных автор создает привязки к данным модели на соответствующих элементах. Таким образом, при обновлении данных модели Knockout автоматически обновит шаблоны.

Для каждой функциональности, реализуемой с помощью CoffeeScript, повторяющейся на нескольких страницах, создается отдельный модуль. Код из этого модуля может использоваться внутри любых других модулей, подключаемых на страницах.

Разработаны следующие модули.

1. modules. Содержит объявление глобальной переменной NM, в поля которой записываются объекты модулей. Этот файл всегда загружается первым, поэтому переменная доступна в любое время.

2. startup. Служит для запуска кода файлов после готовности документа. Его можно исполнять при наличии определенного элемента на странице. Так как библиотека Sprockets, которую использует фреймворк, загружает все модули, нужно определить, что именно из загруженного нужно исполнить, поэтому на каждой странице есть элемент с уникальным идентификатором. В итоге код, относящийся к определенной странице, будет выполнен только на ней.

3. maps. Служит для создания и управления картами. С его помощью можно создавать маркеры, линии или круги для указания местоположения объектов или маршрутов, регулировать отображения подложек.

4. cookies. Стандартный интерфейс для управления cookies не слишком удобен. Код данного модуля содержит функции-обертки для легкого управления cookies.

5. dialogs. Позволяет заменять стандартные диалоги подтверждения на аналогичные настраиваемые. Для соответствия внешнего вида диалогов общему виду приложения было необходимо настроить их отображение, а также код для обработки нажатий на кнопки подтверждения или отмены.

6. pagination. Содержит код, позволяющий применять постраничное отображение моделей Knockout.

7. utils. Содержит функции, не относящиеся непосредственно к определенной функциональности, но используемые для некоторых операций. Например, для обрезания картинок для отображения их внутри на главной странице в форме квадратов.

8. dao. Служит для отправки JSON-запросов на сервер с данными и обработки полученных данных.

9. knockoutBindings. Модуль, содержащий дополнительные привязки Knonkout. Например, здесь описана привязка для загрузки обрезанной картинки для отображении в фиксированной области. Привязка применяется с указанием названия и параметров. Используя дополнительные привязки, можно избежать дублирования кода.

10. favorites. Служит для управления списком избранного.

11. imageGallery. Позволяет отображать картинки в виде галереи.

CoffeeScript управляет динамическими элементами страниц. Для каждой страницы существует модуль для реализации динамики. После загрузки страницы исполняется код соответствующего ей модуля. Для этого используется модуль startup. Его задачей является исполнение кода CoffeeScript при обнаружении элемента, указанного с помощью идентификатора или класса. Каждый модуль содержит код с объявлением функций и вызовом функции whenLoaded. Ее параметрами являются идентификатор или класс элемента, при обнаружении которого необходимо исполнить код, и функция обратного вызова с кодом, который необходимо исполнить. В эту самую функцию обратного вызова можно поместить вызовы объявленных ранее функций. Таким образом код, помещенный внутри второго парамента-функции whenLoaded будет выполнен только при обнаружении определенного элемента на странице. В дальнейшем код страницы помещается внутрь элемента с идентификатором, который находится только на этой

странице. Затем вызывается whenLoaded из модуля startup и передается в качестве первого аргумента этот уникальный идентификатор, а в качестве второго — функция с кодом, который необходимо выполнить только на этой странице.

Если пользователь использует смартфон для доступа к сайту Новой Москвы, то он увидит страницы, адаптированные для использования на данном устройстве. Страницы мобильной версии сверстаны в виде одной колонки. На каждой странице есть разворачиваемая боковая панель с формами для поиска объектов, маршрутов и событий. Также для удобства использования сайта предусмотрены кнопки для разворачивания скрытых элементов. Таким образом, например, можно развернуть или свернуть список интересных фактов на странице объекта при нажатии на кнопку «Интересные факты». Разворачиваемые элементы реализованы с помощью Bootstrap JavaScript-кода. Мы отмечаем в шаблоне в атрибутах элементов, что хотим сделать этот элемент разворачиваемым, и привязываем к нему соответствующую кнопку.

Портал позволяет сохранять объекты, маршруты и события, понравившиеся пользователю. На каждой странице данных элементов расположена кнопка «В избранное». При нажатии на кнопку происходит сохранение элемента в списке избранного. Список избранного хранится в поле favorites локального хранилища браузера в JSON-формате.

Для отображения элементов из списка избранного создана страница, разбитая на вкладки. Для каждого из списков объектов, маршрутов и событий создаются модели Knockout. Они действуют так же, как и на других страницах, только данные получают не из внешнего сервера, а из списка избранного. Для извлечения массива элементов используется функция getFavoriteElement, которой в качестве ключа передается название извлекаемой сущности.

На основе избранных объектов реализована возможность строить собственные маршруты. Для построения маршрутов создана специальная форма, где можно указать название, описание и тип маршрута, выбрать подходящие для построения объекты и посмотреть предварительный результат на карте.

При переходе на страницу построения маршрута создается модель Knockout со списком избранных объектов. В модели

определены два списка: активные и неактивные объекты. С данной моделью связаны списки в видовом шаблоне. При перемещении объекта из списка в список (нажатием на кнопки добавить/удалить или drag and drop) производится изменение состояния модели, после чего Knockout обновляет вид.

На карте можно увидеть выбранные на данный момент активные объекты. Внутри маркеров обозначен порядок объекта внутри списка. Для построения маршрута необходимо выбрать службу, с помощью которой будет построен маршрут. На выбор предлагаются Yandex, Google и Open Street Maps. Затем нажатием на кнопку <<построить маршрут>> отправляется запрос в одну из служб и в качестве параметров передаются координаты объектов из списка активных. В ответ приходит массив координат промежуточных точек. Маршрут отображается на карте с использованием полученных координат.

Список созданных маршрутов отображается на странице избранного. При нажатии на маршрут из списка открывается его страница. Так как поля созданного маршрута идентичны принимаемому, есть возможность использовать ту же knockout-модель, что используется для отображения маршрутов, принимаемых с внешнего сервера. В этом случае поля модели заполняются из локального хранилища.

Ссылки

1. Hogan B. P. Web Design For Developers // The Pragmatic Programmers. LLC, Raleigh, NC, and Dallas, TX. 2009. 300 с.

2. Duckett J. HTML and CSS: Design and Build Websites. John Wiley & Sons, 2011. 496 с.

3. Valim J. Crafting Rails Applications: Expert Practices for Everyday Rails Development. Pragmatic Bookshelf, 2013. 324 с.

УДК 004

Моделирование и анализ свойств протоколов распространения данных в распределенных системах с использованием раскрашенных сетей Петри

В. Г. Агаджанова, Д. Ю. Чалый

Ярославский государственный университет им. П. Г. Демидова
E-mail: agadvika@yandex.ru, chaly@uniyar.ac.ru

Формализм раскрашенных сетей Петри и их применение является актуальным направлением исследований в современной науке, что подтверждается наличием большого количества статей по этой тематике. Многие авторы этих статей используют CPNTools в качестве инструментального средства моделирования и анализа построенных моделей. В нашей статье рассматриваются протоколы из семейства Gossip-протоколов, которые широко используются для распространения информации в современных распределенных системах.

Основным результатом работы является оригинальная модель протокола и анализ ее функционирования на примере нескольких топологий коммуникационных сетей.

Ключевые слова: раскрашенные сети Петри, сети Петри, CPNTools, Gossip-протокол, Gossip-basedprotocol, ColouredPetriNets.

Введение

Одним из ярких примеров применения сетей Петри является работа [1]. В ней на простом модельном примере рассмотрено применение вложенных сетей Петри для моделирования распределенных систем. Авторами было определено расширение формализма вложенных сетей Петри за счет приписывания некоторым переходам дополнительных условий срабатывания. Приведены условия, при которых использование условий срабатывания на переходах сохраняет свойства базовой модели, в частности разрешимость некоторых поведенческих свойств.

В работе [2] авторы предлагают проводить имитационное моделирование и построение графа достижимости для NP-сетей путем перевода NP-сетей в раскрашенные сети Петри и использования инструментария CPNTools в качестве виртуальной машины для исполнения и средства автоматического анализа исходных NP-сетей.

В работе [3] авторы промоделировали и проанализировали протокол управления передачей (TCP) с помощью раскрашенных сетей Петри (CPN). Они представили свою CPN-модель и примеры того, как можно изучить проблемы производительности протокола TCP. Показали, как можно модифицировать эту модель, чтобы представить протокол ARTCP. Авторы утверждают, что их модель может использоваться в качестве основы для построения формальных моделей будущих модификаций протокола TCP.

В работе [6] авторы поставили перед собой цель изучить полезность раскрашенных сетей Петри (CPN) и CPNTools для анализа безопасности защищенного модуля Trusted Platform Module (TPM). В качестве примеров они взяли Object-specific authorization protocol (OSAP) и Session Key Authorization Protocol (SKAP).

Целью работы [7] являлась разработка комплексной модели связи для сети взаимодействующих спутников, при использовании сетей Петри, позволяющих реконфигурировать подсети в тот момент, когда появляются спутниковые ошибки. Для моделирования взаимодействия спутников в сети использовались раскрашенные сети Петри (CPN).

В статье [8] описано применение моделирования многоагентных систем. Авторы используют многоуровневый подход, основанный на раскрашенных сетях Петри для моделирования сложных одновременных разговоров среди агентов в многоагентных системах.

Все это демонстрирует, что проблематика коммуникационных протоколов, их моделирование и анализ являются актуальной проблемой, а раскрашенные сети Петри представляют собой подходящий формализм для проведения анализа этих объектов.

Протокол Gossip

Протокол Gossip — это протокол для распределенных систем, состоящих из равноправных узлов. Данный протокол используют в распределенных системах потому, что его специфика проста и он имеет удобную структуру, в отличие от ряда других

протоколов распределенных систем. Чаще всего алгоритм данного протокола конструируют таким образом, что узлы взаимодействуют главным образом с соседними узлами и лишь иногда с узлами, которые находятся далеко. Этим протоколам необходимо обеспечить достаточную степень связности, чтобы избежать риска отключения узла от сети.

Gossiping — это альтернатива классическому методу отправки пакетов, который использует случайный выбор, чтобы снизить потребление ресурса в коммуникационной системе. Вместо того чтобы без разбора выполнять пересылку данных на всех своих соседей, узел передает данные только на одного случайно выбранного соседа. Если Gossiping-узел получает данные от соседа, он может переслать данные обратно соседу, если он случайным образом выбрал его.

Вариации протокола Gossip:

- Pushepidemic — узел передает имеющуюся информацию случайно выбранному соседу;

- Pullepidemic — узел запрашивает информацию у смежных узлов.

Модель протокола Gossip

Система CPNTools позволяет создавать иерархические модели, т. е. модель внутри модели. На рис. 1 видно, что в нашей модели содержатся еще три модели, они представлены на основной странице Top в виде переходов Init, Add и Check. В первом переходе происходит инициализация, во втором — добавление (при надобности) новых элементов в систему, в третьем — осуществляется проверка и передача информации узлам, которые ранее ею не обладали.

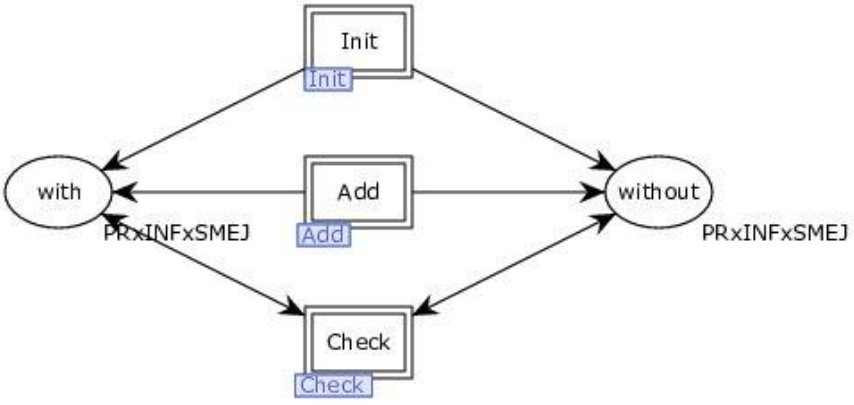


Рис. 1. Модель протокола Gossip

Как видим, позиции with и without имеют тип PRxSMEJxINF — он моделирует узел, список смежности и данные о наличии информации. Причем with содержит фишки, моделирующие узлы, обладающие информацией, а without — фишки, моделирующие узлы, не обладающие информацией. Рассмотрим подробнее подсеть, в которой происходит передача информации.

Основной подсетью, моделирующей обмен информацией, является Check, здесь происходит передача информации от узлов, содержащих информацию, к узлам, не содержащим информацию.

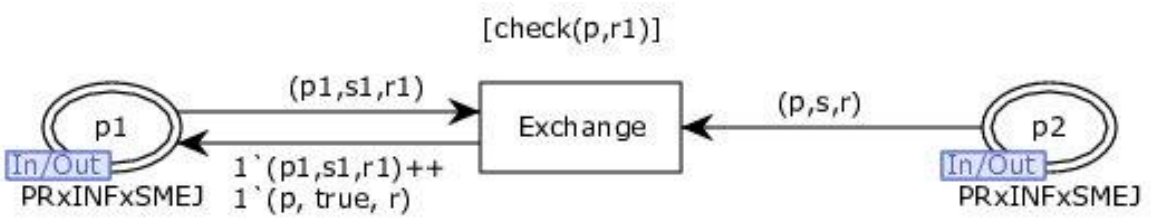


Рис. 2. Модель передачи информации от одного узла другому

Узел p1 — это тот, в котором содержатся узлы, имеющие новую информацию. Узел p2 — узел, в котором содержатся узлы, не содержащие новой информации. При срабатывании перехода Exchangemы вызываем функциюcheck (рис.3).

```
fun check (p, r) =
  if r<>nil
  then (if hd r=p
  then true
  else  check (p, tl r))
  else false
```

Сеть случайным образом выбирает один элемент из p1 и один элемент из p2 и далее проверяет, есть ли у элемента p1 в списке смежных элемент p2. Этим занимается функция check.

Рис. 3. Функция check

Если она находит этот элемент, тогда у элемента из узла p2 изменяется значение inf на true и он переходит в узел p1. Если не находит, она берет следующий элемент.

Сеть продолжает свою работу до тех пор, пока все элементы из узла p2 не станут обладать информацией и не перейдут в узел p1.

Рассмотрим две модификации модели. Модель со временем (рис. 4) и модель с ограниченным количеством передаваемых сообщений (рис. 5).

На рис. 4 представлена первая модификация нашей модели. О ней говорилось в пункте 2. С помощью этого улучшения можно теперь проследить, какое количество времени требуется программе, для того чтобы завершить процесс. В данном языке знак

@- является знаком времени. Время меняется в зависимости от задаваемых параметров. Время можно увидеть в списке деклараций, рядом с количеством шагов.

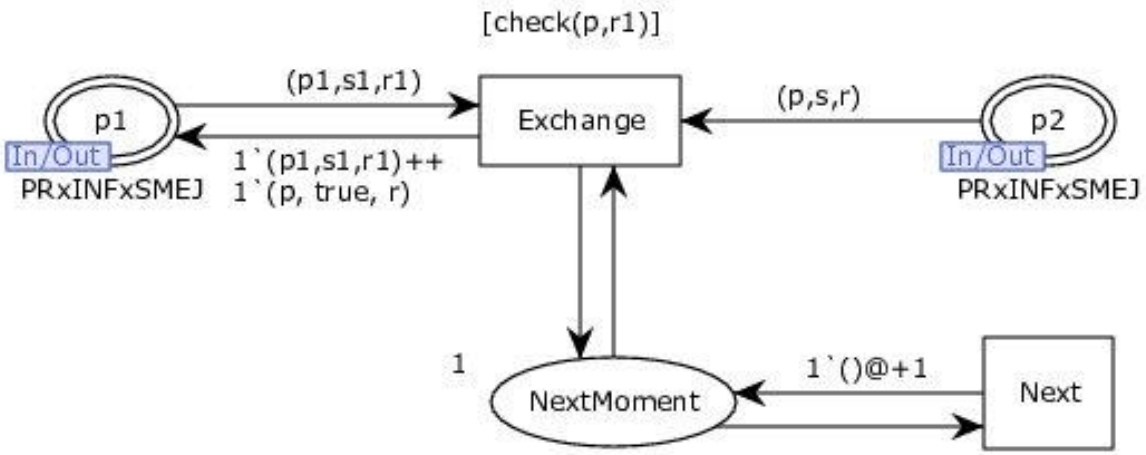


Рис. 4. Модификация модели

За определенный момент времени переход Exchange успевает сработать множество раз, при добавлении позиции NextMoment сеть сама решает, когда наступает следующий момент времени. В NextMoment из Exchange поступает фишка, и она дальше перемещается в переход Next. Значит, наступил другой момент времени, у перехода Next увеличивается счетчик, а фишка возвращается обратно в сеть. На рис. 5 мы видим модифицированную подсеть Check.

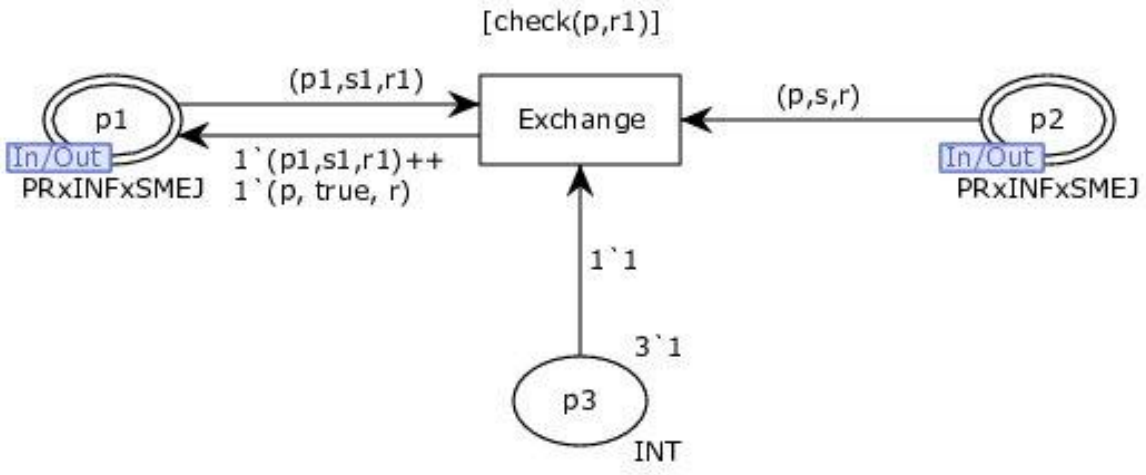


Рис. 5. Модификация модели с ограничением количества передаваемых сообщений

На данной подсети появилась новая позиция p3. Как видим, она имеет обычный числовой тип данных и имеет фишку 3`1. Это

означает, что переход Exchange сработает только 3 раза и затем сеть остановится. Поскольку сеть случайным образом выбирает узлы, которым необходимо передать информацию, при каждом срабатывании сети без информации оказываются различные узлы.

Заключение

В ходе работы в системе CPNTools была построена модель Gossip-протокола, который широко используется для распространения информации в современных распределенных системах. Данная модель была модифицирована до модели Gossip-протокола со временем, что позволило проанализировать модель по количеству затрачиваемого времени. Была сделана еще одна модификация модели, в которой вводились некоторые ограничения относительно количества передаваемых сообщений в системе. На примере нескольких топологий коммутируемых сетей построенная модель была проанализирована по многим параметрам, все результаты и анализ приведены в работе.

Ссылки

1. Ломазова И. А. Вложенные сети Петри: моделирование и анализ распределенных систем с объектной структурой. М.: Научный Мир, 2004. 208 с.

2. Dworzanski L., Lomazova I. CPN tools-assisted simulation and verification of nested Petri nets //Automatic Control and Computer Sciences. 2013. Volume 47. Issue 7. P. 393–402.

3. Chaly D., Sokolov V. An Extensible Coloured Petri Net Model of a Transport Protocol for Packet Switched Networks // Lecture Notes in Computer Science. 2003. Volume 2763. P. 66–75.

4. Чалый Д. Ю. Моделирование и анализ сетевых транспортных протоколов с помощью раскрашенных сетей Петри: дисс. …канд. физ.-мат. наук. Ярославль, 2006. 148 с.

5. Башкин В. А. Функциональное программирование на языке SML: методические указания. Ярославль: ЯрГУ, 2007. 39 с.

6. Analysis of two authorization protocols using Colored Petri Nets / Y. Seifi, S. Suriadi, E. Foo, C. Boyd // International Journal of Information Security. 2014. Volume 14. Issue 3. P. 221–247.

7. Einafshar A., Razavi B., Sassani F. Integrated Reconfiguration of Multi-Satellite Network Communication Using Colored Petri Nets // Integrated Systems: Innovations and Applications. 2015. P. 3–28.

8. Ebadi T., Purvis M., Purvis M. A Colored Petri Net Model to Represent the Interactions between a Set of Cooperative Agents // Lecture Notes in Computer Science. 2012. Volume 6573. P. 141–152.

9. Дмитриев В. Н., Тушнов А. С., Сергеева Е. В. Имитационное моделирование системы мониторинга многозвенной сети передачи данных // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2013. № 2. С. 86–91.

УДК 519.681.5: 519.682

Тестирование библиотеки RPMLib на примере задачи о распределении тепла

С. С. Алексеев, В. В. Васильчиков

Ярославский государственный университет им. П. Г. Демидова
E-mail: gwblaize @gmail.com, vasilch@uniyar.ac.ru

В статье описывается параллельный алгоритм решения задачи о распределении тепла, ориентированный на использование библиотеки рекурсивно-параллельного программирования RPMLib, приводятся результаты численного эксперимента.

Ключевые слова: параллельные вычисления, рекурсия, .NET.

Введение. Цели исследования

В данной работе описывается параллельный алгоритм решения задачи о распределении тепла, ориентированный на использование библиотеки рекурсивно-параллельного (РП) программирования RPMLib, а также результаты его тестирования с использованием ее текущей версии [1, 2]. Устройство и основные возможности библиотеки RPMLib описаны в [3, 4], там же рассмотрены результаты ее тестирования на примере задачи о клике.

Эти результаты наглядно продемонстрировали возможность достижения очень значительного ускорения при работе в параллельном режиме за счет использования предлагаемых компонентов.

Однако следует отметить, что задача о клике практически идеально соответствовала концепции рекурсивно-параллельного программирования, чем отчасти и объясняются такие хорошие результаты. Особенно ценным в этом плане было следующее.

• Большая трудоемкость параллельных ветвей программы при весьма небольших объемах данных, передаваемых между модулями. С неравномерным распределением этой трудоемкости отлично справлялся механизм динамической балансировки загрузки процессорных модулей (ПМ).

• Применение метода ветвей и границ в параллельном режиме позволило отсекать заведомо неперспективные ветви в значительно большем количестве, чем при последовательном исполнении, что дополнительно ускоряло работу.

• Задача позволяла эффективно организовать именно рекурсивное разбиение работы, что значительно сокращало временные издержки на ее распределение по системе и сбор результатов.

Не все задачи, допускающие распараллеливание, обладают таким набором достоинств. Интересным представляется исследование самых разных задач и, возможно, даже проведение некоторой классификации с целью выявления типов задач, хорошо вписывающихся в концепцию РП-программирования.

В качестве первого примера для такого исследования решено было взять задачу о распределении тепла в прямоугольной пластине. С одной стороны, эта задача чуть ли не чаще всех используется для демонстрации возможностей параллельных вычислений, с другой — она является своего рода антиподом задачи о клике в плане применимости концепции РП-программирования. Она не имеет ни одного достоинства из перечисленных выше, зато имеет множество недостатков, которые могут уничтожить потенциальную выгоду от параллельного исполнения.

• Малый объем собственно вычислений, приходящихся на единицу передаваемых данных. В параллельных системах, не имеющих общедоступной прямоадресуемой памяти, это самая главная проблема.

• Как следствие предыдущего недостатка, необходимость жесткой привязки вычислений к отдельным ПМ, а значит исключение миграции параллельных ветвей для балансировки загрузки.

• Вследствие сказанного процесс начального распределения работы приходится организовывать не путем рекурсивного деления, а централизованно, путем рассылки с главного процессорного модуля.

• Необходимость жесткой синхронизации вычислений в соседних ПМ, что приводит к дополнительным задержкам. Да и сама схема синхронизации оказалась не так проста, как могло показаться.

Все перечисленное с самого начала работы не внушало особенного оптимизма в плане достижения хорошего (да и вообще какого-нибудь) ускорения за счет параллельного исполнения. Однако авторам хотелось, во-первых, протестировать работу библиотеки RPMLib в «стрессовых» условиях, во-вторых, получить хотя бы грубую оценку отношения объема вычислений к передаваемым данным, достаточного для получения выгоды от параллельности.

Задача о распределении тепла

Напомним вкратце постановку задачи. Процесс распространения тепла в пластине описывается следующим уравнением:

$$\frac{\partial u(x,y,z)}{\partial t} = \frac{\partial^2 u(x,y,z)}{\partial^2 x} + \frac{\partial^2 u(x,y,z)}{\partial^2 x} \qquad (1)$$

Рассмотрим этот процесс на прямоугольной пластине размером $sizeX * sizeY$. Граничные и начальные условия можно задать произвольно.

Для перехода к соответствующему разностному уравнению вводится пространственно-временная сетка с координатами $\Delta x = \Delta y$ и с фиксированным шагом по времени Δt. Теперь решать задачу мы будем численно, разбив область решения равномерной сеткой и заменив производные конечными разностями.

Введем следующие обозначения для узлов сетки: $x_i = i\Delta x$, $y_j = j\Delta y$, $t_k = k\Delta t$, $u_{i,j,k} = u(x_i, y_j, t_k)$. Теперь, воспользовавшись явной разностной схемой, описанной в книге А. А. Самарского [5], с учетом равенства $\Delta t/(\Delta x^2) = \Delta t/(\Delta y^2)$, имеем следующее уравнение для нахождения значений функции $u(x_i, y_j, t_k)$ в узлах сетки в каждый очередной времени:

$$u_{i,j,k+1} = (u_{i-1,j,k} + u_{i+1,j,k} + u_{i,j-1,k} + u_{i,j+1,k} - 4u_{i,j,k}) \times \left(\frac{\Delta t}{\Delta x^2}\right) + u_{i,j,k} \qquad (2)$$

Мы не рассматриваем здесь более подробно сопутствующие математические вопросы, например вопрос о сходимости, поскольку для нас представляет интерес только возможность эффективного распараллеливания вычислений итерационного процесса (2) с использованием библиотеки RPMLib.

Алгоритм решения задачи

Опишем основные моменты построения параллельного алгоритма решения поставленной задачи. Очевидно, что самый естественный способ разбиения вычислений — это разрезать всю пластину размера $\boldsymbol{sizeX} * \boldsymbol{sizeY}$ на прямоугольные полоски и поручить каждому модулю циклические (по времени) вычисления на каждой полоске. Поэтому было изначально решено закрепить каждую такую полоску за конкретным ПМ, причем в таком варианте нет даже необходимости устраивать рекурсивное деление работы, просто достаточно назначить каждому модулю свою подзадачу. Для этой цели мы воспользовались методом P_Send(), предоставляемым нашей библиотекой.

Основная проблема состояла в том, как организовать корректную и по возможности ненакладную синхронизацию по завершении очередного временного шага. В этот момент «соседние» ПМ должны обменяться вычисленными значениями на границах полосок и начать очередную итерацию цикла по времени. Для решения этой задачи на каждом модуле мы завели такой набор данных.

• Два массива для отправки левой границы и столько же для правой границы. Один из массивов используется в текущих вычислениях, другой содержит ранее вычисленные значения и предназначен для отправки соседу.

• Такие же две пары массивов для получения границ, вычисленных соседями.

• Массивы событий (тип ManualResetEventSlim) из расчета по одному на каждый из перечисленных массивов. Эти события используются для синхронизации и не позволяют отправлять еще не готовые данные или принимать данные во все еще используемый массив.

Для приема и отправки данных на каждом модуле запускаются отдельные фоновые потоки вычислений. Кроме того, все действия по сетевому взаимодействию также осуществля-

ются в отдельных потоках, впрочем, это уже забота библиотеки RPMLib.

Итерация основного цикла (по времени) выглядит таким образом.

• Вычисляем границы.

• Устанавливаем признак готовности к отправке.

• Производим вычисления на оставшейся части полоски (середина).

• Ждем, пока соседи пришлют вычисленные ими границы для данной итерации.

Фоновые потоки для отправки границ в бесконечном цикле ждут установки признака готовности данных к отправке и далее отправляют их по назначению

Прием границ, вычисленных соседями, происходит в потоках, запущенных библиотекой RPMLib, по получении данных основной поток вычислений информируется об этом с использованием механизма событий.

Результаты тестирования. Выводы

В процессе тестирования использовались компьютеры на базе четырехъядерного процессора Intel Core i3 с тактовой частотой 3.07 GHz и 4 GB оперативной памяти, работающие под управлением 64-разрядной ОС Windows 7 с.NET Framework 4.0. Пропускная способность сети 100 Mb/s.

Описанный ранее алгоритм был реализован на языке C# с использованием библиотек [1; 2]. Приняв за единицу пространственного измерения $\Delta \boldsymbol{x} = \Delta \boldsymbol{y}$, мы провели моделирование процесса распространения тепла для пластин разного размера $\boldsymbol{sizeX} * \boldsymbol{sizeY}$, а именно: $\mathbf{1000000} * \mathbf{10}$, $\mathbf{100000} * \mathbf{100}$, $\mathbf{10000} * \mathbf{1000}$ и $\mathbf{1000} * \mathbf{10000}$. Здесь во всех случаях объем собственно вычисления на временном слое был одинаков, различался только объем передаваемой между параллельно работавшими компьютерами информации о границах, причем между первым и последним из перечисленных вариантов эта разница была тысячекратной.

Во всех случаях ускорения за счет параллельного решения задачи достигнуто не было, более того, было обнаружено значительное замедление. Этот факт не был неожиданным и объяснялся недостаточным объемом вычислений, приходящихся на одну операцию передачи по сети. Поскольку одной из основ-

ных целей исследования являлось экспериментальное определение этого отношения, достаточного для получения выгоды от параллельности, было решено искусственно увеличить объем вычислений за счет их повтора. Было отмечено, что уже десятикратное увеличение вполне достаточно, чтобы получить при работе на 4 компьютерах ускорение примерно в полтора раза. Эксперимент с количеством повторов, равным 25, продемонстрировал еще большее ускорение.

В табл. 1 приводятся данные о времени выполнения одного шага во времени в зависимости от количества задействованных рабочих станций. Результаты приведены для задачи размером $\mathbf{10000 * 1000}$. Результаты экспериментов для пластин других размеров не приводятся, поскольку очень похожи на данные, приведенные здесь. Поэтому был сделан вывод, что объем пересылаемых данных (а он, как мы отметили ранее, изменялся тысячекратно) влияет на скорость работы куда меньше, чем количество операций передачи данных по сети. Разумеется, если этот объем очень велик, такое заявление с нашей стороны было бы, скорее всего, неверным, однако в пределах 80 килобайт (максимальный объем на одну пересылку в наших экспериментах) это утверждение представляется справедливым.

Таблица 1

Длительность вычислений на одном временном слое (в мс)
Пластина размера 10000 * 1000

Кол-во ПМ	1 повтор	10 повторов	25 повторов
1	442.2	4320	10823.8
2	1840.7	2869.4	5829
4	3813.7	3760.4	3991.7
8	4559.9	4560.3	4579.3

На рис. 1 представлены данные о достигнутом ускорении.

Можно также отметить снижение ускорения на 8 ПМ. Оно объясняется, во-первых, уменьшением отношения "объем вычислений / количество пересылок", во-вторых, замедлениями, вызванными необходимостью перед продолжением вычислений дождаться результатов от соседних модулей.

В целом, можно сделать вывод, что основные цели исследования были достигнуты, а именно:

• продемонстрирована работоспособность библиотеки RPMLib даже в случае решения задач, очень плохо вписывающихся в концепцию рекурсивного распараллеливания;

• установлено, что объем данных, приходящихся на одну операцию передачи по сети, практически не сказывается на времени работы;

• эмпирически удалось определить достаточное для параллельных вычислений значение отношения «объем вычислений / количество пересылок».

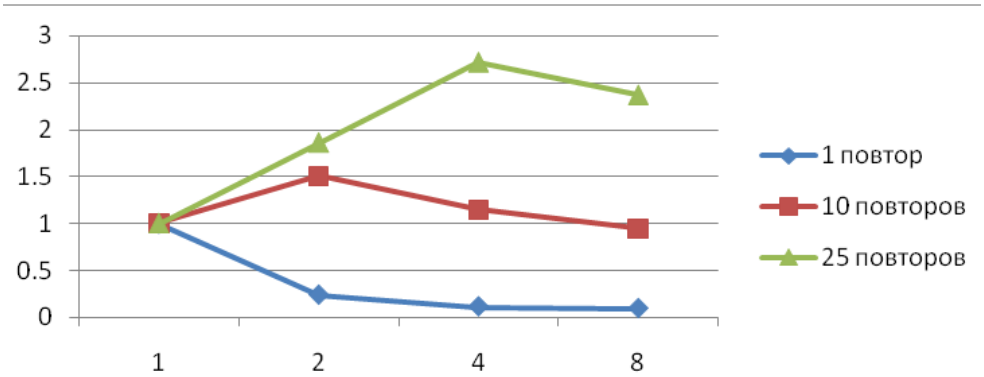


Рис. 1. Зависимость ускорения от количества компьютеров

Ссылки

1. Васильчиков В. В. Коммуникационный модуль для организации полносвязного соединения компьютеров в локальной сети с использованием .NET Framework. // Свидетельство о государственной регистрации программы для ЭВМ № 2013619925. 2013.

2. Васильчиков В. В. Библиотека поддержки рекурсивно-параллельного программирования для .NET Framework. // Свидетельство о государственной регистрации программы для ЭВМ № 2013619926. 2013.

3. Васильчиков В. В. О поддержке рекурсивно-параллельного программирования в .NET Framework. // Моделирование и анализ информационных систем. Т. 21. № 2. Ярославль: ЯрГУ, 2014. С. 15–25.

4. Vasilchikov V. V. On the Recursive-Parallel Programming for the .NET Framework // Automatic Control and Computer Sciences, 2014. Vol. 48. No. 7. P. 636–641.

5. Самарский А. А. Теория разностных схем. М.: Наука, 1977. 656 с.

УДК 541.1

Кратные волны в кольце из пороговых нейронов

С. Е. Ануфриенко, Н. Ю. Волкова

Ярославский государственный университет им. П. Г. Демидова
E-mail sanufienko@rambler.ru, newstilllife@yandex.ru

В статье рассматривается модель порогового нейрона, основанная на дифференциальном уравнении с запаздыванием. Исследуется кольцевая структура из пороговых нейронов; вопрос о минимальном размере кольца из диффузионно связанных пороговых нейронов, в котором может побежать кратная волна из активных нейронов.

Ключевые слова: пороговый нейрон, кольцевые нейронные структуры, кратная волна, дифференциальное уравнение с запаздыванием.

Введение

Нейрон — это структурно-функциональная единица нервной системы. Эта клетка имеет сложное строение, высокоспециализирована и в структуре содержит ядро, тело клетки и отростки. В организме человека насчитывается более ста миллиардов нейронов. Сложность и многообразие функций нервной системы определяются взаимодействием между нейронами, которое, в свою очередь, представляет собой набор различных сигналов, передаваемых в рамках взаимодействия нейронов с другими нейронами или мышцами и железами. Сигналы испускаются и распространяются с помощью ионов, генерирующих электрический заряд, который движется по телу нейрона.

Наличие в мозге замкнутых структур обнаружено уже давно. Кольцевые образования играют важную роль в механизмах памяти.

В этой статье рассматривается кольцо из пороговых нейронов и описан периодический режим, называемый кратной волной, решен вопрос о минимальном размере кольца.

Модель порогового нейрона

Под влиянием внутренних или внешних процессов мембранный потенциал медленно растет, и по достижении им некоторого значения, называемого пороговым, нейрон генерирует кратковременный высокоамплитудный импульс (спайк). Во время спайка мембранный потенциал быстро растет и затем быстро падает, а его значение становится меньше, чем вначале. После спайка мембрана гиперполяризована, нейрон находится в рефрактерном состоянии. Постепенно мембранный потенциал восстанавливает нормальное значение.

У некоторых нейронов после выхода из состояния гиперполяризации мембранный потенциал достигает некоторого значения (потенциал покоя), меньшего, чем пороговое. Для того чтобы нейрон сгенерировал новый спайк, требуется внешнее воздействие. Такие нейроны называются пороговыми. Амплитуда спайка не зависит от силы воздействия, если сигнала достаточно для возбуждения нейрона.

Величину мембранного потенциала ($u(t)$), будем отсчитывать от уровня максимальной гиперполяризации. Уравнение, описывающее динамику $u(t) \geq 0$, имеет вид [10]:

$$\dot{u} = \lambda\left[-1 - f_{Na}(u) + f_K\big(u(t-1)\big)\right]u + \varepsilon \qquad (1)$$

Параметры:

$\lambda \gg 1$ — отражает высокую скорость протекания электрических процессов;

$0 < \varepsilon \ll 1$ — учитывает токи утечки, проходящие через мембрану порогового нейрона;

функции $f_{Na}(u)$ и $f_K(u)$ — достаточно гладкие, монотонно убывают к нулю при $u \to \infty$ быстрее, чем $O(u^{-1})$ (они описывают поведение натриевых и калиевых каналов).

Введем обозначения:

$$\alpha = 1 + f_{Na}(0) - f_K(0) > 0,$$
$$\alpha_1 = f_K(0) - 1 > 1,$$
$$\alpha_2 = f_{Na}(0) + 1 > \alpha_1.$$

Число $f_K(0) - f_{Na}(1) - 1 > 0$ связано с пороговым значением: будем считать, что спайк нейрона начинается в момент времениt_s, такой, что

$$u(t_s) = 1 \text{ и } u(t) < 1 \text{ при } t_s - 1 < t < t_s.$$

Правая часть уравнения (1) при $u = 0$ положительна, а при достаточно больших u отрицательна, значит уравнение (1) имеет положение равновесия [8]

$$u = u_* \approx \frac{\varepsilon}{\lambda\alpha} \ll 1$$

Доказательство устойчивости положения равновесия и поведение нейрона на различных промежутках времени описано в учебном пособии [9].

Формулы, описывающие динамику мембранного потенциала порогового нейрона, имеют следующий вид:

$$u(t) = \begin{cases} e^{\lambda\alpha_1(t+o(1))} & \text{при } t \in [\delta, 1-\delta] \\ e^{\lambda(\alpha_1-(t-1)+o(1))} & \text{при } t \in [1+\delta, 1+\alpha_1-\delta] \\ \dfrac{\varepsilon + o(1)}{\lambda\alpha_2} & \text{при } t \in [1+\alpha_1+\delta, 2+\alpha_1-\delta] \\ \dfrac{\varepsilon + o(1)}{\lambda\alpha} & \text{при } t \geq 2+\alpha_1+\delta \end{cases}$$

Колебание в системе пороговых нейронов

Рассмотрим кольцо из N одинаковых диффузионно связанных пороговых нейронов. Каждый нейрон связан с предыдущим и следующим за ним нейронами. Пусть $u_i(t) \geq 0$ — мембранные потенциалы ($i = 1, \dots, N$).

Описание динамики сети системой дифференциальных уравнений, обозначим ее (2):

$$\dot{u}_1 = \lambda\left[-1 - f_{Na}(u_1) + f_K\left(u_1(t-1)\right)\right]u_1 + \varepsilon + e^{-\lambda\sigma}(u_N - u_2 - 2u_1);$$

$$\dot{u}_i = \lambda\left[-1 - f_{Na}(u_i) + f_K\left(u_i(t-1)\right)\right]u_i + \varepsilon + e^{-\lambda\sigma}(u_{i-1} - u_{i+1} - 2u_i), \qquad i = 2, \dots, N-1;$$

$$\dot{u}_N = \lambda\left[-1 - f_{Na}(u_N) + f_K\left(u_N(t-1)\right)\right]u_N + \varepsilon + e^{-\lambda\sigma}(u_{N-1} - u_1 - 2u_N).$$

Параметры $\lambda, \varepsilon, f_{Na}(u)$ и $f_K(u)$ имеют тот же смысл.

Смысл параметра $\sigma (0 < \sigma < \alpha_1)$ заключается в следующем: $e^{-\lambda\sigma}$ подавляет слабые сигналы.

Состояние равновесия системы уравнений $u_i = u_* \approx \frac{\varepsilon}{\lambda\alpha}$ экспоненциально устойчиво.

Волны в кольце из N диффузионно связанных пороговых нейронов

Начальные условия:

$u_1(s) = \varphi_1(s),\ s \in [-1, 0].$

$\varphi_1(s)$ — непрерывные на $s \in [-1,0]$ функции для первого нейрона, которые удовлетворяют следующим условиям:

$$\varphi_1(0) = 1,\ 0 \le \varphi_1(s) \le max\left(e^{\lambda\alpha s/2}, \frac{1}{\lambda}\right).$$

В момент времени $\mathrm{t} = 0$ начинается спайк первого нейрона. Для остальных нейронов, будем считать, что

$$u_i(s) = u_* \approx \frac{\varepsilon}{\lambda\alpha} \text{ при } s \in [-1,0].$$

Динамика мембранного потенциала первого нейрона (при отсутствии внешнего воздействия) описывается следующей системой:

$$u_1(\mathrm{t}) = \begin{cases} \mathrm{e}^{\lambda\alpha_1(t+o(1))} & t \in [\delta, 1-\delta] \\ \mathrm{e}^{\lambda(\alpha_1-(t-1)+o(1))} & t \in [1+\delta, 1+\alpha_1-\delta] \\ \frac{\varepsilon+o(1)}{\lambda\alpha_2} & t \in [1+\alpha_1+\delta, 2+\alpha_1-\delta] \\ \frac{\varepsilon+o(1)}{\lambda\alpha} & t \ge 2+\alpha_1+\delta \end{cases} \quad (3)$$

$o(1)$ — слагаемые, которые стремятся к нулю при$\lambda \to \infty$.

Исследования по волне в кольце из N диффузионно связанных пороговых нейронов можно подробно посмотреть в [9].

Динамика мембранного потенциала нейронов:

$u_i(t) \approx u_1\big(t - \tau(i-1)\big)$

(с точностью до $o(1)$) при $t > \tau(i-1),\ i = 3, \dots, N$

Результаты исследования показывают, что во время спайка и на интервале времени длины единица после него нейрон невосприимчив к воздействию извне. Продолжительность этого периода:

$$\mathrm{T}_{\text{нев.}} = 2 + \alpha_1 + o(1)$$

Для того чтобы волна не затухала, к моменту начала спайка N-го нейрона первый должен выйти из состояния невосприимчивости:

$$(N-1)\tau > \alpha_1 + 2,\ \tau = \frac{\sigma}{\alpha_1}.$$

Отсюда получим, что:

$$\sigma > \frac{\alpha_1(\alpha_1+2)}{N-1}.$$

Из того, что

$$\sigma > \frac{\alpha_1(\alpha_1+2)}{N-1} \text{ и } \sigma < \alpha_1$$

легко видим, что:

$$\frac{\alpha_1(\alpha_1+2)}{N-1} < \alpha_1,$$

$$\alpha_1\big(\alpha_1 + (3-N)\big) < 0,$$

$$0 < \alpha_1 < \frac{N-3}{1} \text{ и } \alpha_1 > 1.$$

Значит, данное неравенство имеет решение при $N-3>1$, т. е. $N>4$.

То есть для того, чтобы к моменту начала спайка каждого нейрона следующий за ним вышел из состояния невосприимчивости и по кольцу побежала волна из активных нейронов, нам необходимо кольцо минимум из пяти диффузионно связанных пороговых нейронов.

Кратная волна в кольце из пороговых нейронов

Если число нейронов в кольце велико, то существуют более сложные режимы взаимодействия нейронов между собой, например кратные волны.

Для начала можно рассмотреть двукратную волну. Пусть в нулевой момент времени начался спайк у первого нейрона. Предположим, что по каким-либо причинам в тот момент времени, когда первая волна еще не дошла до последнего нейрона, начался новый спайк первого нейрона. В результате по кольцевой структуре вслед за первой побежит вторичная волна возбуждения. Такой режим и называется двукратной волной.

Есть период времени, в течение которого нейрон невосприимчив к воздействию:

$$\mathrm{T}_{\text{нев.}} = 2 + \alpha_1 + o(1)$$

Для того чтобы волна не затухала, к моменту начала спайка нейрона первый должен выйти из состояния невосприимчивости:

$$(N-1)\tau > 2(\alpha_1+2),\ \tau = \frac{\sigma}{\alpha_1}.$$

Отсюда получим, что:

$$\sigma > \frac{2\alpha_1(\alpha_1+2)}{N-1}.$$

Из того, что

$$\sigma > \frac{2\alpha_1(\alpha_1+2)}{N-1} \text{ и } \sigma < \alpha_1$$

получим

$$\frac{2\alpha_1(\alpha_1+2)}{N-1} < \alpha_1,$$
$$\alpha_1\big(2\alpha_1 + (5-N)\big) < 0,$$
$$0 < \alpha_1 < \frac{N-5}{2} \text{ и } \alpha_1 > 1.$$

Значит, данное неравенство имеет решение и $\frac{N-5}{2} > 1$, т. е. $N > 7$.

Получили, что кольцо из восьми диффузионно связанных пороговых нейронов — минимальное кольцо, в котором может побежать двойная волна из активных нейронов.

Вследствие внешнего воздействия первый нейрон начнет спайк, и пойдет первая волна. Первый нейрон выйдет из состояния невосприимчивости, когда пятый начнет свой спайк. Воздействуем на первый нейрон так, чтобы он начал свой спайк одновременно с пятым. Тогда к моменту начала спайка каждого нейрона следующий за ним выйдет из состояния невосприимчивости. Следовательно, по кольцу побежит двукратная волна.

Если кольцо будет большей размерности, мы сможем пустить k-кратную волну. Сначала запускаем первую волну, каждая следующая волна сможет пойти, когда первый нейрон будет выходить из состояния невосприимчивости. То есть, получим:

$$(N-1)\tau > k(\alpha_1 + 2),\ \tau = \frac{\sigma}{\alpha_1}.$$

Отсюда видим, что:

$$\sigma > \frac{k\alpha_1(\alpha_1+2)}{N-1},$$

Из того, что

$$\sigma < \alpha_1 \text{ и} \sigma > \frac{k\alpha_1(\alpha_1+2)}{N-1}$$

получим:

$$\frac{k\alpha_1(\alpha_1+2)}{N-1} < \alpha_1,$$
$$\alpha_1\big(k\alpha_1 + (2k+1-N)\big) < 0,$$
$$0 < \alpha_1 < \frac{-2k-1+N}{k} \text{ и } \alpha_1 > 1.$$

Значит, данное неравенство имеет решение при$\frac{-2k-1+N}{k} > 1$, т. е. $N > 3k+1$.

В случае с диффузионно связанными пороговыми нейронами для k-кратной волны необходимо кольцо минимум из $3k + 2$ нейронов.

Ссылки

1. Ходоров Б. И. Общая физиология возбудимых мембран. М.: Наука, 1975. 406 с.

2. Hodgkin A. L., Huxly A. F. Action potentials recorded from inside a nerve fiber // Nature. 1939. Vol. 144. P. 710–711.

3. Ходжкин А. Л. Нервный импульс. М.: Мир, 1965. 125 с.

4. Катц Б. Нерв, мышца, синапс. М.: Мир, 1968. 248 с.

5. Кол К. С. Нервный импульс (теория и эксперимент) // Теоретическая и математическая биология. М.: Мир, 1968. С. 154–193.

6. Блюм Ф., Лейзерсон А., Хофстефер Л. Мозг, разум и поведение. М.: Мир, 1988. 191 с.

7. От нейрона к мозгу / Дж. Г. Николлс, А. Р. Мартин, Б. Дж. Валлс, П. А. Фукс. М.: УРСС, 2003.

8. Кащенко С. А., Майоров В. В. Модели волновой памяти. М.: ЛИБРОКОМ, 2009. 288 с.

9. Майоров В. В., Ануфриенко С. Е. Импульсные нейросети: учебное пособие. Ярославль: ЯрГУ, 2006. 98 с.

10. Майоров В. В., Мышкин И. Ю. Математическое моделирование нейронов сети на основе уравнений с запаздыванием // Математическое моделирование. 1990. Т. 2. № 11. С. 64–76.

УДК 519

Выделение контуров на изображениях в ОС Android

Н. П. Баранов, А. А. Сивов

Ярославский государственный университет им. П. Г. Демидова
E-mail: baranov.nikita@gmail.com, mm05@mail.ru

В данной статье рассматриваются три алгоритма для выделения контуров в изображении. Цель работы — создание приложения для операционной системы Android, которое позволяет выделить контуры в изображении тремя алгоритмами с последующим сохранением полученного результата. Приложение было разработано и протестировано на ряде устройств.

Ключевые слова: Android, разработка приложения, эмулятор Android, обработка изображения, выделение контуров.

Введение

Выделение границ (выделение краев) — термин в теории обработки изображения и компьютерного зрения, частично из области поиска объектов и выделения объектов, — основывается на алгоритмах, позволяющих выделять точки цифрового изображения, в которых резко изменяется яркость или есть другие виды неоднородностей.

Основной целью обнаружения резких изменений яркости изображения является фиксация важных событий и изменений мира.

В идеальном случае результатом выделения границ является набор связанных кривых, обозначающих границы объектов, граней и оттисков на поверхности, а также кривые, которые отображают изменения положения поверхностей. Таким образом, применение фильтра выделения границ к изображению может существенно уменьшить количество обрабатываемых данных, из-за того что отфильтрованная часть изображения считается менее значимой, а наиболее важные структурные свойства изображения сохраняются.

Мобильные телефоны давно перестали быть чем-то необычным и великолепно справляются со своей функцией — являются средством коммуникации между людьми. При этом недавно появившиеся, но уже прочно вошедшие в нашу жизнь смартфоны настолько функциональны, что трудно сказать, чего они не умеют: это и плеер, и фотоаппарат, и возможность использования интернет-ресурсов, и прочее. По сути, все смартфоны стали небольшой копией компьютера, который постоянно можно иметь при себе.

В наше время все больше и больше смартфонов, коммуникаторов, планшетных ПК и других видов устройств, удобных для использования как в повседневной жизни, так и, например, в заграничных поездках выпускаются на базе ОС Android. Каковы же причины распространения данной операционной системы? Во-первых, Android поддерживает большое количество устройств разных производителей. Во-вторых, Android характеризуется высокой доступностью средств разработки. Средства разработки для платформы Android бесплатны, в то время как разработка, к примеру, под iPhone (от компании Apple) требует немалых начальных финансовых вложений. Кроме всего перечисленного, преимуществом ОС Android является наличие бесплатных библиотек для работы со сторонними ресурсами (YandexMapKit, GoogleMap API, др.), в то время как для WindowsPhoneMobile такие библиотеки не распространены [1].

Разработка программного обеспечения

Приложения для Android являются программами в нестандартном байт-коде для виртуальной машины Dalvik. DalvikVirtualMachine — основанная на регистрах виртуальная машина, она оптимизирована для низкого потребления памяти. Программы для Dalvik пишутся на языке Java. Несмотря на это, стандартный байт-код Java не используется, вместо него Dalvik VM исполняет байткод собственного формата. После компиляции исходных текстов программы на Java [2] (при помощи javac) утилита dx из «Android SDK» преобразует .class файлы в формат .dex, пригодный для интерпретации в Dalvik. В версии Android 1.5 разработчики добавили NativeDevelopmentKit, который позволяет писать собственные низкоуровневые модули для системы на языке C/C++, опираясь на стандартные linux-библиотеки.

В KitKat у виртуальной машины Dalvik появился конкурент в виде ART.ART (аббревиатура термина «Android Runtime») — это новая среда выполнения приложений, написанная на C/C++, которая отличается от существующей в Android виртуальной машины Dalvik тем, что все приложения в системе уже скомпилированы, а значит потребность в JIT-компиляторе отпадает. Таким образом, ART позволяет запускать приложения на разном оборудовании без предварительной адаптации со стороны разработчиков. Помимо этого, на запуск приложений в новых условиях уходит в два раза меньше времени. В ART есть и недостатки, один из которых связан с принципом работы в условиях ART. Данная среда приводит к тому, что вся необходимая информация переводится в машинно-ориентированный язык еще во время установки приложений (AOT компиляция), а это требует дополнительного времени, из-за чего весь процесс установки сильно растягивается, а приложения занимают больше места, т. к. все время скомпилированы.

Оператор Собеля

Оператор Собеля [3] основан на свертке изображения небольшими сепарабельными целочисленными фильтрами в вертикальном и горизонтальном направлениях, поэтому его относительно легко вычислять. С другой стороны, используемая им аппроксимация градиента достаточно груба, особенно это сказывается на высокочастотных колебаниях изображения.

Оператор вычисляет градиент яркости изображения в каждой точке. Так находится направление наибольшего увеличения яркости и величина ее изменения в этом направлении. Результат показывает, насколько «резко» или «плавно» меняется яркость изображения в каждой точке, а значит вероятность нахождения точки на грани и ориентацию границы. На практике вычисление величины изменения яркости (вероятности принадлежности к грани) надежнее и проще в интерпретации, чем расчет направления.

Строго говоря, оператор использует ядра 3×3, с которыми сворачивают исходное изображение для вычисления приближенных значений производных по горизонтали и по вертикали. Пусть $\mathbf{A}$ — исходное изображение, а $\mathbf{G}_x$ и $\mathbf{G}_y$ — два изображения, где каждая точка содержит приближенные производные по x и по y. Они вычисляются следующим образом:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A},$$

где * обозначает двумерную операцию свертки.

Координата x здесь возрастает «направо», а y — «вниз». В каждой точке изображения приближенное значение величины градиента можно вычислить, используя полученные приближенные значения производных:

$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$ (имеется в виду поэлементно)

Оператор Кэнни

Оператор обнаружения границ изображения. Был разработан в 1986 г. Джоном Кэнни и использует многоступенчатый алгоритм для обнаружения широкого спектра границ в изображениях [4].

Кэнни изучил математическую проблему получения фильтра, оптимального по критериям выделения, локализации и минимизации нескольких откликов одного края. Он показал, что искомый фильтр является суммой четырех экспонент. Он также показал, что этот фильтр может быть хорошо приближен первой производной Гауссианы. Кэнни ввел понятие подавления не-максимумов, которое означает, что пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента.

Хотя его работа была проведена на заре компьютерного зрения, детектор границ Кэнни до сих пор является одним из лучших. Кроме особенных частных случаев, трудно найти детектор, который бы работал существенно лучше, чем детектор Кэнни.

Алгоритм состоит из пяти отдельных шагов.

1. **Сглаживание**. Размытие изображения для удаления шума.

2. **Поиск градиентов**. Границы отмечаются там, где градиент изображения приобретает максимальное значение.

3. **Подавление не-максимумов**. Только локальные максимумы отмечаются как границы.

4. **Двойная пороговая фильтрация**. Потенциальные границы определяются порогами.

5. **Трассировка области неоднозначности**. Итоговые границы определяются путем подавления всех краев, не связанных с определенными (сильными) границами.

Перед применением детектора обычно преобразуют изображение в оттенки серого, чтобы уменьшить вычислительные затраты. Этот этап характерен для многих методов обработки изображений.

Дискретный оператор Лапласа

Фильтр используется для подсветки (выделения) границ и повышения резкости [5]. Его применение подчеркивает разрывы уровней яркостей на изображении и подавляет области со слабыми изменениями яркостей.

Суть данного преобразования состоит в построении маски фильтра и его наложении на изображение. Маска представляет собой двумерный массив (чаще всего это 3x3), значение элементов которого определяет итог преобразования. В каждой точке изображения отклик фильтра вычисляется как сумма произведений элементов маски фильтра на соответствующие значения пикселей в области под маской.

Для **преобразования Лапласа** построение маски фильтра происходит с применением второй производной.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Преобразование действует на перепады яркости (например, в виде ступенчатого перепада) во всех направлениях, а также выделяет изолированные точки, тонкие линии и острые углы объектов.

Заключение

В результате выполнения выпускной квалификационной работы было создано программное обеспечение для ОС Android, решающее поставленную задачу. В приложении реализованы основные методы выделения контуров: операторы Собеля, Кэнни и Лапласа. В ходе решения этой задачи были получены собственные реализации указанных операторов, которые могут быть применены в ситуациях, когда не доступны такие библиотеки, как OpenCV [6], а также могут служить кодовой базой для модификации методов выделения границ в исследовательских целях. Помимо собственных реализаций, была внедрена поддержка широко используемой в промышленных и научных решениях биб-

лиотеки OpenCV, что позволяет выделять контуры на изображениях с помощью имеющихся в данной библиотеке методов.

Ссылки

1. Медникс З., Дорнин Л. Программирование под Android. СПб.: Питер, 2013. 560 с.
2. Шилдт Г. Java. Полное руководство (Восьмое издание). М.: ВИЛЬЯМС, 2012. 1104 с.
3. Рассел Дж. Оператор Собеля. VSD, 2013. 106 с.
4. Pattern Analysis and Machine Intelligence. 1986. 6 нояб. (№ 8). 699 с.
5. Рассел Дж. Оператор Лапласа. VSD, 2013. 115 с.
6. Библиотека OpenCV. URL: http://opencv.org

УДК 004.921

Создание статических трехмерных моделей с использованием скриптового языка Maxscript

К. Д. Бельская, Н. В. Легков

Ярославский государственный университет им. П. Г. Демидова
E-mail: belskaya_kd@mail.ru, legnick@yandex.ru

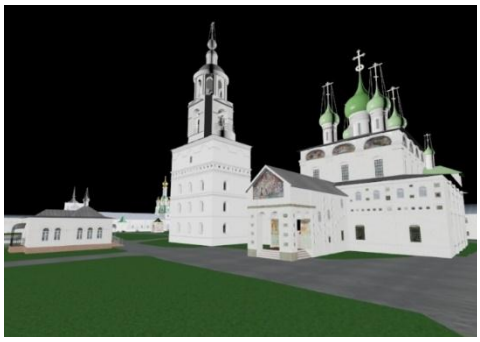
В статье идет речь о моделировании статических трехмерных моделей, описан процесс создания скрипта генерации многогранных объектов, особенности создания объектов средствами языка Maxscript при моделировании статической трехмерной модели Толгского монастыря.

Ключевые слова: трехмерная графика, трехмерное моделирование, Ds MAX, скрипт, Maxscript.

В настоящее время трехмерная графика применяется в различных отраслях и сферах деятельности: ее активно применяют для создания игр и фильмов, в архитектуре и строительстве, в медицине и физике, а также во многих других областях.

Востребованным становится и создание моделей уже построенных зданий. Такие модели можно использовать при реконструкции и ремонте объектов, с их помощью можно создавать демонстрационные фотографии, видеоролики, виртуальные экскурсии.

Рассмотрим более подробно процесс создания трехмерной модели Толгского монастыря в г. Ярославле в программе 3Ds MAX. Перед началом моделирования были выделены следующие этапы: создание моделей храмов и других зданий, моделирование ландшафта, соединение моделей зданий в единый проект.



Для разработки моделей зданий был выбран метод работы с редактируемыми поверхностями типа Editable Poly (Редактируемая полигональная поверхность).

Также для некоторых элементов архитектуры применялось моделирование на основе сплайновых фигур. При этом использовались модификаторы: Surface (Поверхность), Lathe (Вращение вокруг оси), Sweep (Выгнутость), Extrude (Выдавливание) и BeveL (Выдавливание со скосом), а также составной объект Loft (Лофтинг). Большинство объектов моделей являются высокополигональными, поэтому использовался материал Multi/Sub-Object (Многокомпонентный). Этот метод позволяет назначить объекту более одного материала на уровне грани посредством идентификатора материала.

Для моделирования ландшафта были рассмотрены несколько методов: составной объект Terrain, плагин Populate: Terrain, модификатор Displace, Paint Deformation в Edit Poly, модификатор Surface из сплайнов. После исследования этих методов был выбран третий метод (модификатор Displace), т. к. в данном случае он намного эффективней, прост в использовании и настройке.

После создания моделей храмов, других зданий и ландшафта с помощью функции импорта все модели были собраны в единый проект.

В программе 3Ds MAX существует множество методов для разработки трехмерных моделей. При создании модели

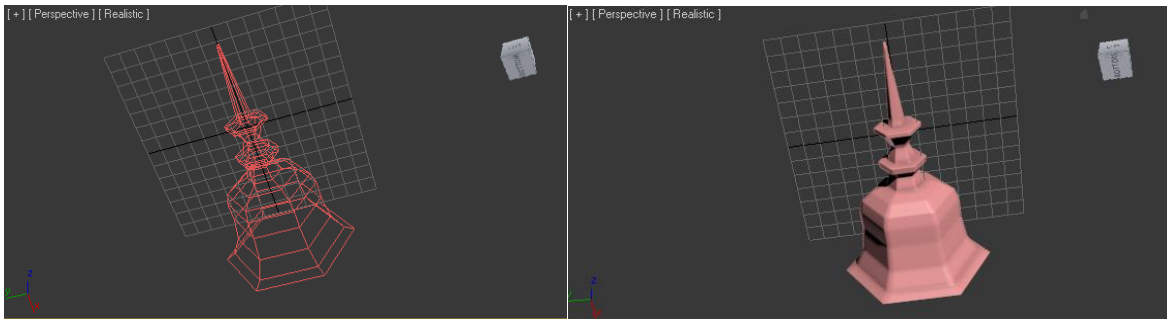
Толгского монастыря стандартного функционала программы порой было недостаточно для решения узконаправленных задач, таких как моделирование множества похожих объектов, возникала необходимость использования при моделировании встроенного языка программирования Maxscript.

Для создания куполов применялось моделирование с помощью сплайнов и с последующим применением модификатора Lathe (Вращение вокруг оси). Такое моделирование позволяет быстро достичь требуемой формы объекта. Но многогранные купола моделировать несколько сложнее, поэтому были рассмотрены следующие методы: Lathe (Вращение вокруг оси), инструмент Extrude в Edite Poly, модификатор Surfase.

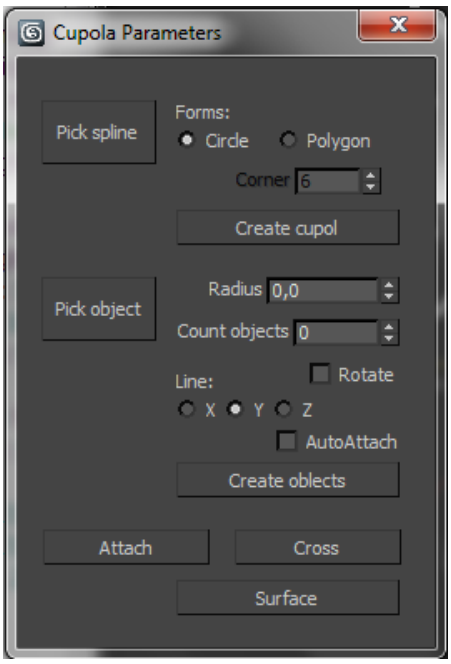
Первый способ позволяет быстро на основе сплайна создавать требуемый объект. Настройка параметров модификатора позволяет создавать многогранные формы, но при этом есть ряд недостатков: на узких участках наблюдается деформация (например, «перекручивание» на шпиле) и некорректно отображаются некоторые грани и полигоны при визуализации, из-за чего искажается форма объекта или даже полностью теряется.

Метод работы с редактируемыми поверхностями (Edite Poly) занимает много времени и является трудоемким, особенно при создании множества похожих объектов. Кроме того, объект, созданный этим способом, выглядит угловато и неестественно.

Последний из рассмотренных методов, моделирование с помощью модификатора Surfase, позволяет добиться лучшего результата. Метод включает в себя два этапа: создание каркаса объекта и применение к нему модификатора Surface. Сложность заключается в создании каркаса объекта.



Для корректного применения модификатора необходимо построить ровный каркас, который создается из сплайнов, что при работе только стандартными средствами является очень долгим и трудоемким процессом. Для оптимизации был создан следующий функционал. Скрипт используется для создания тел вращения и куполов.



На основе нарисованного сплайна создается объект методом вращения (Create cupol), при этом выбирается форма Circle (круг) или Polygon (многоугольник).

В случае выбора многоугольной формы предлагается выбрать количество углов.

Если качество полученного объекта неудовлетворительно, можно создать его другим способом — когда объект создается поэтапно: основные направляющие каркаса, затем они соединяются, и накладывается поверхность. Параметры настройки:

- выбор сплайна (Pick object);
- радиус (Radius);
- количество углов/объектов (Count objects), поворот объектов;
- выбор оси;
- создание заданного количества объектов (Create objects);
- объединение объектов в один сплайн (Attach);
- построение вспомогательных ребер (Cross);
- создание поверхности (Surface).

Стандартные средства 3Ds MAX не всегда обеспечивают выполнение поставленной задачи. Использование скриптов позволяет максимально оптимизировать процесс решения таких задач.

Ссылки

1. Ким Ли. 3D Studio MAX Искусство трехмерной анимации Platinum Edition (+CD). СПб.: Диасофт-ЮП, 2005.

2. Келли Л. Мэрдок Autodesk 3ds Max 2008. Библия пользователя. М.: Вильямс, 2008.

УДК 004

Моделирование организации управления трафиком средствами AnyLogic

Е. Ю. Власова, В. А. Соколов

Ярославский государственный университет им. П. Г. Демидова
E-mail: helenavlasova39@gmail.com, valery-sokolov@yandex.ru

В статье описано построение имитационной модели управления трафиком на участке транспортной сети; разработанный интеллектуальный контроллер, производящий локальное адаптивное управление перекрестком.

Ключевые слова: AnyLogic, имитационное моделирование, дискретно-событийное моделирование, агентное моделирование, модель, дорожное движение, автомобиль, светофор, контроллер, трафик, управление трафиком.

Введение

Транспортная инфраструктура — одна из важнейших инфраструктур, обеспечивающих жизнь городов и регионов. В последние десятилетия во многих крупных городах исчерпаны или близки к исчерпанию возможности экстенсивного развития транспортных сетей. Решение таких задач невозможно без математического моделирования транспортных сетей. Математические модели, применяемые для анализа транспортных сетей, весьма разнообразны по решаемым задачам, математическому аппарату, используемым данным и степени детализации описания движения. Одним из видов математических моделей, позволяющих воспроизводить все детали движения, включая развитие процесса во времени, являются имитационные модели. Они позволяют быстро и с хорошей точностью прогнозировать характеристики сложных систем и оптимизировать существенные параметры, выбирая соответствующие параметры управления. Самым популярным средством построения имитационных моделей сегодня считается среда разработки AnyLogic.

Содержательная часть

Процесс создания модели управления трафиком на участке транспортной сети можно разделить на три основных этапа:

1. Моделирование движения автомобилей.
2. Моделирование работы светофора.
3. Моделирование управления трафиком.

Для моделирования движения автотранспорта в данной работе использован дискретно-событийный подход. Автомобили представляют собой заявки, т. е. объекты класса Entity. В основе каждой дискретно-событийной модели лежит диаграмма процесса. Простейшая потоковая диаграмма состоит из 5 элементов: `source`, `queue`, `conveyor`, `hold` и `sink`. `Source` создает заявки в настраиваемые моменты времени; `queue` хранит заявки в определенном порядке; `conveyor` перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними; `hold` блокирует/разблокирует поток заявок на определенном участке блок-схемы; `sink` уничтожает поступившие заявки.

Для наглядности в проект была добавлена карта Красной площади и размечены ломаными линиями пути движения автомобилей. В результате разметки получились траектории движения автомобилей (рис. 1).

Для каждой линии нужно добавить свои объекты `queue`, `conveyor` и `hold`.

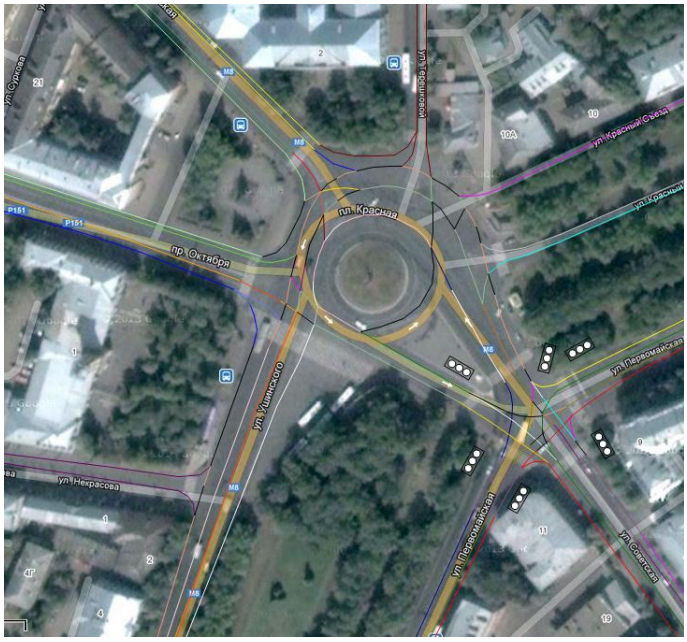


Рис. 1. Карта с разметкой

В результате получилась разветвленная сеть, моделирующая передвижение автомобилей, которые с заданной вероятность выбирают свою траекторию движения.

Каждый светофор представляет собой индивидуального агента со своими характеристиками, но поведение каждого из них влияет на систему в целом. Поэтому для моделирования работы светофора был использован метод Агентного моделирования и его главный инструмент в среде AnyLogic — диаграмма состояний. Для удобства моделирования и во избежание повторения однотипных действий был разработан отдельный класс Java — Светофор, в котором сама диаграмма состояний, а также параметры, отвечающие за переключение сигналов, являются полями. В дополнение к полям в классе имеются методы, обеспечивающие саму работу со светофором, а также рисунок для графического представления данного объекта на диаграмме (рис. 2).

Центральным звеном разрабатываемой модели является объект контроллер, который управляет работой светофоров. Контроллер определяет время, через которое должны переключаться сигналы светофоров. По истечении заданного времени он посылает светофорам сообщения с помощью ранее определенной функции (рис. 3).

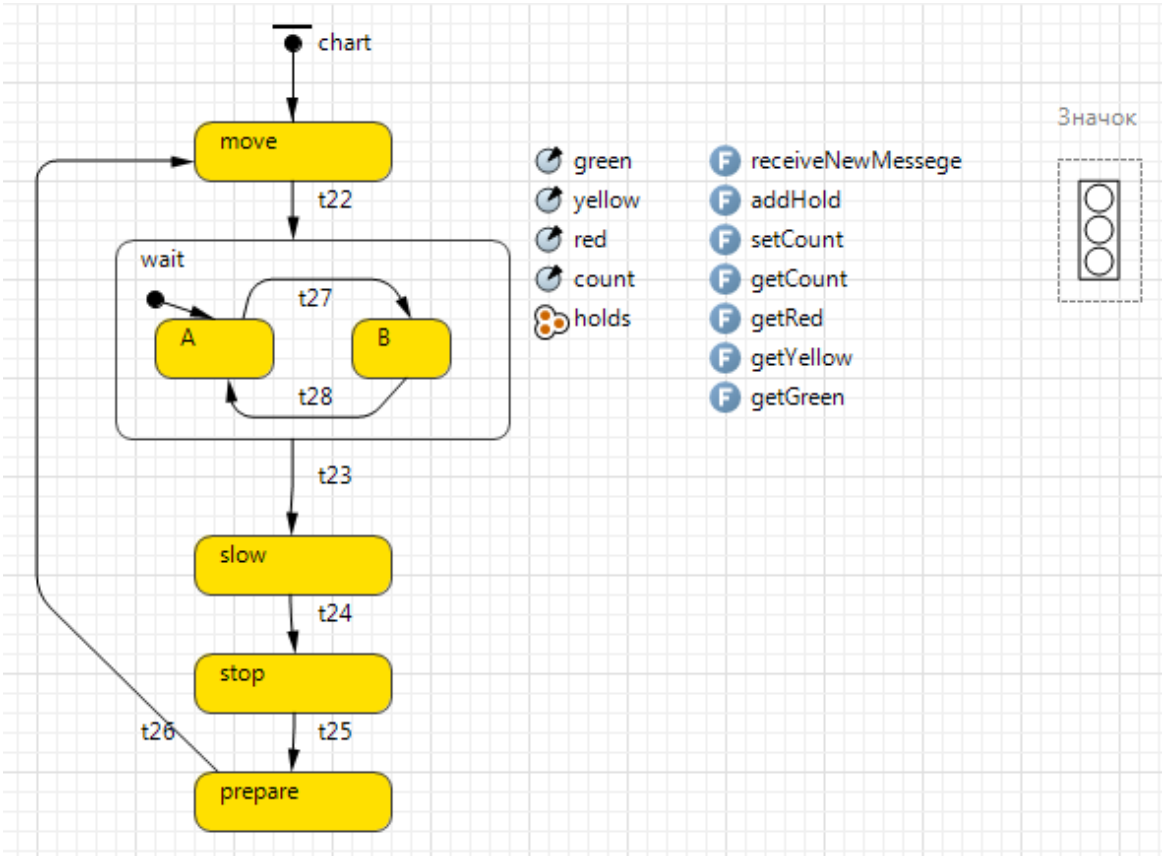


Рис. 2. Класс Светофор

Поступление автомобилей в модель осуществляется в соответствии с непрерывным равномерным распределением с помощью функции uniform (A, B). Параметры A и B определялись путем визуального исследование трафика движения на площади. Начальным условием для работы контроллера является реальное время работы светофоров на выбранном участке транспортной сети. Это время меняется, когда контроллер начинает учитывать интенсивность трафика на данных улицах.

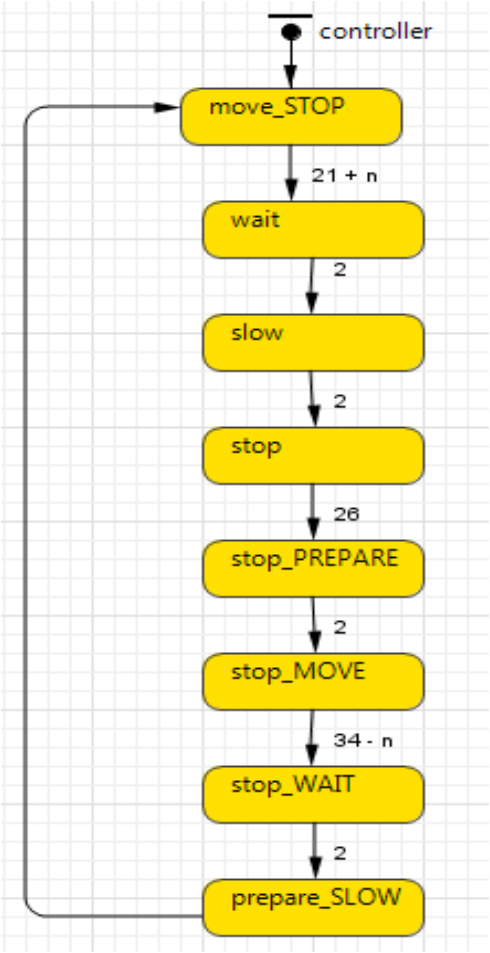


Рис. 3. Контроллер

В рамках данной модели интенсивностью трафика будем считать *количество автомобилей, которые останавливаются на красный свет за одну итерацию работы контроллера.*

Для измерения пропускной способности были определены необходимые отрезки пути. При моделировании движения автомобилей каждой линии на карте соответствует собственный объект `conveyor`. Он отвечает за перемещение автомобилей и может подсчитать количество автомобилей, стоящих на светофоре. Каждому конвейеру определен отдельный параметр, который хранит это значение. В дополнительной диаграмме состояний производится сравнение этих данных для двух пересекающихся улиц, и, в зависимости от результата, определяется количество секунд, на которое будет меняться длительность красного сигнала светофора.

На представленном на рис. 4 графике можно заметить, что при прочих равных условиях для сходной интенсивности

на обеих пересекающихся дорогах показатели длительности красного сигнала разнятся не сильно.

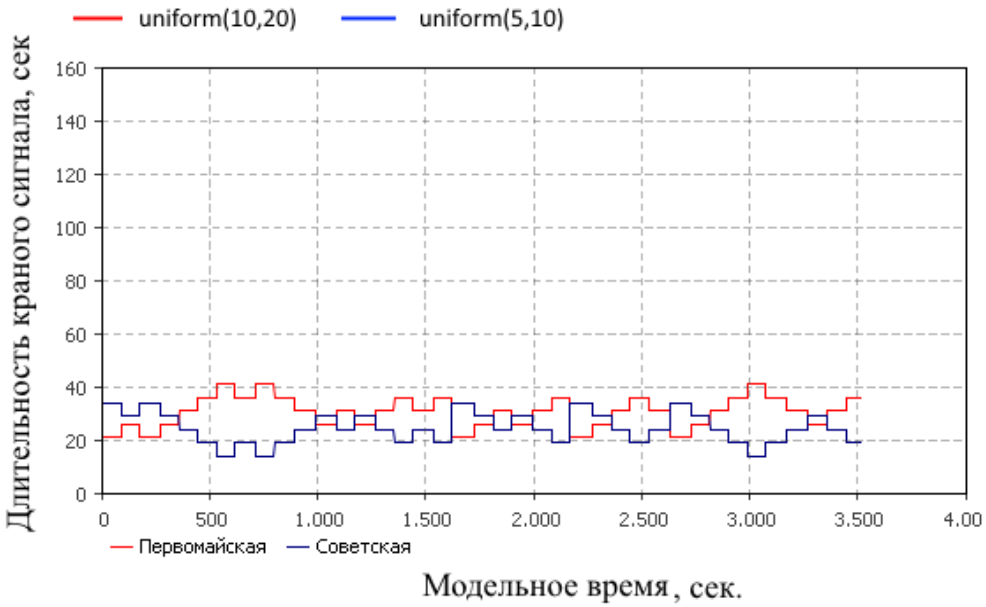


Рис. 4. График длительности красного сигнала

При увеличении интенсивности трафика только на одной из дорог контроллер увеличивает длительность красного сигнала для другой дороги с этого перекрестка (рис. 5).

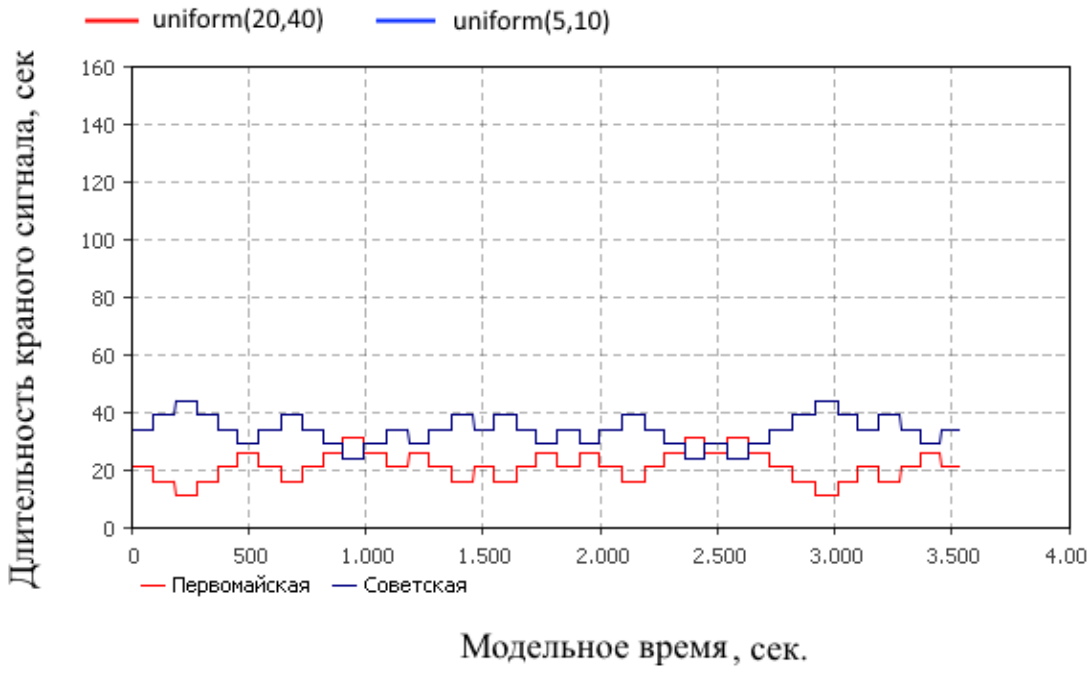


Рис. 5. График длительности красного сигнала
при увеличении интенсивности трафика на одной из дорог

Заключение

В результате выполнения данной работы автором были проанализированы характеристики трафика транспортной сети в пределах Красной площади г. Ярославля. На основе полученной информации в среде AnyLogic построена структура данных, описывающая работу светофора, и разработан интерфейс для управления ею. Разработан интеллектуальный контроллер, производящий локальное адаптивное управление перекрестком. Управление производится при помощи перераспределения длительностей светофорных фаз для разных направлений на основании информации об интенсивности трафика участка дороги. Перекресток управляется независимо от соседних перекрестков.

Предложенная модель контроллера позволяет сократить время прохождения участка сети транспортным средством.

Ссылки

1. Боев В. Д., Кирик Д. И., Сыпченко Р. П. Компьютерное моделирование: пособие для курсового и дипломного проектирования. СПб.: ВАС, 2011. 348 с.

2. Гасников А. В., Кленов С. Л. Введение в математическое моделирование транспортных потоков: учебное пособие. М.: МФТИ, 2010. 362с.

3. Мезенцев К. Н. Моделирование систем в среде AnyLogic 6.4.1: учебное пособие. Ч. 1, 2 / под ред. проф. А. Б. Николаева. М.: МАДИ, 2011.

4. Осоргин А. Е. AnyLogic 6: лабораторный практикум. Самара: ПГК, 2011.

5. AnyLogic. Многоподходное имитационное моделирование. URL: http://www.anylogic.ru/

УДК 004.93

Интабуляция звуковых файлов

В. А. Голубцов

Ярославский государственный университет им. П. Г. Демидова
E-mail: vsv10000ego@gmail.com

Статья посвящена проблеме распознавания звука, в частности распознавания табулатуры музыкального произведения. Описан алгоритм для распознавания табулатур и процесс создания ПО для автоинтабуляции.

Ключевые слова: табулатура, распознавание, музыка, интабуляция, wav.

В данной работе описаны особенности создания соответствующего ПО. Полученному приложению дано название tab_reader. Приложены исходные коды. Представлена необходимая теоретическая база, рассмотрены основные аппаратные проблемы и границы применимости разработанного ПО.

Помимо автоматического распознавания, в программе есть возможность вручную анализировать файл по средствам просмотра различных проекций спектрограмм. На вход программе подается wavфайл, на выходе — форматированный список нот с привязкой по времени.

Программа написана на языке C++11. GUI сформирован при помощи библиотеки Qt. Для прорисовки графиков используется Qwt.

Функции БПФ и чтения wav предоставлены библиотекой studlib, разработанной на базе ЯрГУ М. Л. Мячиным.

Алгоритм работы автоматического распознавания:

1. Поиск моментов атаки по перепаду энергии сигнала.

2. Анализ окон до атаки и после. Формирование первичного списка нот.

3. Удаление различных фоновых звуков, в частности обертонов.

Первая часть алгоритма — это поиск моментов атаки (звукоизвлечения), т. е. сначала локализуем сигнал. В такой предобработке есть масса плюсов. Во-первых, после нее уйдет зависимость от параметров ширины и шага окна (иначе пришлось бы просить пользователя вводить данные параметры). Во-вторых, зная точно моменты атаки, мы уже получаем «ритмический скелет» табулатуры.

Одна из основных проблем при распознавании будет так называемая в цифровой обработке сигналов «проблема кратных частот», она же «проблема обертонов» в акустике. Стоит учитывать, что проблема будет не в том, что мы чего-то не распознаем, а в том, что найдем лишнее. Даже без программной постобработки при наличии «скелета» пользователю не составит труда удалить лишние ноты. Отсюда проблема временной локализации выходит на передний план.

Энергия покоя любой системы есть постоянная, мощность есть перепад энергии

$$P = (E2-E1)/T.$$

Любой перепад мощности говорит о внесении в систему энергии, самый значительный источник энергии в системе «гитарист — гитара» есть сила «атаки» (какое-то трение одежды и тиканье часов не требуют большой силы, следовательно вносят мало энергии). Проанализировав мощность системы от времени, можно сделать вывод о том, когда был момент атаки.

Анализ мощности (в версии выпускной работы) ведется скользящим окном. Нужно учесть, что из-за долгого времени затухания струн и вероятных коротких звуков (тиканье часов, скрежет о струны и прочее) нет гарантии, что пик энергии будет в окне момента атаки. По этой причине поиск моментов атаки ведется по пику отношения мощности текущего и предыдущего окон.

Вторая часть алгоритма — это непосредственный анализ отчетов возле моментов атаки. Вполне очевидно, что информация о сыгранной ноте кроется в разнице частотно-амплитудных составляющих сигнала, точнее некоторых небольших окон до и после момента атаки включительно.

Стоит учитывать, что спектр, по которому ведется сравнение, должен быть проекцией спектрограммы, т. е. полученный суммой спектров (в смысле максимизации спектральных составляющих) в некой малой окрестности от интересуемого момента. Это необ-

ходимо, т. к. каждая спектральная составляющая может пульсировать, однако полупериод таких колебаний, как правило, короткий и не превышает 150 отчетов (для частот нот, используемых при игре на гитаре, во всех используемых тестах). Конкретных параметров, влияющих на полупериод, или какого-то теоретического обоснования эффекта на данный момент мною не найдено, однако замечено, что максимальный полупериод наблюдается в окрестности низких частот, размер полупериода пропорционален общей мощности сигнала (линейность не установлена). Характер колебаний спектральных составляющих зависит исключительно от природы сигнала, так, например, для чистого синуса колебания равномерны, а для затухающего при наличии обертонов (такого, как струна) период увеличивается в низкочастотной области.

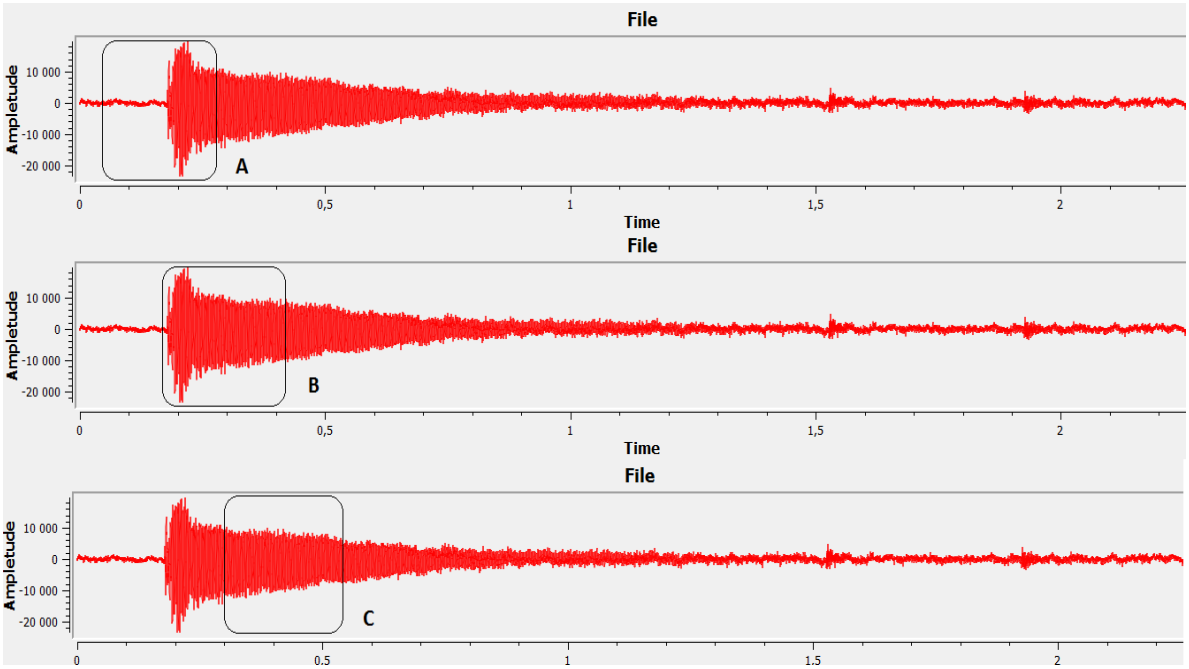


Рис. 1. Tab_reader движение окна в первой части алгоритма

Третьей частью алгоритма стоит провести постобработку — оставить только самые достоверные ноты, отсеять ноты на заднем плане.

Для вычисления сложности будем считать количество умножений, полагая, что каждому умножению соответствует константное количество сложений. В пользу такого подхода говорит то, что нам заранее известна сложность БПФ, посчитанная таким образом: $O(W\log W)$, где W — ширина окна.

Части алгоритма выполняются последовательно, значит общую сложность $O(A)$ можно вычислить как:

$$O(A) = O(a_1) + O(a_2) + O(a_3).$$

Вычислим $O(a_1)$. В каждом окне для просчета мощности выполняется W умножений. Количество таких окон N/S, где N — количество отчетов в файле, S — шаг поиска момента атаки. Так как переменные W и S можно задать константами, сложность первой части алгоритма будет линейна. Также стоит заметить, что W и S незначительны относительно N.

$$O(a_1) = O(W*N/S).$$

Вычислим O (a2). Максимальную ширину окна для формирования звуковых квантов можно задать константой. Эмпирически выявлено, что для уверенной локализации нот в спектральной области достаточно окна шириной 0.2 с (8 820 отчетов, для БПФ берем 8 192 отчетов.). Имеется в виду, что пики спектральных составляющих, соответствующих ноте, вместе с основными боковыми лепестками и с учетом «размазанности» спектра, характерной для такого стохастического процесса, как звукоизвлечение, умещаются на расстоянии полутона от частоты ноты.

С учетом имеющейся после первого шага информации сложность алгоритма равна:

$$O(a_2) = O(K*WlogW),$$

где K — количество моментов атаки.

Вычислим O(a3). На третьем шаге не делается никаких больших вычислений, кроме нескольких сравнений, их сложность сравнима со сложностью умножения.

$$O(a_3) = O(K)$$

$$O(A) = O(W*N/S) + O(K*WlogW) + O(K)$$

Относительно количества отчетов алгоритм линеен:

$$(O(A) = O(N)).$$

Таким образом сложность алгоритма будем считать равной – $O(W*N/S) + O(K*Wlog_2W) + O(K)$. Для сравнения сложность алгоритма без пред- и постобработки – $O(Wlog_2W*N/S)$, имеется в виду алгоритм обрабатывающий вновь поступающую информацию сразу. Для каждой композиции отношение параметров

может быть различно. Даже при худших для используемого в программе алгоритма параметрах, а именно $K = N/S$, когда ноты стоят максимально близко, для того чтобы их можно было распознать, проигрыш несуществен.

При $K = N/S$ локализация по времени и частоте не является приемлемой, для достаточной локализации необходимо $K<<N/S$. Алгоритм без предобработки одновременно как бы пытается локализовать сигнал и по времени и по частоте, что противоречит теореме о локализации спектра (фактически принципу неопределенности), при этом от табулатуры всегда требуется хорошая локализация именно по времени.

Немаловажный вопрос, как близко по времени должны быть расположены ноты, чтобы программа их распознала.

Минимальный размер окна спектра, при котором каждой ноте соответствует минимум одна спектральная составляющая, равен 0.1 с (4 410 семплов при частоте дискретизации 44 100 Гц. Для БПФ округленные до 4 096 семплов.). Следовательно, ноты, между которыми нет отрезка в 0.1 с, не будут распознаны корректно. Из этого же можно вычислить ширину окна поиска по мощности, 0.1 с.

Результаты тестирования

Погрешность по времени на всех тестах меньше 0.1 с (с учетом погрешности испытателя). Алгоритм определил ноты G3 и G4, однако при распознавании E2 и A2 возникли трудности, ниже представлен спектр сигнала в моменты звукоизвлечения E2 и A2. Спектры взяты как проекция спектрограммы, полученной скользящим окном шириной 0.3 с, перекрытие 90 %, с применением окна Хана.

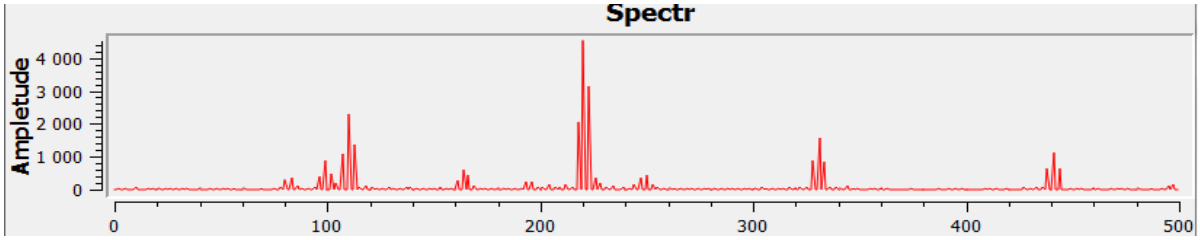


Рис. 2. Tab_reader спектр A2

Однако если просмотреть профессиональную запись, то можно найти ту же ноту без заниженной основной гармоники.

На рис. 3 представлен спектр первой ноты композиции The bardsong группы Blind guardian.

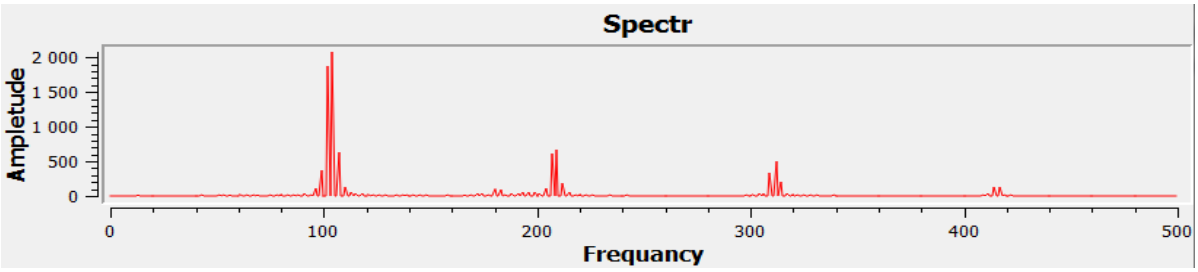


Рис. 3. А2 профессиональная запись

Обертоны, как правило, ниже основной частоты и не сильно повлияют на работу алгоритма, если из-за плохого аппаратного обеспечения не будет занижена основная гармоника.

Также алгоритм плохо работает для последовательности одинаковых нот, сыгранных слишком быстро. Ноты, стоящие на расстоянии < 0.2 с, не успевают утихнуть и распознаются как одна, при расстоянии в 0.3 с для таких нот алгоритм работает. Подобные проблемы может решить дополнительная статистическая постобработка с учетом музыкальной природы звука, а именно попадания в размер, такт.

Ссылки

1. Мячин М. Л. Лекции по цифровой обработке сигналов. Ярославль: ЯрГУ, 1996–2004. 205 с.

2. Дубровский Д. Ю. Компьютер для музыкантов-любителей и профессионалов. М.: ТРИУМФ, 1999. 400 с.

УДК 004

Методы обфускации для языка функционального программирования

Д. С. Данилов, В. А. Башкин

Ярославский государственный университет им. П. Г. Демидова
E-mail: danilov.dmitry00@gmail.com, v_bashkin@mail.ru

В статье рассматриваются методы обфускации функциональных языков программирования. Целью работы является рассмотрение общих методов обфускации для императивных языков и разработка методов, относящихся к функциональным языкам.

Ключевые слова: обфускация, императивный язык, функциональный язык, методы обфускации, SML.

Введение

В современном мире хорошо развита информационно-технологическая индустрия. Все больше людей занимаются разработкой программного обеспечения в целях его продажи. Но информацию легко копировать и распространять нелегально, что наносит ущерб разработчикам, которые владеют лицензией на данный программный продукт.

Существует множество способов защиты авторских прав на программы. Все они делятся на два вида: юридический и технический. Юридический способ — защита программного продукта с помощью закона: приобретение патента, авторских прав, лицензии. Эти меры могут предотвратить желание нелегально использовать программный продукт, но фактической защиты интеллектуальной собственности не обеспечивают.

Второй способ, технический, более практичен. Он подразумевает защиту программ путем включения в них каких-либо средств защиты продукта. Наиболее распространены такие виды защиты, как выполнение программ на стороне сервера, использование «водяных» знаков, отпечатков пальца, добавление процедуры проверки исходного кода, шифрование кода и обфускация.

Большинство существующих обфускаторов предназначены для запутывания кода, написанного на императивных языках программирования. Нами разработаны методы обфускации функциональных языков.

Постановка задачи

Одной из важных составляющих работы является ознакомление с понятием обфускации; рассмотрение существующих методов обфускации программ, написанных на императивных языках программирования.

Другая важная составляющая — разработка методов обфускации для функциональных языков программирования.

Целью работы являлось создание обфускатора для функционального языка программирования на примере языка SML.

Обфускация

Обфускация (obfuscation — запутывание) — это метод защиты программного продукта путем усложнения процесса понимания кода. Суть обфускации в том, чтобы запутать исходный код, устранить логические связи в нем, что затруднит изучение программного продукта третьими лицами.

Принцип обфускации направлен не на какие-либо сложные вычисления для расшифровки, а на психологические слабости человека. Код, не поддающийся логике, намного сложнее понять, чем хорошо структурированный. Именно это используется при обфускации.

Процесс обфускации

Процесс трансформации программного кода является обфускацией, если соблюдены следующие условия:

• код после обфускации должен существенно отличаться от исходного, но выполнять то же самое,

• изучение логики работы программы должно стать более трудным и занимать больше времени,

• применение обфускации к одному и тому же коду несколько раз должно давать разные результаты,

• создание деобфускационной программы неэффективно.

Алгоритмы обфускации

Алгоритм Колберга

1. Загрузка элементов программы.

2. Загрузка библиотек.

3. Осуществление обфускации над программой путем выделения фрагмента кода и определения наиболее эффективного процесса трансформации для него. Этот этап повторяется до тех пор, пока не будет достигнут требуемый уровень обфускации или допустимое увеличение ресурсов.

4. Генерация трансформируемой программы.

Данный алгоритм не использует какой-то определенный метод обфускации, поэтому считается общим алгоритмом.

ChenxiWang`s алгоритм

В качестве входных данных алгоритм принимает типичную процедуру, написанную на языке высокого уровня. Процесс обфускации каждой такой процедуры состоит из трех этапов:

• создание графа потока управления этой процедуры (граф задается множеством блоков и множеством связей, соединяющих их), после чего граф разбивается путем замены циклических конструкций в нем на конструкции типа «if (условие) goto»;

• нумерация всех блоков в графе и добавление в код процедуры переменной, хранящей номер следующего выполняемого блока;

• приведение графа к однородному («плоскому») виду.

Свойства функциональных языков

• Строгая типизация.

• Модульность.

• Чистота (отсутствие побочных эффектов).

• Отложенные (ленивые) вычисления.

• Краткость и простота.

• Функции — это значения.

Заключение

Разработана программа-обфускатор функционального языка SML. В ней реализованы такие методы обфускации, как добавление непрозрачных предикатов, добавление недостижимого кода, использование сопоставления с образцом, **устранение библиотечных вызовов.**

В будущем программу можно развивать, добавив новые методы обфускации, позволяющие работать не с одной функцией, а с целым набором, т. е. с полной программой. Также имеется возможность создания графического интерфейса обфускатора

для удобства работы пользователя. Такие преобразования позволят использовать обфускатор для запутывания более сложных функциональных программ.

Ссылки

1. Лифшиц Ю. М. Запутывание (обфускация) программ: обзор. URL: http://logic.pdmi.ras.ru/~yura/of/survey1.pdf

2. Чернов А. В. Анализ запутывающих преобразований программ // Библиотека аналитической информации. URL: http://www.citforum.ru/security/articles/analysis

УДК 81.322.2

Обзор методов извлечения гипо/гиперонимов из коллекций документов

М. С. Каряева, В. А. Соколов

Ярославский государственный университет им. П. Г. Демидова
E-mail: mari.karyaeva@gmail.com, valery-sokolov@yandex.ru

В статье описаны методы извлечения гипонимов и гиперонимов из коллекции документов, в качестве которой могут выступать толковые словари, словари предметных областей и другие справочники с информацией, представленной в виде терминологической статьи с термином и его определением.

Ключевые слова: компьютерная лингвистика, семантические отношения, меры семантической близости, терминоведение, гипонимы, гиперонимы.

Введение

Согласно статистике, приведенной на ресурсе Internet Live Stats [1], за секунду интернет-трафик составляет 27 тыс. Гб. Новости, посты, электронные сообщения — это информация, которую необходимо своевременно обрабатывать для анализа новостей, социальных сетей, позитивных и негативных отзывов.

В качестве модели для хранения извлеченной информации используются базы данных. Обычно это разновидности словарей, такие как классификатор, онтология, таксономия или тезаурус. Последний вариант наиболее популярен сегодня и применяется в таких лингвистических ресурсах, как WordNet [2], PyTez [3], Yarn [4]. Данные ресурсы имеют общую схему построения базы данных, которая отличает тезаурус от другой разновидности словарей. Базовой словарной единицей в тезаурусах является не отдельное слово, а так называемый синонимический ряд («синсет»), объединяющий слова со схожим значением и являющийся узлом сети. Однако, кроме отношения синонимии, могут быть представлены и другие виды отношений, например родо-видовое отношение, отношение «часть — целое», отношение несимметричной ассоциации, отношение симметричной ассоциации.

Люди имеют уникальную способность определять взаимосвязь между двумя лексическими единицами. Например, большинство согласится, что автомобиль и шина взаимосвязаны, а автомобиль и дерево — нет. Но само понятие «связность» имеет широкое значение: два понятия могут быть взаимосвязаны, потому что одно понятие является частным случаем другого (транспортное средство, мотоцикл) или одно понятие является частью другого (автомобиль, шина). Кроме того, между понятиями существуют отношения синонимии (автомобиль, машина). Установление отношений между понятиями может решать ряд проблем, например: классификация текстовых документов, автоматическое создание баз данных, улучшение качества поисковых запросов.

Выявление взаимосвязи между понятиями с использованием ручной разметки — трудоемкий и дорогостоящий процесс, который не может подчиняться объективным правилам извлечения отношений. Создание автоматических правил для извлечения отношений между понятиями сможет ускорить процесс, а привлечение метрик для оценки определит качество полученных результатов.

Отношение «вид — род»

Отношение «вид — род» (is-kind-ofrelation) — объединение однородных лексических единиц в класс на основе их принадлежности к более общему родовому понятию. *Ассонанс*, *консонанс*, *диссонанс* принадлежат одному классу и являются гипонимами, или видовыми понятиями, к термину *рифма*, который

является гиперонимом, или родовым понятием. Однако не ко всем видовым понятиям можно подобрать родовое понятие, так, например, у термина *рифма* нет родового понятия.

Пример из «Поэтического словаря» А. П. Квятковского [5], где определения состоят из одного предложения, в котором первое словосочетание является гипонимом к термину.

РИ́ФМА (от греч. ρυθμός — соразмерность) — композиционно-звуковой повтор преимущественно в конце двух или нескольких стихов, чаще — начиная с последнего ударного слога в рифмуемых словах.

→ ПОВТО́РЫ — стилистические признаки, присущие поэзии и этим отличающие ее от прозы, как противостоящей стилевой категории.

В примере продемонстрированы термины с определениями, из которых можно извлечь родо-видовые отношения:

РОД_ВИД (ПОВТОРЫ; РИФМА)

Использование лексико-синтаксических шаблонов

Первый метод, который позволял извлекать отношения между терминами, был опубликован в 1992 г. Мартой Херст [6], профессором в области лингвистики, которая занималась разработкой принстонского WordNet. Именно она определила метод извлечения семантических отношений с использованием лексико-синтаксических шаблонов и привела примеры и результаты их применения. Это один из самых простых способов для извлечения отношений. Необходимо ввести конструкцию, которая бы встречалась в коллекции документов, и с помощью этой конструкции можно было бы выявить понятия, состоящие в семантических отношениях. Довольно удобно применять такие конструкции для словарей, где вся информация упорядочена и составлена по общему правилу представления информации, а именно «термин — определение».

Примеры шаблонов из работы Марты Херст:

- such NP as {NP ,}* {(or | and)} NP

  ... works by such authors as Herrick, Goldsmith, and Shakespeare.
- NP {, NP}* {,} or other NP

  Bruises, wounds, broken bones or other injuries...
- NP {, NP}* {,} and other NP

  ... temples, treasuries, and other important civic buildings.

Алгоритм выявления шаблонов из коллекции документов в виде словаря:

1. Ручной поиск пар терминов в коллекции документов, между которыми есть родо-видовые отношения.
2. Фиксация «окружения» вокруг найденных пар терминов.
3. Анализ «окружения», попытка ручной формализации лингвистических конструкций «окружения».

В работе [7] приведены примеры шаблонов, используемых для извлечения гипонимов и гиперонимов для русской лексики в системе Serelex [8], которая устанавливает семантическую связь между словами без учета типа отношений.

Пример из «Поэтического словаря» Квятковского, где используется конструкция для извлечения родо-видовых отношений:

ТЕРМИН — … #$к_{(предлог)}$#... # ТЕРМИН#... # $относятся_{(гл., мн.ч.)}$ #... #$(ТЕРМИН_N)^+$, где N — натуральное число.

ЖАНР (франц. genre — род, вид) — в русской поэтике под словом Ж. разумеется определенный вид литературных произведений, принадлежащих одному и тому же роду. Различаются три рода художественной литературы — эпос, лирика и драма. К эпическим Ж.относятся: эпопея, былина, сказка, поэма, роман, повесть, новелла, рассказ, басня, художественный очерк и т. п.

ЗВУКОВЫ́Е ПОВТО́РЫ — эвфонический прием, заключающийся в повторении внутри стиха и в соседних стихах группы одинаковых или похожих звуков. К З. п. в широком значении относятся аллитерация, рифма, ассонанс, диссонанс.

РОД_ВИД (ЖАНР; ЭПОПЕЯ/БЫЛИНА/СКАЗКА)
РОД_ВИД (ЗВУКОВЫЕ ПОВТОРЫ;
АЛЛИТЕРАЦИЯ/РИФМА/АССОНАНС/ДИССОНАНС)

Использование методов машинного обучения

Вторая разновидность методов для извлечения гионимов и гиперонимов основана на методах с использованием машинного обучения [9]. Принципы основаны на таких дисциплинах, как методы оптимизации, теория вероятностей и др.

Алгоритм

1. Выбор пар (гипоним, гипероним).
2. Поиск предложений, где употребляется каждая пара.

3. Анализ предложения для извлечения шаблона (графы, деревья).

4. Тренировка классификатора на созданном наборе данных.

Классификаторы [10]:

1.Maximum Entropy Classifiers

2.Support Vector Machine

3.Naive Bayes classifier

Последний этап — это применение методов машинного обучения, когда на созданной выборке можно научить алгоритм классифицировать все остальные термины.

Использование эмпирических правил

Данный подход предполагает набор правил, который может быть выведен эмпирическим путем и не имеет никакого теоретического обоснования. Суть метода в том, что определения строятся по неким правилам. В некоторых словарях конструкция построения определения строго фиксирована. Таким образом, можно разработать ряд правил, которые с высокой точностью будут извлекать отношения.

Например, правило «извлечение первого и второго существительного в определении» или «извлечение первого существительного в именительном падеже» часто указывает на отношение «род — вид» между термином и извлеченным существительным. Ниже продемонстрированы примеры терминов и их определений из «Википедии»:

Рома́н — литературный жанр, как правило прозаический, который предполагает развернутое повествование о жизни и развитии личности главного героя (героев) в кризисный, нестандартный период его жизни.

Ко́шка — домашнее животное, одно из наиболее популярных (наряду с собакой) «животных-компаньонов».

Англи́йский бульдо́г — короткошерстная порода собак 2-й группы МКФ, подгруппа молоссов и мастифов.

Заключение

Использование методов извлечения гипонимов и гиперонимов из коллекции документов необходимо для автоматического построения таких баз даных, как тезаурус. Каждый из перечис-

ленных методов по извлечению родо-видовых отношений имеет как преимущества, так и недостатки.

Метод с применением лексико-синтаксических шаблонов гарантирует высокую точность, поскольку обычно определения в словарях составлены по единому принципу, однако не гарантирует высокой полноты.

Метод, основанный на машинном обучении, удобен в использовании для любой коллекции документов и нет необходимости в привлечении экспертов для ручного извлечения терминов и оценке их «окружения».

Объединение данных методов может гарантировать результат для создания как тезауруса общей направленности, так и тезаурусов предметных областей.

Ссылки

1. Internet Live Stats. URL: http://www.internetlivestats.com/
2. WordNet. URL: https://wordnet.princeton.edu/
3. Тезаурус РуТез. URL: http://www.labinform.ru/pub/ruthes/index.htm
4. Тезаурус YARN. URL: http://russianword.net/
5. Квятковский А. П. Поэтический словарь / науч. ред. И. Роднянская. М.: Советская энциклопедия, 1966. 376 с.
6. Hearst M. A. Automatic acquisition of hyponyms from large text corpora // Proceedings of the 14th conference on Computational linguistics. Volume 2. Association for Computational Linguistics, 1992. С. 539–545.
7. Sabirova K., Lukanin A. Automatic Extraction of Hypernyms and Hyponyms from Russian Texts // Supplementary Proceedings of the 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST 2014) / ed. by D. I. Ignatov, M. Y. Khachay, A. Panchenko, N. Konstantinova, R. Yavorsky, D. Ustalov. 2014. Т. 1197. С. 35–40.
8. Serelex — поиск семантически связных слов. URL: http://serelex.cental.be/ru
9. Relation Extraction. URL: http://web.stanford.edu/class/cs124/lec/rel.pdf
10. Воронцов К. В. Машинное обучение: курс лекций. URL: http://www.machinelearning.ru/

УДК 519

Методы восстановления текстуры на изображении

И. А. Козырев, О. А. Дунаева

Ярославский государственный университет им. П. Г. Демидова
E-mail: lyudjik@gmail.com, olady@gmail.com

В статье описано решение задачи предобработки эндоскопических изображений желудка, рассматривается метод восстановления текстуры на месте засвеченных областей на основе уравнения Пуассона.

Ключевые слова: цифровая обработка изображений, восстановление текстуры, бесшовное наложение, эндоскопия желудка.

В последнее время в различных отраслях, в том числе и в медицине, получили распространение системы принятия решений. Для ранней диагностики рака желудка врачи используют увеличительную эндоскопию, при этом эндоскопические изображения анализируются экспертом. Поэтому задача автоматической классификации эндоскопических изображений желудка по типу слизистой является очень актуальной. Стоит отметить, что на эндоскопических изображениях часто возникают различного рода артефакты, которые мешают их автоматической обработке. Данная статья посвящена задаче предобработки эндоскопических изображений желудка для их дальнейшей классификации. Целью предобработки является удаление нежелательных бликов на эндоскопических изображениях. Для решения этой задачи необходимо восстановить текстуру изображения на месте засвеченных областей. В статье описан метод бесшовного наложения изображений на основе уравнения Пуассона, программа, основанная на этом методе.

Для начала поясним, что такое бесшовное наложение изображений [2] (image blending). Бесшовным наложением изображений называется метод, позволяющий наложить часть одного изображения на другое так, чтобы не было заметно швов на результирующем изображении в местах наложения (рис. 1).

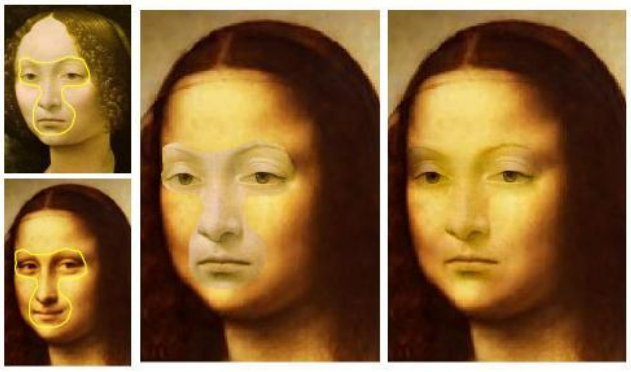


Рис.1. Пример бесшовного наложения

В основе метода бесшовного наложения изображений лежит уравнение Пуассона, которое общем виде можно сформулировать следующим образом. Пусть Ω — замкнутое подмножество $\mathbb{R}^2$ с границей $\partial\Omega$ (рис. 2), а **f** — неизвестная скалярная функция, заданная внутри Ω. Наконец, пусть **v** — векторное поле, заданное на Ω. Необходимо восстановить функцию **f**, векторное поле градиентов которой равно **v**. Отсюда получаем уравнение Пуассона $\Delta f = v$.

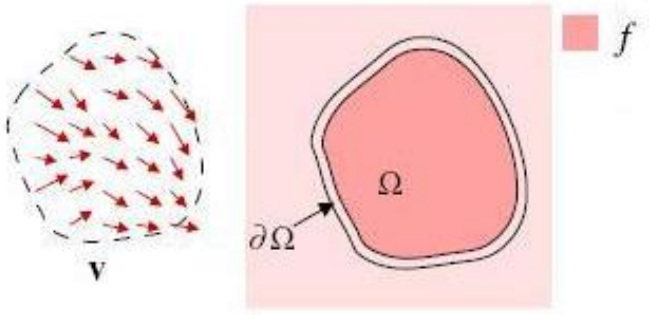


Рис. 2. Начальные условия задачи

Чтобы доопределить задачу, следует добавить краевые условия. Эти условия бывают трех видов:

• Дирихле: $f|_{\partial\Omega} = f^*|_{\partial\Omega}$: ограничения накладываются на саму функцию f на границе;

• Неймана:$f^*|_{\partial\Omega} = f'|_{\partial\Omega}$: условия накладываются на производную;

• Смешанные.

Здесь f^* — некоторая заранее заданная функция.

Уравнение Пуассона в общем виде чаще всего применяется в решении задач математической физики, однако данное уравнение можно применить и к обработке изображений.

Пусть есть два изображения A и B, а также задана некоторая область Ω, в которой будет происходить наложение. На изображение B будет производиться наложение соответствующей области Ω из изображения A. Задача состоит в построении изображения H, градиент которого в области Ω будет равен градиенту изображения A в этой же области ($\Delta H|_\Omega = \Delta B|_\Omega$), а пиксели на границе Ω изображения H должны совпадать с пикселями изображения B на этой же границе ($H|_{\partial\Omega} = A|_{\partial\Omega}$).

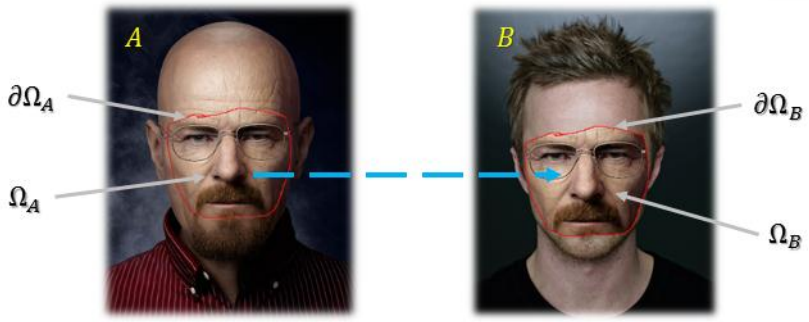


Рис. 3. Начальные условия задачи в обработке изображений

Таким образом, получается система из k уравнений, где k — количество пикселей, которые мы должны найти для изображения H. В разработанной программе для решения данной системы линейных уравнений был использован метод простых итераций Якоби.

На основе метода бесшовного наложения изображений была разработана программа, автоматически восстанавливающая текстуру на эндоскопическом изображении в местах бликов. На вход программы подается эндоскопическое изображение и его маска, указывающая на места бликов (рис. 4).

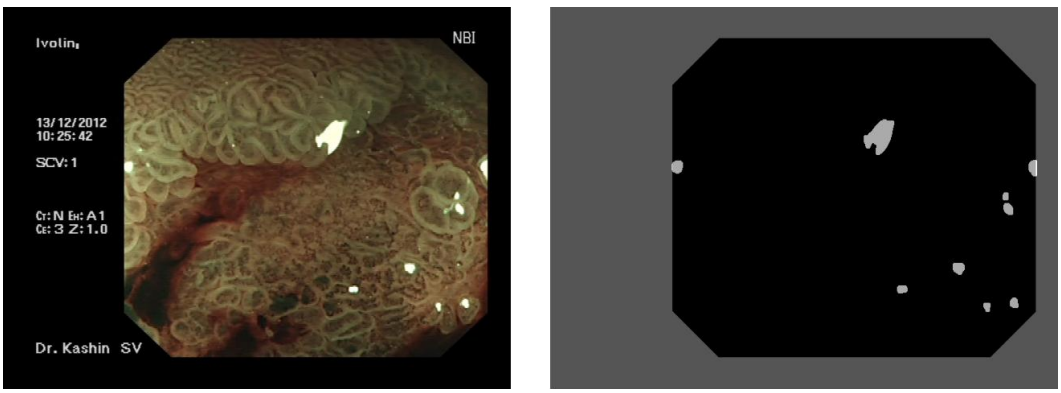


Рис. 4. Входные данные для приложения. Слева — исходное изображение; справа — маска исходного изображения

Алгоритм разработанной программы состоит из нескольких основных шагов (рис 5).

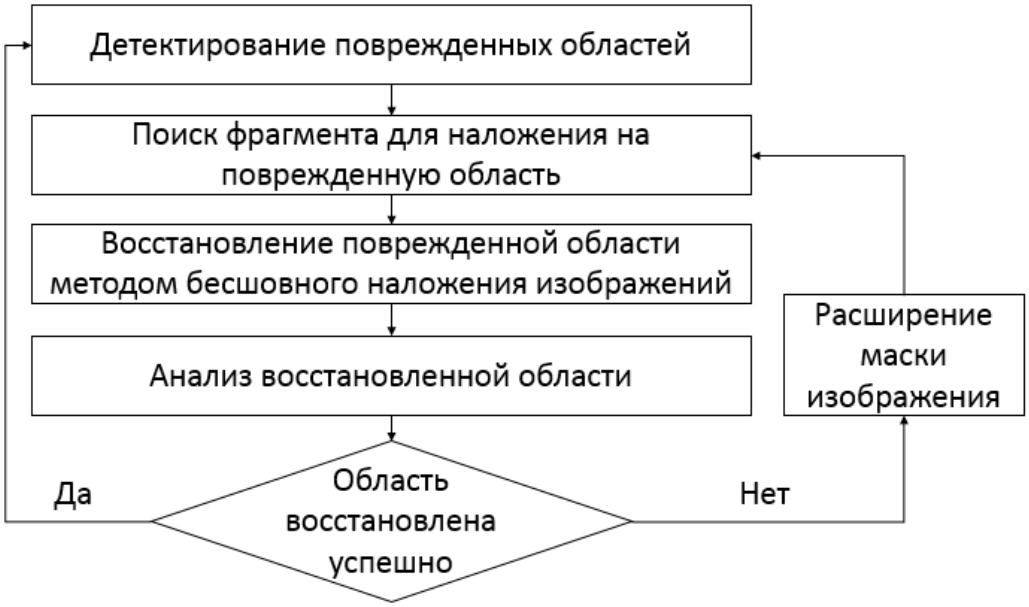


Рис. 5. Шаги работы алгоритма

На рис. 6 показан пример восстановления текстуры на эндоскопическом изображении. Как можно заметить, разработанная программа для восстановления текстур в местах бликов на эндоскопических изображениях показала довольно хорошие результаты.

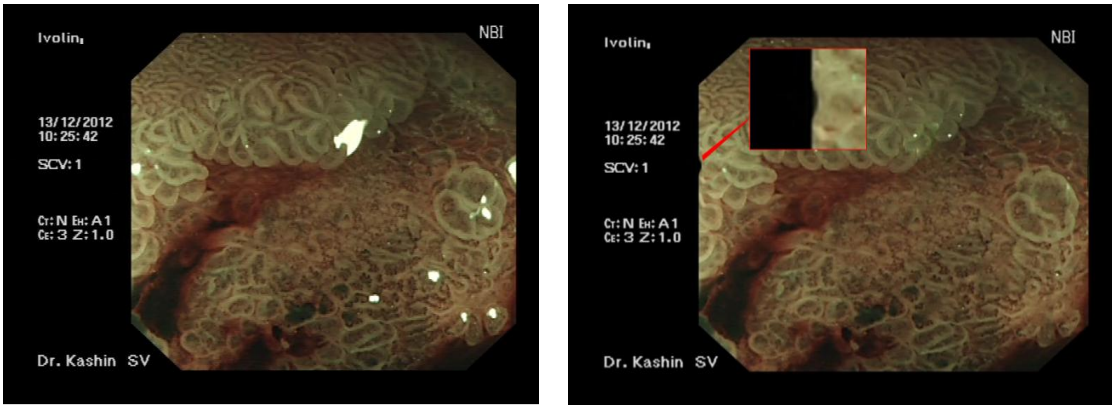


Рис. 6. Слева — исходное изображение; справа — результат

Однако данный метод восстановления текстуры иногда работает не очень хорошо. В случае если граница поврежденного или засвеченного участка лежит вплотную к краю изображения, то абсолютно черная граница будет влиять на цветовой баланс в восстановленной области. Это происходит в связи с тем, что восстановление поврежденного участка происходит исходя из границ

этого участка. Как можно заметить на рис. 6, одна из поврежденных областей лежала вплотную к границе окаймляющего шестиугольника, поэтому восстановление прошло не совсем удачно.

Несмотря на некоторые недостатки данного алгоритма, разработанная программа достаточно качественно восстанавливает поврежденные участки и может помочь улучшить качество последующей обработки эндоскопических изображений желудка, что позволит лучше диагностировать рак на ранних стадиях.

Ссылки

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2005.

2. Кононов В., Конушин В. Применение уравнения Пуассона в задачах обработки изображений. Компьютерная графика и мультимедиа. URL: http://cgm.computergraphics.ru/issues/issue17/poissonimaging

3. Jianbing Shen, Xiaogang Jin, Chuan Zhou, Charlie C. L. Wang. Gradient based image completion by solving the Poissonequation. URL: http://www.cad.zju.edu.cn/home/jin/papers/cg2006.pdf

УДК 51

Вопросы достижимости для векторных систем сложения

А. М. Королева, Ю. А. Белов

Ярославский государственный университет им. П. Г. Демидова
E-mail: naskormi2@yandex.ru, belov45@yandex.ru

В статье описаны эквивалентные в постановке задачи достижимости системы, такие как сети Петри, системы векторного сложения (VAS) и системы векторного сложения с состояниями (VASS). Рассматривается основная проблема для систем — проблема достижимости.

Ключевые слова: системы векторного сложения, системы векторного сложения с состояниями, сети Петри, достижимость, дерево покрытия.

Система векторного сложения (VAS) является одним из нескольких математических языков моделирования для описания распределенных систем. Существует разновидность векторного сложения, которая предполагает наличие конечного управления, — системы векторного сложения с состояниями. При добавлении управляющих состояний возможно уменьшение размерности системы, однако вычислительная мощность при этом не меняется.

Системы векторного и системы векторного сложения с состояниями эквивалентны во многом сетям Петри, введенными ранее Карлом Адамом Петри.

Система векторного сложения состоит из конечного набора целых векторов. Начальный вектор рассматривается в качестве исходных значений нескольких компонент, и векторы VAS рассматриваются как переходы. Эти компоненты не могут опускаться ниже нуля. Система векторного сложения с состояниями — это VAS, оснащенная управляющими состояниями. Точнее, это конечный ориентированный граф с дугами, помеченными векторами. VASS имеют то же ограничение: значения компонент никогда не должны опускаться ниже нуля.

N-мерной системой векторного сложения W называется конечное подмножество множества $\mathbb{Z}^n$, n-мерная система векторного сложения (VAS — *vector addition system*) представляет собой пару $W = (d, W)$, где $d \in \mathbb{N}^n$ — начальный вектор, а $W \subseteq \mathbb{Z}^n$ — схема векторного сложения.

N-мерной системой векторного сложения с состояниями называется система векторного сложения W, снабженная конечным множеством управляющих состояний $Q = \{q_1, q_2, \ldots, q_k\}$ и отношением $T \subseteq Q \times W \times Q$, задающим правило переходов. Переход $(q, w, q') \in T$, или просто $q \to (q', w)$, означает, что из вектора x состояния q порождается вектор $x + w$ с последующим переходом в состояние q’ при условии выполнения $x + w \geq 0^n$.

N-мерная система векторного сложения с состояниями представляет собой набор $V = (x_0, q_0, W, Q, T)$, где (W, Q, T) — это система векторного сложения с состояниями, а $x_0 \in \mathbb{N}^n$ и $q_0 \in Q$ — начальный вектор и начальное состояние соответственно.

Известно, что векторным системам сложения (и системам с состояниями) эквивалентны сети Петри. Для сетей Петри проблема достижимости решается с помощью дерева покрытия. Дерево представляется множеством состояний сети. Это дерево показывает все маркировки, непосредственно достижимые из начальной маркировки. Было доказано, что алгоритм для дерева покрытия конечен и не может создавать новые концевые вершины бесконечно.

Сети Петри, системы векторного сложения и системы векторного сложения с состояниями эквивалентны друг другу, т. к. каждую систему можно построить из любой другой. Для задачи достижимости систем векторного сложения можно также использовать построение дерева покрытия. Можно построить неполное дерево покрытия, в котором будет рассматриваться только часть переходов, не превышающих определенного вектора. И для любой вершины, которую нужно достичь, будет показан путь (являющийся W-путем) последовательности векторов, благодаря которому будет достигаться нужная достижимая вершина из начального вектора.

Была рассмотрена задача достижимости, и некоторые варианты ее решения. Для всех трех систем дерево всегда может быть построено, т. к. системы эквивалентны друг другу и могут быть смоделированы каждая из любой другой.

Ссылки

1. Кузьмин Е. В. Счетчиковые машины: учебное пособие. Ярославль: ЯрГУ, 2010. 128 с.

2. Jérôme Leroux. The reachability problem for vector addition systems. URL: http://highlights-conference.org/pub/leroux.pdf

3. Florent Avellaneda, Rémi Morin. Checking Two Structural Properties of Vector Addition Systems with States. URL: http://movep.lif.univ-mrs.fr/documents/avellaneda-slides.pdf

4. Hopcroft J. E. On the reachability problem for 5-dimensional vector addition systems. URL: http://www.researchgate.net/publication/220151637_On_the_Reachability_Problem_for_5-Dimensional_Vector_Addition_Systems

5. Котов В. Е. Сети Петри. М.: Наука, 1984. 160 с.

6. Jérôme Leroux. Vector Addition System Reachability Problem. URL: http://www.lsv.ens-cachan.fr/Events/Pavas/slides-Leroux.pdf

УДК 414.3

Использование идентификационных меток для прототипа мобильного сервиса для контроля и поиска вещей

К. В. Лагутина

Ярославский государственный университет им. П. Г. Демидова
E-mail: lagutinakv@mail.ru

В статье описаны технические показатели и характеристики различных типов меток, определяются наиболее эффективные метки для реализации сценариев использования системы помощи в поиске вещей, сборе и контроле их наличия. Из четырех рассмотренных технологий электронных меток выбраны три, удовлетворяющие требованиям указанной системы.

Ключевые слова: электронные метки, NFC, RFID, Bluetooth, Wi-Fi.

Каждый человек сталкивается с проблемой найти и собрать нужные вещи, а также следить за тем, чтобы их не забыть или не потерять. По статистике сайта «Бюро находок», чаще всего теряются небольшие предметы: документы, драгоценности, телефоны и ключи [1]. При этом к владельцам возвращается 75–80 % потерянных вещей [2].

В связи с этим возникает задача: разработать систему, которая помогает в поиске, сборе вещей, контроле наличия вещей.

Удобным инструментом для быстрого поиска предметов является электронная метка. Это компактный чип, который может быть прикреплен практически к любому предмету. Кроме того, каждая метка имеет свой уникальный номер (UID), тем самым позволяя однозначно себя идентифицировать, и может контактировать со специальным сканером на расстоянии. Таким образом, с использованием электронных меток процесс поиска предмета или проверки его наличия рядом с владельцем существенно упрощается и ускоряется.

Предлагаемая автором система включает в себя электронные метки, сканер и мобильное приложение. Вещи снабжаются метками. Сканер помещается в сумку, ящик стола, любой другой контейнер. В мобильном телефоне создается база данных предметов с метками, а связь между всеми компонентами системы осуществляет мобильное приложение. Приложение должно осуществлять идентификацию предметов, создание списков предметов и проверку списка на наличие вещей рядом с владельцем.

Существует четыре типа технологий электронных меток: Wi-Fi, Bluetooth, RFID, NFC.

Wi-Fi (англ. Wireless Fidelity — беспроводная точность) — технология передачи данных между электронными устройствами посредством их беспроводного объединения в сеть или подключения к сети Интернет.

Электронные метки, основанные на этой технологии, работают независимо, передают данные измерения мощности сигнала, что может использоваться для определения расстояния между меткой и мобильным устройством. Кроме того, электронные метки предоставляют двустороннюю связь для взаимодействия с системой управления. Некоторые метки (например, технологии Ekahau) также имеют оптические датчики вмешательства, которые подают сигнал системе в том случае, если они были сняты с предмета.

Срок работы меток составляет 5 лет. Радиус действия — до 150 метров. Средние размеры 45 х 55 х 20 мм [3].

Bluetooth — это технология беспроводной передачи данных на небольшое расстояние между различными типами устройств. Взаимодействие между электронными метками, применяющими этот стандарт связи, может осуществляться на расстоянии до 100 м.

Беспроводная технология Bluetooth с низким энергопотреблением (Bluetooth low energy, Bluetooth LE) была выпущена в 2009 г. Основным достоинством этой версии спецификации ядра Bluetooth является сверхмалое пиковое энергопотребление. Электронные метки, использующие Bluetooth с низким энергопотреблением служат около года, не требуя дополнительной подзарядки. Размер подобной метки составляет 25 х 25 х 5 мм [4].

NFC — сокращение от Near Field Communication. Это международный стандарт для бесконтактного обмена данными. В отличие от большинства других технологий, таких как беспро-

водные сети и Bluetooth, максимальное расстояние между двумя взаимодействующими устройствами составляет не более 15 см.

Есть целый ряд NFC-меток, информацию с которых можно считывать при помощи смартфона: от простых наклеек и брелоков до сложных карт с интегрированными шифровальными средствами. Метки различаются и по технологии чипа. Самая распространенная — NDEF, она поддерживается большинством NFC-меток.

Существенное преимущество NFC над Bluetooth — более короткое время установки соединения. Вместо выполнения инструкций по согласованию для идентифицирования устройства Bluetooth связь между двумя устройствами NFC устанавливается сразу (менее чем за одну десятую секунды).

Максимальная скорость передачи данных NFC (424 кбод) меньше, чем Bluetooth (24 Мбод). Срок работы батареи — до трех лет. Средний размер составляет 35 х 35 х 5 мм. У NFC небольшой радиус действия, который обеспечивает бóльшую степень безопасности и делает NFC подходящей для переполненных пространств, где установление соответствия между сигналом и передавшим его физическим устройством (и как следствие, его пользователем) могло бы иначе оказаться невозможным [5].

RFID (Radio Frequency Identification, радиочастотная идентификация) — это самостоятельное направление, входящее в группу автоматической идентификации и регистрации объектов при помощи радиочастотного канала связи. Передача цифрового кода производится при помощи антенны, вмонтированной в электронную метку и представляющей с ней одно целое. Считывание уникального кода из памяти метки производится по запросу другого устройства — сканера.

Сканирующее устройство может получать данные с нескольких меток одновременно, при этом каждая RFID-метка будет идентифицирована однозначно. RFID-считывание не требует прямой видимости: сканер способен взаимодействовать и со скрытыми метками.

По типу источника питания RFID-метки делятся на три вида: пассивные, активные и полупассивные.

Пассивные RFID-метки не имеют встроенного источника энергии. Максимальное расстояние между меткой и сканером варьиру-

ется от 10 см до нескольких метров в зависимости от размеров антенны. Срок службы пассивных меток не ограничен.

Активные RFID-метки имеют собственный источник питания и не зависят от энергии считывателя, вследствие чего они читаются на дальнем расстоянии. Время работы батарей — до 10лет. Радиус считывания метки — до 300 метров.

Полупассивные (полуактивные) RFID-метки во многом аналогичны пассивным, однако оснащены собственной батареей. Дальность действия этих меток зависит только от чувствительности приемника сканера, поэтому они могут функционировать на большем расстоянии и с лучшими характеристиками, чем полностью пассивные метки.

Все четыре типа электронных меток могут быть использованы в рамках предлагаемой системы, т. к. все они могут вступать во взаимодействие со смартфоном непосредственно или с помощью специального сканера (RFID), при этом обмен данными происходит достаточно быстро для пользователя [8].

С целью дальнейшего определения наиболее эффективных типов меток для использования в предлагаемой системе автором были разработаны несколько сценариев использования. Сценарий использования (use case) — это спецификация последовательности действий, которые может осуществлять система во время взаимодействия с внешними действующими лицами (actors): людьми, компьютерными системами и процессами. Методика сценариев использования применяется для выявления функциональных требований к поведению системы и к ее элементам [6].

В соответствии с постановкой задачи предложены основные сценарии использования системы. Во всех случаях внешним действующим лицом выступает пользователь мобильного приложения.

• Пользователь добавляет вещь с электронной меткой в базу данных приложения.

• Пользователь создает новый список вещей, выбирая предметы вручную или с помощью приложения, которое сканирует окружающую среду на наличие электронных меток и отмечает соответствующие вещи в списке. Далее в любой момент пользователь может редактировать список.

• Пользователь проверяет, все ли вещи из выбранного списка он собрал.

• Приложение сканирует окружающую среду на наличие меток и отмечает соответствующие предметы как собранные.

• Исходя из описанных сценариев использования, можно выделить следующие требования к электронным меткам.

• Наличие уникального идентификатора у каждой электронной метки. Это необходимо для однозначного определения того, какие конкретные предметы пользователь уже собрал, а какие — еще нет. Этому критерию соответствуют все типы меток.

• Радиус действия не менее 20–40 см. Электронные метки с меньшим радиусом действия будет необходимо вручную подносить к смартфону для обнаружения; это создаст дополнительные трудности для пользователя. Этому критерию соответствуют все типы меток, кроме NFC.

• Время работы батареи не менее полугода. Маленький срок жизни метки означает, что электронную метку, прикрепленную к предмету, будет необходимо часто менять; это также доставляет пользователю неудобства. Этому критерию соответствуют все типы меток.

• Небольшие размеры: метки должны быть легкими и компактными. Этому критерию соответствуют все типы меток: наибольший размер имеют Wi-Fi-метки: 45 x 55 x 20 мм, и этого уже достаточно, чтобы прикрепленная электронная метка не мешала использованию предметов.

Таким образом, в рамках проекта можно использовать электронные метки трех типов: Wi-Fi, Bluetooth, RFID — они обладают большим радиусом действия и достаточно высоким сроком жизни, поэтому с ними можно реализовать все разработанные сценарии использования. Что касается последней технологии, NFC-метки имеют небольшой радиус действия и фактически для идентификации их необходимо очень близко подносить к устройству, а это неудобно для пользователя и непригодно для предлагаемых способов работы с системой.

Ссылки

1. Бюро находок: поисковый портал. URL: http://buro.nahodok.ru/ (дата обращения: 08.06.2015).

2. Международная служба возвратов. URL: http://www.europoisk.com/about/ (дата обращения: 08.06.2015).

3. Radio Frequency Identification Fundamentals and Applications Design Methods and Solutions / ed. by Cristina Turcu. InTech, 2010. 334 p.

4. Getting Started with Bluetooth Low Energy. Tools and Techniques for Low-Power Networking / Kevin Townsend, Carles Cufí, Akiba, Robert Davidson. O'Reilly Media, 2014. 180 p.

5. Vedat Coskun, Kerem Ok, Busra Ozdenizci. Professional nfc application development for Android / John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, United Kingdom, 2013. 632 p.

6. Kurt Bittner, Ian Spence. Use Case Modeling. Addison-Wesley Professional, 2003. 347 p.

УДК 519

Моделирование распределенных динамических систем при помощи асинхронных клеточных автоматов

Е. А. Левичев, В. А. Башкин

Ярославский государственный университет им. П. Г. Демидова
E-mail: ego-levichev@yandex.ru, v_bashkin@mail.ru

В статье рассматриваются и сравниваются различные типы клеточных автоматов на примере задачи моделирования поведения толпы. Цель работы — произвести сравнение моделирующих возможностей синхронного и асинхронного клеточных автоматов.

Ключевые слова: асинхронный клеточный автомат, синхронный клеточный автомат, моделирование поведения толпы в экстремальных ситуациях.

Введение

Задача моделирования поведения толпы при помощи клеточных автоматов известна давно. Для этой задачи обычно используются только синхронные КА.

Но нет достоверных сведений о том, как с этой задачей справился бы асинхронный автомат и какие видимые различия в их работе могут быть. Кроме того, использование распределенных асинхронных систем КА имеет некоторые преимущества перед применением синхронных, такие как, например, возможность использовать узлы с большой разницей в производительности.

Постановка задачи

В работе ставилась задача наглядно показать сходство либо различие моделирующих способностей синхронного и асинхронного КА. Если они окажутся схожими, то это может оказаться важным при использовании типа КА в решении других подобных задач. Особенно сильно будет ощущаться выгода от использования асинхронного КА, если моделирование будет распределяться на системе с узлами различной производительности.

Клеточный автомат

Клеточный автомат — дискретная модель, изучаемая в математике, теории вычислимости, физике, теоретической биологии и микромеханике. Включает регулярную решетку ячеек, каждая из которых может находиться в одном из конечного множества состояний, таких как 1 и 0. Решетка может быть любой размерности. Для каждой ячейки определено множество ячеек, называемых окрестностью. К примеру, окрестность может быть определена как все ячейки на расстоянии не более 2 от текущей (окрестность фон Неймана ранга 2). Для работы клеточного автомата требуется задание начального состояния всех ячеек и правил перехода ячеек из одного состояния в другое. На каждой итерации, используя правила перехода и состояния соседних ячеек, определяют новое состояние каждой ячейки. Обычно правила перехода одинаковы для всех ячеек и применяются сразу ко всей решетке.

Применение клеточных автоматов

- Алгоритмическая разрешимость задач.
- Процессоры на клеточных автоматах.
- Генерация случайных чисел.
- Моделирование систем типа «реакция — диффузия».

Моделирование поведения толпы

Одной из наиболее серьезных проблем наших дней является обеспечение безопасности людей в нештатных ситуациях.

Во многих таких ситуациях главной угрозой здоровью и жизни людей часто оказывается возникающая при этом паника.

Поведение большой группы людей в стандартной ситуации легко поддается предсказанию и хорошо описывается вероятностным образом. Здесь работает закон больших чисел: даже если один человек по каким-то причинам решит действовать нетривиально, его действия никак не повлияют на группу в целом.

Для математического моделирования динамики толпы оказалось возможным применить клеточные автоматы.

Синхронный и асинхронный клеточный автомат

Синхронный:

• один шаг автомата — это один ход каждой клетки,

• не гибок в использовании в распределенных системах.

Асинхронный:

• один шаг автомата — один или несколько ходов случайного количество клеток,

• гибок в использовании в распределенных системах.

Сравнение работы разных типов автоматов

В ходе работы автоматов ведется статистика. По оси «Ох» откладывается начальное количество живых клеток, по оси «Оу» — количество шагов автомата, до того момента, пока все живые клетки не закончатся. По результатам статистики можно судить о схожести моделирующих способностей автоматов.

Заключение

В процессе работы создана программа, наглядно показывающая процесс моделирования поведения толпы в экстремальных ситуациях с использованием синхронного и асинхронного типов клеточных автоматов. Приведена наглядная статистика их действия и сравнение показателей работы.

Целью работы было сравнить моделирующие возможности различных типов автоматов. Цель достигнута: основываясь на результатах графиков, можно заключить, что синхронный и асинхронный клеточные автоматы имеют примерно одинаковые моделирующие возможности для данной задачи.

Также с помощью более сложных пользовательских карт можно глубже исследовать моделирующие возможности разных типов автоматов.

Ссылки

1. Нейман Дж. фон. Теория самовоспроизводящихся автоматов: пер. с англ. М.: Мир, 1971.

В оригинале: von Neumann J. Theory of Self-Reproducing Automata: ed. and compl. by A. Burks. University of Illinois Press, 1966.

2. Тоффоли Т., Марголус Н. Машины клеточных автоматов. М.: Мир. 1991.

УДК 004

Моделирование и анализ свойств протокола для p2p-сетей при помощи раскрашенных сетей Петри

С. А. Максимов, Д. Ю. Чалый

Ярославский государственный университет им. П. Г. Демидова
E-mail:maksimov.s.and@gmail.com, chaly@uniyar.ac.ru

В статье объектом исследования является протокол Chord, который моделируется при помощи раскрашенных сетей Петри и среды моделирования CPN Tools. Целью работы является проведение анализа семантических свойств протокола при помощи методов, разработанных для этого формализма.

Ключевые слова: протокол Chord, раскрашенные сети Петри, CPN Tools.

Введение

Исследование свойств протоколов передачи данных является важной и актуальной задачей. В работе смоделирован и проанализирован протокол Chord [7] — протокол получения данных из распределенной хэш-таблицы (distributed hash table, DHT), задающий алгоритм поиска узла, на котором содержится искомая порция данных. Chord служит для организации эффективного доступа к узлам распределенной системы.

Моделирование является одним из основных методов исследования протоколов в коммутируемых сетях. В настоящее время разработано большое количество инструментальных средств, ориентированных на анализ моделей, представленных с помощь сетей Петри. Существует много современных инструментов, работающих с сетями Петри, примером таких средств являются ARP tool, CoopnBuilder, CPN Tools, HISIm, Petri .NET Simulator, Petri Net Toolbo1, Petruchio, Snoop2. Среди перечисленных средств большими возможностями выделяется среда моделирования CPN Tools [6], которая и была выбрана для использования. В рамках работы была поставлена задача построить модель протокола Chord в системе CPN Tools и проанализировать работу модели на примере нескольких топологий коммутируемых сетей.

Принципы работы протокола Chord

Протокол Chord [7] — алгоритм для соединения равноправных узлов в локальной сети, использующий распределенную хэш-таблицу (DHT). Принцип DHT служит для организации эффективного доступа к большому числу блоков данных, разбросанных по нескольким серверам и хранящихся на них примерно в одном виде. Каждый из таких блоков данных должен обладать некоторым уникальным ключом.

Предоставляемые DHT методы работы с данными:

- get (key)
- delete (key)
- insert (key, value)

С помощью хеш-функции (HASH()) множество ключей блоков данных отображается в кольцо Chord Ring (последний и первый элемент этого упорядоченного множества хешей считаются связанными). Узлам (серверам) присваиваются ключи (nodekey), принадлежащие множеству Chord Ring, — узлы «распределяются» по кольцу.

Моделирование архитектуры протокола

В данной статье разобрана не вся модель, а только основные ее части. Модель построена с применением иерархии. Самой общей из них является «Top». На рис. 1 изображено разбиение задачи на три подзадачи: добавление или инициализация (Add), удаление (Delete), обновление finger table (RFT) и тестирование

(Test). Finger table каждого пира представляет собой список, элементы которого — это номера узлов, связанных с данным пиром. Пир характеризуется идентификатором и finger table. В позиции «Peers» осуществляется хранение информации о пирах.

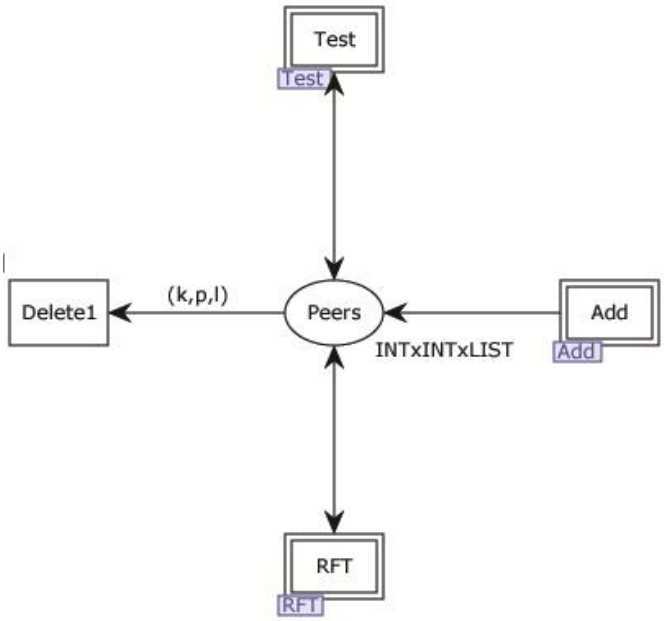


Рис. 1. Моделирование состояний системы

Моделирование обновления служебных таблиц

Второй самой важной частью модели является граф (рис. 2), в котором происходит заполнение (или обновление) finger table.

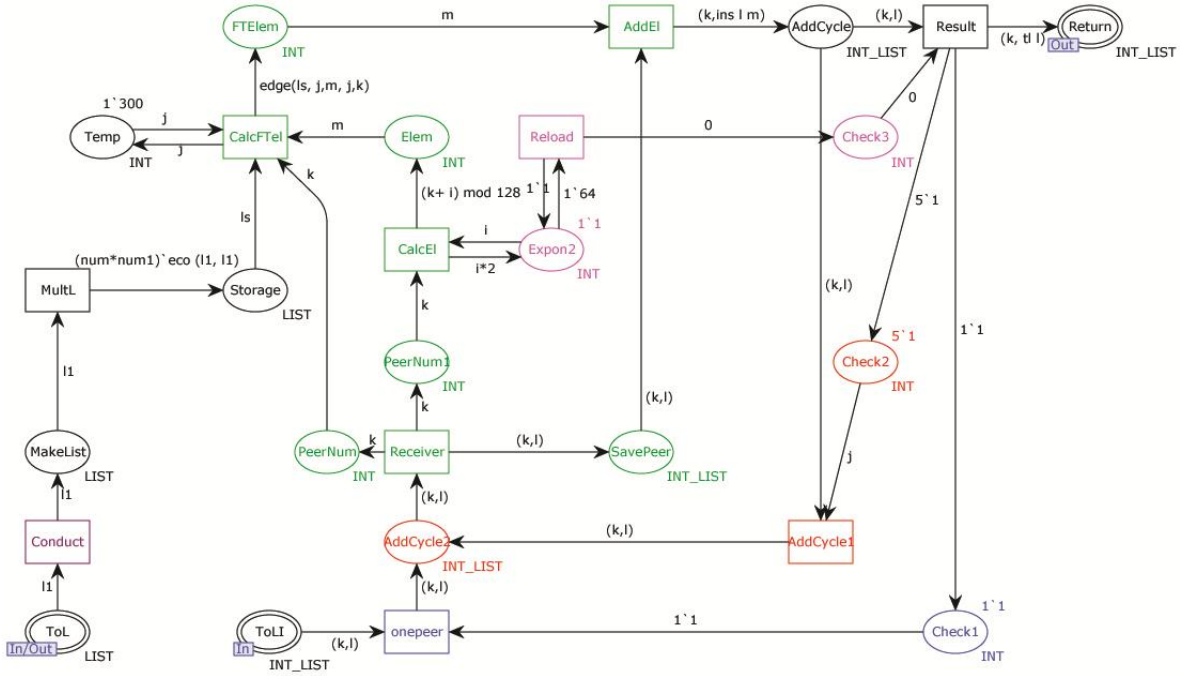


Рис. 2. Обновление таблицы

Далее будет описываться работа каждого перехода и позиции, некоторые из них работают вместе, поэтому будут объединены в группы. MultL создает столько списков всех пиров, сколько раз будет работать цикл. 1peer и Check1 «пропускают» в цикл только по одной метке. AddCycle, AddCycle1 и Check2 проверяют, чтобы таблица была полностью заполнена. CaclEl и Expon2 — цикл для расчета степеней двойки. CalcFTel при помощи рекурсивной функции edge() рассчитывает следующий элемент, который будет необходимо добавить в finger table. Когда в цикл перестанут входить данные, пиры возвращаются в Return.

Эксперименты

Рассмотрим работу модели на следующих примерах.

- Несколько примеров топологий кольца. Проверка работоспособности, модели протокола при разном количестве пиров в сети: 10, 7, 5.

- Запрос данных из системы. Запрос моделируется двумя параметрами, на каком узле он сейчас находится и куда он должен в итоге поступить.

- Динамический сценарий работы системы. Инициализация системы, удаление узла, обращение к удаленному узлу.

Рассмотрим один эксперимент. На рис. 3 приведена начальная разметка сети. Выполняется проверка работоспособности для сети из 5 пиров.

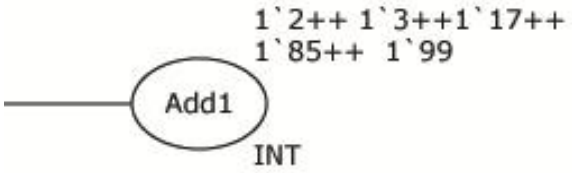


Рис. 3. Добавление фишек, соответствующих узлам

При таких входных данных исполнение модели сработало за 178 шагов. На рис. 4 виден результат работы, который показывает корректную инициализацию кольца DHT.

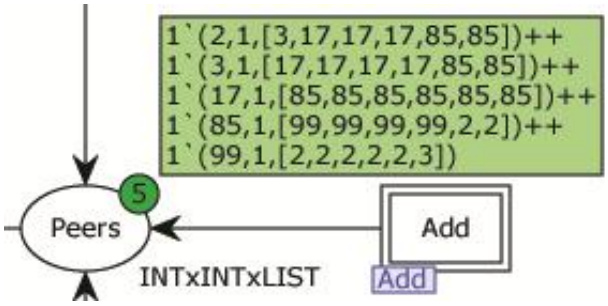


Рис. 4. Данные о пирах в DHT после выполнения сети Петри

В табл. 1 виден результат трех экспериментов, где производится выполнение модели с разными исходными данными.

Таблица 1

Количество шагов, требуемых для инициализации системы

Количество узлов	Количество шагов
5	178
7	248
10	353

Для более детального анализа проверим свойства графа достижимости. Для получения данных о графе достижимости был написан следующий программный код:

```
val q = NoOfNodes();
val w = NoOfArcs ();
val r = EntireGraphCalculated ();
val y = ListLiveTIs();
val u = ListDeadMarkings();
val i = st_Mark.Top'Peers 1 100;
UpperInteger (Mark.Top'Peers 1);
LowerInteger (Mark.Top'Peers 1);
```

Данный программный код позволяет исследовать свойства:

1. Статистический анализ графа достижимости.

- NoOfNodes() — количество вершин в графе достижимости;
- NoOfArcs()— количество дуг;
- EntireGraphCalculated() — функция проверяет, весь ли граф построен;
- st_Mark — определяет, какая разметка содержится в позиции with на странице Top 100-го узла графа достижимости.

2. Проверка графа на наличие тупиковых состояний.

- ListDeadMarkings()— возвращает список тупиковых состояний в графе достижимости. В нашей модели в любом тупиковом состоянии все пиры должны иметь обновленные DHT.

3. Проверка на наличие бесконечных циклов.

- ListLiveTIs()— выводит список бесконечных циклов в графе достижимости. Модель должна выполнить свою задачу за конечное число шагов, т. е. бесконечных циклов быть не должно.

4. Проверка ограниченности.

- UpperInteger — возвращает максимальное количество фишек в позиции with на страничке Top;

- LowerInteger — возвращает минимально количество фишек в позиции with на страничке Top. В модели должно быть минимальное количество 0, а максимальное равно количеству фишек в сети.

Построен граф достижимости для инициализации трех пиров с идентификаторами 2, 70, 120. Свойства графа достижимости показаны как результат срабатывания кода, разработанного для среды CPN Tools:

```
val q = 14180 : int
val w = 29441 : int
val r = true : bool
val y = [] : TI.TransInst list
val u = [14178,14177] : Node list
val i =
 "Top'Peers 1: 1`(2,1,[70,70,70,70,70,70])++\n1`(70,1,[120,120,120,120,12#"
 : string
val i =
 "Top'Peers 1: 1`(2,1,[70,70,70,70,70,70])++\n1`(70,1,[120,120,120,120,12#"
 : string
val it = 3 : int
val it = 0 : int
```

Как видим, всего вершин в графе 14 180, дуг — 29 441. Значение переменной r означает, что граф достижимости построен полностью. Если в переменной y хранится пустой список — в графе нет бесконечного цикла, что подтверждает корректность протокола, т. к. процесс инициализации должен проходить за конечное время. В переменную u записываются вершины, которые являются тупиковыми состояниями. Следующие 2 переменные как раз и проверяют, какие метки находятся в состояниях 14 178 и 14 177. Последние 2 переменные указывают на ограничение позиции Peers, которое равно от 0 до 3.

Заключение

В работе рассмотрены вопросы моделирования и анализа протокола Chord с помощью раскрашенных сетей Петри, описана работающая модель и модификации к ней. Добавлена возможность мониторинга выполнения запроса. Запросом является построение маршрута от одного узла к другому.

Эксперименты, проведенные над моделью протокола Chord, показали корректность протокола. А именно, при об-

ращении к только, что удаленному узлу ссылка на данный узел не будет получена.

Предложены подходы к анализу полученной модели. В дальнейшем предполагается проведение более детального анализа производительности протокола Chord.

Ссылки

1. Зайцев Д. А., Шмелева Т. Р. Моделирование телекоммуникационных систем в CPN Tools: учебное пособие по курсу «Математическое моделирование информационных систем». Одесса, 2008. 68 с.

2. Козюра В. Е., Непомнящий В. А., Новиков Р. М. Верификация раскрашенных сетей Петри методом проверки моделей: учебное пособие. Новосибирск, 2001. 26 с.

3. Башкин В. А. Функциональное программирование на языке SML: методические указания. Ярославль: ЯрГУ, 2007. 39 с.

4. Kulik J., Rabiner W., Balakrishnan H. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks // International Conference on Mobile Computing and Networking. Seattle, 1999.

5. Allavena A., Demers A., Hopcroft J. Correctness of a Gossip-based Membership Protocol // Symposium on Principles of Distributed Computing (PODC). Las Vegas, Nevada, 2005. P. 292–103.

6. Chaly D., Sokolov V. An Extensible Coloured Petri Net Model of a Transport Protocol for Packet Switched Networks // Lecture Notes in Computer Science Volume 2763, 2003. P. 66–75.

7. Chord: A scalable peer-to-peer lookup service for internet applications / I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan // ACM SIGCOMM Computer Communication Review. 2001.

8. Документация системы CPN Tools. URL: http://www.cpntools.org (дата обращения: 23.06.2015).

УДК 004.422

Разработка мобильного гида для города Ярославля

Е. О. Максимычев, Н. С. Лагутина, Д. Е. Озерова

Ярославский государственный университет им. П. Г. Демидова
E-mail: makevg93@mail.ru, lagutinans@rambler.ru, deozero@mail.ru

В статье описывается мобильный гид по г. Ярославлю, позволяющий осуществлять выбор интересующей экскурсии из списка доступных экскурсий, просматривать информацию о каждой точке экскурсии, прокладывать маршрут до нее. Приложение определяет местоположение человека и автоматически включает звуковое сопровождение точки экскурсии, рядом с которой находится пользователь. Мобильный гид включает в себя экскурсии, составленные сотрудником Музея истории города Ярославля и студентами ЯрГУ им. П. Г. Демидова.

Ключевые слова: мобильное приложение, мобильный гид, клиент-серверное приложение, геолокационные данные.

Историю использования компьютеров в музейной деятельности можно отсчитывать с 1960-х гг., когда музейные специалисты за рубежом задумались о необходимости совершенствовать технологию работы с данными о музейных коллекциях. Позднее возможности компьютера стали использовать не только в фондовой работе, но и на музейных экспозициях и в других областях. Например, стали создаваться электронные киоски с сенсорными дисплеями. За небольшую плату посетитель итальянских музеев уже в 1980-е гг. мог получить доступ к дополнительным сведениям о музее, просматривать изображения экспонатов, даже тех, которые находятся в запасниках, знакомиться с другими музеями города или страны. В Японии в Музее изящных искусств в 1980-х гг. была открыта специальная галерея, где на телеэкране можно было увидеть экспонаты из запасников, а также из крупнейших музеев мира. Были представлены даже около ста полотен из российского Эрмитажа.

Компьютеризация музеев нашей страны началась примерно на 15 лет позже, чем в европейских станах. В отечественном музееведении вопрос об использовании компьютера впервые подняли сотрудники Государственного Эрмитажа. В 1996 г. по инициативе трех крупнейших московских музеев: Московского Кремля, Пушкинского и Третьяковской галереи — была учреждена Ассоциация по документации и новым информационным технологиям в музеях (АДИТ). Вскоре интерес к применению компьютеров для хранения данных о коллекциях стали проявлять не только крупные столичные музеи, но и небольшие региональные. Вслед за этим компьютеризированные технические средства как вспомогательный материал стали использоваться на экспозициях и выставках. Появились аудиогиды.

Аудиогид — это персональный электронный экскурсовод, используемый для самостоятельного знакомства с экспозицией музея, выставки или местностью. Обычно аудиогид состоит из нескольких аудиофрагментов. Фрагменты нумеруются и «привязываются» к схеме (карте) осматриваемой части музея или к номерам экспонатов. Если аудиогид предполагает связный и законченный рассказ из фрагментов, то его также называют аудиоэкскурсией. Некоторые модели аудиогидов оснащены дисплеем, на котором отображается номер воспроизводимого файла, громкость и другая дополнительная информация. Более современные аудиогиды могут воспроизводить видео и отображать фотографии на дисплее. В музеях аудиогиды предлагаются в виде специального устройства, представляющего собой антивандальный mp3-плеер. Все чаще появляются аудиогиды второго поколения, основанные на карманном персональном компьютере (КПК). В этом случае они позволяют не только прослушивать фонограмму, но и получать на экране дополнительную текстографическую информацию.

В настоящее время в магазинах мобильных приложений, в частности для платформы Windows Phone, представлены аудиогиды, но большинство из них являются мобильными гидами для зарубежных городов и стран, например для Нью-Йорка[1]. Для г. Ярославля ни одного подобного приложения не реализо-

[1] New York City Guide. URL: https://www.windowsphone.com/ru-ru/store/app/new-york-city-guide/c9be644b-5f6d-4428-a897-fd43d310af30

вано. Аналогичным российским приложением является гид по культурным местам Ульяновска[1].

Поэтому актуальной задачей является разработка мобильного гида для Ярославля, который авторы назвали «City Guide». Данное приложение будет полезно для людей, которые приехали из других городов или стран и хотят больше узнать о достопримечательностях города. Мобильный гид могут использовать и жители города, которые хотели бы подробнее ознакомиться с историей родного края.

Приложение обладает следующей функциональностью:

• возможностью выбора определенной экскурсии,

• отображением экскурсии на карте,

• возможностью выбрать интересующую точку экскурсии,

• отображением информации о точке экскурсии,

• определением текущего местоположения пользователя,

• воспроизведением аудиоинформации в зависимости от местонахождения пользователя,

• возможностью проложить маршрут до любой точки экскурсии.

Разработка приложения «City Guide» велась под мобильную платформу Windows Phone. Платформа Windows Phone — это молодая, развивающаяся мобильная операционная система, адаптированная под планшеты и смартфоны, которая в данный момент стала достаточно популярной.

Архитектура приложения состоит из трех звеньев. Первым звеном является клиентское приложение. Вторым — серверное приложение. А третьим выступает сервер базы данных, в качестве которого используется MS SQL Server Express [1].

Для реализации клиентской части необходимо решить следующие задачи: разработать систему хранения данных об экскурсиях, разработать удобный и понятный пользовательский интерфейс, реализовать взаимодействие с картами Bing и со службами геолокации, разработать модель данных, обрабатывающую принятую информацию от сервера. Серверная часть

[1] Гид по культурным местам Ульяновска. URL: https://www.windowsphone.com/ru-ru/store/app/гид-по-культурным-местам-ульяновска/fc439f6f-7464-4e09-9d5a-f12c2171ac

приложения обладает следующей функциональностью: хранит данные об экскурсиях (текстовая информация, изображения, аудиозаписи), обрабатывает запрос и передает требуемую информацию клиенту.

Каждая экскурсия состоит из названия, номера и списка точек. А каждая точка экскурсии состоит из названия, номера, координат широты и долготы, описания, изображения и аудиозаписи. Для хранения информации об экскурсиях был выбран формат Xml, как один из широко распространенных способов хранения данных. Xml-формат файла для хранения экскурсий включает в себя следующие элементы: тег Excursions — контейнер, содержащий набор экскурсий; тег Excursion — описывает отдельную экскурсию; тег Point — описывает точку экскурсии.

На стороне клиента важным элементом является пользовательский интерфейс. Интерфейс пользователя (UI) — это часть программы, которая отвечает за отображение данных на экране устройства и диалог с пользователем. Интерфейс пользователя непосредственно связан со сценариями использования (UseCase). UseCase — описание последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей или других программных систем [2]. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя.

В процессе работы были предложены и реализованы следующие сценарии использования:

- Выбор локальной экскурсии.
- Выбор онлайн-экскурсии.
- Просмотр информации о точке экскурсии.
- Прокладывание маршрута.
- Определение местоположения пользователя.
- Просмотр информации о приложении.

Каждый сценарий подробно описывает отдельную функцию приложения. Например, выбор локальной экскурсии состоит из следующих шагов.

1. Пользователь находится на главном экране приложения на секции «Экскурсии».

2. Приложение загружает список экскурсий из локального файла.

3.Пользователь выбирает интересующую его экскурсию из списка нажатием на нее.

4. Открывается карта с выбранной экскурсией и точками, входящими в нее.

Во время экскурсии приложение работает с картами Bing Maps (картографический сервис Microsoft) и геолокационными данными пользователя и функционирует в зависимости от его местонахождения. Мобильный гид автоматически с помощью служб геолокации определяет координаты местонахождения пользователя и, если человек находится вблизи очередной точки экскурсии, включает воспроизведение аудиофайла, а также отображает текстовую и другую необходимую информацию. Для отображения точек экскурсии на карте и взаимодействия со службами геолокации используются пространства имен, входящие в библиотеку Windows Phone SDK: Windows.Devices.Geolocation; Windows.UI.Xaml.Navigation; Windows.Services. Maps; Windows.UI.Xaml.Controls.Maps [3].

Серверная часть используется главным образом для хранения данных о большом количестве доступных экскурсий. Для получения списка этих экскурсий клиент отправляет запрос на сервер. Ответ приходит в формате JSON — текстовом формате обмена данными, основанном на языке JavaScript. Для обработки клиентом полученных данных в формате JSON от серверного приложения использовалась библиотека Newtonsoft.Json. Данная библиотека является сегодня самой популярной для работы с данными в формате JSON на языке C#.

Таким образом, реализованное мобильное приложение «City Guide» хранит информацию о нескольких экскурсиях по г. Ярославлю и позволяет осуществлять выбор из этих экскурсий. Основными функциями приложения являются выбор экскурсии, отображение маршрута из точек экскурсии на карте города, переход между отдельными точками экскурсии в произвольном порядке, воспроизведение аудио- и текстовой информации о точках экскурсии, демонстрация соответствующего изображения.

Приложение «City Guide» тестировалось на эмуляторе Windows Phone, которое является частью Windows Phone SDK, и на смартфоне Lumia 930.

Ссылки

1. Фримен Э., Сьерра К. Паттерны проектирования. СПб.: Питер, 2011. 656 с.
2. Russell J. Use Case Points. М.: Книга по требованию, 2013. 80 с.
3. Nagel C. Professional C# 5.0 and .NET 4.5.1. Wiley, 2014. 1560 с.

УДК 004.054

Автоматизация тестирования Android-приложений

А. О. Мартынов

Ярославский государственный университет им. П. Г. Демидова
E-mail: andremartynov@gmail.com

В статье рассматривается процесс создания автоматических тестов для Android-приложений на примере фитнес-приложения Walky Doggy, описываются сценарии тестирования и приводятся результаты работы автоматических тестов.

Ключевые слова: тестирование, Android-приложение, сценарии тестирования.

Тестирование является неотъемлемой частью жизненного цикла разработки программного обеспечения. Это комплексный процесс, преследующий такие цели, как выявление ошибок в работе программы, проверка соответствия качества продукта заявленным требованиям, предотвращение появления дефектов и другие. В общем случае тестирование заключается в планировании, разработке и реализации тестов, а также поддержании их в актуальном состоянии [1].

В больших проектах ручное тестирование очередного варианта может занимать до одной недели.

Автоматизация помогает сократить время тестирования и упростить его процесс, используя программные средства для выполнения тестов и проверки результатов выполнения.

Наиболее распространенной формой автоматизации является тестирование приложений через графический пользовательский интерфейс [2].

Цель данной работы — разработка сценариев тестирования и внедрение автоматических тестов на этапе активной разработки Android-приложения Walky Doggy.

Разработка приложения Walky Doggy велась под мобильную платформу Android. Платформа Android — операционная система для смартфонов, планшетных компьютеров, телевизоров и других устройств. Операционная система основана на ядре Linux и собственной реализации виртуальной машины Java от компании Google.

Walky Doggy от компании FRUCT — это специальное фитнес-приложение, стимулирующее активность пользователя в течение дня. Приложение напоминает пользователю о необходимости выйти на прогулку, имитируя поведение собаки. Главная цель приложения — мотивировать пользователя выходить на прогулку каждый день в одно и то же время. Данное приложение позволяет установить длительность и время прогулок, отследить количество пройденных шагов.

Тестирование является важной составляющей процесса разработки приложения. Для Android тестирование особенно важно, т. к. устройства сильно отличаются друг от друга: размером и разрешением экрана, версией операционной системы, форм-фактором, системой команд процессора, наличием фронтальной камеры, NFC, внешней клавиатуры и т. д. [3].

Тестирование приложений имеет две основные цели: демонстрацию работоспособности приложения и выявление ситуаций, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации. Задача автоматического тестирования — с наибольшей точностью автоматизировать действия, которые выполняет тестировщик.

Для тестирования Android-приложений можно использовать несколько популярных библиотек:

Sikuli Script — открытая кросс-платформенная визуальная среда создания сценариев-скриптов, которая ориентирована на программирование графического интерфейса при помощи изображений (скриншотов). В качестве скриптового языка в Sikuli используется Jython;

Robolectric — открытая кросс-платформенная библиотека для создания Unit-тестов. Запуск тестов происходит на виртуальной Java-машине компьютера, что ускоряет процесс тестирования;

UIAutomator — это аналог инструмента UIAutomation компании Apple для тестирования Android-приложений. Данный инструмент позволяет работать напрямую с элементами пользовательского интерфейса. Таким образом не приходится привязываться к координатам элементов.

Robotium — это Android-библиотека от компании Google для автоматизации тестирования приложений, которая имеет полную поддержку нативных и гибридных приложений. Robotium позволяет легко реализовать мощные и надежные автоматические тесты для Android-приложений. При помощи Robotium разработчик может реализовать тестовые сценарии, охватывающие несколько активностей сразу.

В ходе разработки тестов автором была выбрана библиотека Robotium по ряду причин:

- отсутствие зависимости от разрешения и ориентации экрана,
- прямая работа с элементами пользовательского интерфейса,
- поддержка широкого спектра устройств,
- поддержка эмуляторов.

Первый этап тестирования — разработка сценариев. Сценарии тестирования детально описывают то, как пользователи решают конкретные задачи в определенном контексте приложения. Сценарии используются как примеры поведения пользователей, которые учитываются при разработке системы, а потом берутся за основу при проведении тестов.

Были разработаны следующие сценарии тестирования:

- Проверка корректности срабатывания напоминаний.
- Проверка корректности изменения времени прогулки.
- Проверка корректности изменения длительности прогулки.
- Проверка уникальности прогулок.
- Проверка минимальной длительности прогулки.

• Проверка процесса прогулки.

• Проверка минимального количества прогулок.

• Проверка ручного завершения прогулки.

В качестве примера рассмотрим описание сценария для проверки корректности срабатывания напоминаний. Он состоит из следующих шагов:

1. Запускается приложение.
2. Отображается главный экран.
3. Пользователь нажимает на кнопку «Настройки».
4. Отображается экран настроек.
5. Пользователь нажимает на время прогулки.
6. Отображается диалог установки времени прогулки.
7. Устанавливается время прогулки.
8. Пользователь нажимает на кнопку «Готово».
9. Отображается экран настроек с заданным временем прогулки.
10. Пользователь нажимает на кнопку «back».
11. Отображается главный экран.
12. В заданное время отображается экран с уведомлением о прогулке.

Далее необходимо разработать классы тестирования. В данном случае были написаны два класса. Класс WalkScreenChangeTest содержит сценарии, отвечающие за тестирование главного окна приложения. Класс SettingsScreenTest содержит сценарии, отвечающие за тестирование окна настроек.

После этого для каждого теста и каждого устройства выполняются следующие шаги:

1) установка приложения на устройство,

2) запуск приложения,

3) тестирование приложения с помощью разработанных классов,

4) удаление приложения,

5) сброс состояния устройства.

На каждом шаге нужно собрать и проанализировать данные. Затем на основе этих данных сформировать результат тестирования.

В ходе тестирования были выявлены следующие ошибки.

1. Неправильная работа уведомлений на устройствах с Android 4.2.1.

2. После удаления будильника из списка уведомление продолжало приходить.

3. Неверное поведение приложения при удалении будильника.

4. Неверное поведение приложения при добавлении уже существующего будильника.

Найденные ошибки были исправлены, приложение было опубликовано в магазине Google Play и доступно по адресу: https://play.google.com/store/apps/details?id=org.fruct.yar.walkydoggy.

Данное приложение будет полезно людям с малоактивным образом жизни и пожилым пользователям.

Ссылки

1. Тамре Л. Введение в тестирование программного обеспечения; пер. с англ. М.: Издательский дом Вильямс, 2003. 368 с.

2. Котляров В. П. Основы тестирования программного обеспечения: курс лекций : учеб. пособие для студентов вузов, обучающихся по специальностям в области информ. технологий. М.: Интернет-ун-т информ. технологий, 2006. 288 с.

3. Milano D. T. Android application testing guide. Packt Publishing Ltd., 2011. 332 с.

УДК 519.854

О нецелочисленных гранях релаксационных многогранников задачи булева квадратичного программирования

А. А. Миронычев, А. В. Николаев

Ярославский государственный университет им. П. Г. Демидова
E-mail: caiiika-76@yandex.ru, werdan.nik@gmail.com

Рассматривается задача распознавания целочисленности на релаксациях булева квадратичного многогранника и приводится алгоритм проверки принадлежности точек полностью

нецелочисленным граням релаксации $M_{n,k}$. Существование подобных точек препятствуют эффективному решению задачи методами линейного программирования.

Ключевые слова: распознавание целочисленности, метрический многогранник, нецелочисленные вершины.

Булев квадратичный многогранник BQP_n определяется как выпуклая оболочка точек

$$x_{i,i} + x_{j,j} - x_{i,j} \le 1, \quad (1)$$

$$x_{i,j} \le x_{i,i}, \quad (2)$$

$$x_{i,j} \le x_{j,j}, \quad (3)$$

$$x_{i,j} \ge 0, \quad (4)$$

$$x_{i,j} \in \{0,1\}, \quad (5)$$

для всех $i, j : 1 \le i \le j \le n$ [1].

К задаче линейного программирования на BQP_n сводятся, в частности, задачи булева квадратичного программирования и максимальный разрез. Однако число гиперграней BQP_n растет сверхэкспоненциально быстро и полное описание известно лишь для $n \le 8$, что препятствует непосредственному решению задачи линейного программирования на BQP_n. Так, многогранник BQP_5 задается системой из 364 неравенств, BQP_6 системой из 116 764 неравенств, а последний известный на данный момент BQP_8 — системой из порядка 10^{13} неравенств [2].

Если исключить из системы (1)–(5) условие целочисленности переменных (5), то оставшиеся ограничения описывают релаксационный многогранник M_n, известный как корневой полуметрический.

Определим последовательность $M_{n,k}$ вложенных релаксаций булева квадратичного многогранника:

$$BQP_n = M_{n,n} \subseteq M_{n,n-1} \subseteq \ldots \subseteq M_{n,k} \subseteq \ldots M_{n,3} \subseteq M_{n,2} = M_n,$$

где многогранник $M_{n,k}$ получается дополнением системы (1)–(4) ограничениями булева квадратичного многогранника BQP_k по каждому подмножеству $\{i_1,\ldots,i_k\}$ множества $\{1,\ldots,n\}$ индексов координат [3; 4].

В частности, многогранник $M_{n,3}$, известный как метрический, представляет значительный интерес, т. к. является компактной формулировкой задачи о максимальной разрезе в графе, не содержащем K_5 в форме линейного программирования [5].

На релаксациях $M_{n,k}$ рассмотрим задачу распознавания целочисленности, отвечающую на вопрос: содержит ли грань многогранника, на которой линейная целевая функция достигает своего максимума, хотя бы одну целую вершину.

Одним из важных свойств метрического многогранника является [3]

Утверждение 1. *Неравенства треугольника* $M_{n,3}$ *отсекают все грани корневого полуметрического многогранника* M_n*, содержащие только нецелочисленного вершины.*

Как следствие, если для некоторой линейной целевой функции $f(x)$

$$\max_{x\in M_n} f(x) = \max_{x\in M_{n,3}} f(x),$$

то максимум достигается в целой вершине M_n. В противном случае

$$\max_{x\in M_n} f(x) > \max_{x\in M_{n,3}} f(x),$$

и $f(x)$ достигает максимума на грани, содержащей только нецелочисленные вершины.

Таким образом, задача распознавания целочисленности на корневом полуметрическом многограннике M_n полиномиально разрешима.

Аналогично можно рассмотреть алгоритм решения задачи распознавания целочисленности на метрическом многограннике. Достаточно сравнить значения целевой функции на $M_{n,3}$ и неко-

торой релаксации $M_{n,k}$, ограничения которой отсекают все полностью нецелочисленные грани $M_{n,3}$. Отметим, что, несмотря на сложность построения полного описания $M_{n,k}$, многогранник определяется системой с полиномиальным по n размером, который не превосходит C_n^k на число неравенств в описании BQP_k.

Соответственно, если для некоторого фиксированного k подобная релаксация $M_{n,k}$ существует, то рассматриваемый алгоритм был бы полиномиальным алгоритмом решения задачи распознавания целочисленности на метрическом многограннике. Известно, что эта задача NP-полна, в частности к ней сводится задача 3-выполнимость при различных литералах [6].

В работах [4; 6] на основании полного внешнего описания соответствующих многогранников было установлено, что релаксаций $M_{n,4}$ и $M_{n,5}$ недостаточно для отсечения всех нецелочисленных граней метрического многогранника. Однако этот метод не подходит для произвольной релаксации $M_{n,k}$, т. к. ограничения BQP_k слишком сложны и не известны для $k > 9$. Аналогично не известны и вершины многогранника $M_{n,k}$, кроме целочисленных.

Таким образом, возникает проблема проверки принадлежности точки многограннику $M_{n,k}$, ни грани ни вершины которого не известны.

Опишем метод проверки, не требующий построения полного описания BQP_k. Далее для наглядности рассмотрим решение задачи на примере для $M_{4,3}$. Точку u представим в виде верхней треугольной матрицы, каждый из элементов которой лежит в промежутке $[0, 1]$.

$$u = \begin{matrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ & u_{2,2} & u_{2,3} & u_{2,4} \\ & & u_{3,3} & u_{3,4} \\ & & & u_{4,4} \end{matrix}$$

Далее будем строить проекции u_kточки u на $k(k+1)/2$ — мерные пространства и проверять, можно ли разложить проекцию u_kв выпуклую комбинацию вершин BQP_k:

$$Ax = u_k^*,$$

$$\sum_{j=1}^{k} x_j = 1,$$

$$x \geq 0,$$

где в матрицу A по столбцам записаны координаты вершин BQP_k.

Матрица A в этой системе, имеет размер

$$k(k+1)/2 \times 2^k$$

Для нашего примера построение матрицы A будет выглядеть следующим образом:

вершины BQP_k представим в виде матриц $A_{a_1,a_2,\ldots,a_k}$ ($a_1,\ldots,a_k$ — диагональные элементы), заполняя их в соответствии с системой (1)–(5).

$$A_{a_1,a_2,a_3} = \begin{matrix} a_1 & a_{1,2} & a_{1,3} \\ 0 & a_2 & a_{2,3} \\ 0 & 0 & a_3 \end{matrix}$$

или

$$A_{a_1,a_2,a_3} = \begin{matrix} a_1 & a_1a_2 & a_1a_3 \\ 0 & a_2 & a_2a_3 \\ 0 & 0 & a_3 \end{matrix}$$

На каждом шаге матрица задается новым набором диагональных элементов, $a_i \in \{0,1\}, \forall i$

Далее элементы матрицы записываются в j-й столбец матрицы A в следующем порядке: двигаемся по i-й строке матрицы $A_{a_1,a_2,\ldots,a_k}$, дописывая элементы i-й строки начиная с диагонального в строку s; полученную строку длины $k(k+1)/2$ транспонируем и включаем в матрицу A.

$$s = \left(a_1, a_{1,2}, a_{1,3}, a_2, a_{2,3}, a_3\right)^T$$

Иначе говоря, по столбцам записываются координаты целых вершин BQP_k. Всего таких вершин и столбцов, соответственно, 2^k.

Для $M_{n,3}$ матрица A имеет следующий вид:

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix}$$

Проекция u_k — главная подматрица размерности k. Таких подматриц $C_{n,k}$. $u_k = U_{i_1,\dots,i_k}$ записывается в столбец u^* по тем же правилам, что и столбцы матрицы A.

Если существует вектор $x = (x_1, \dots, x_{k(k+1)/2})$ такой, что система

$$Ax = u_k^*,$$

$$\sum_{j=1}^{k} x_j = 1,$$

$$x \geq 0,$$

разрешима, тогда проекция u_k раскладывается в выпуклую комбинацию вершин BQP_k. Если система разрешима для **каждого** u^*, тогда точка u принадлежит $M_{n,k}$.

Пример. Рассмотрим решение системы для проекции точки u и $M_{4,3}$:

$$u = \begin{matrix} 1/2 & 1/3 & 1/6 & 1/3 \\ & 1/2 & 1/3 & 1/6 \\ & & 2/3 & 1/3 \\ & & & 1/2 \end{matrix}$$

Матрица A для $M_{4,3}$ выглядит следующим образом:

$$A = \begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix}$$

Возьмем главную подматрицу $U_{1,3,4}$:

$$u_k^* = \begin{matrix} 1/2 & 1/6 & 1/3 \\ & 2/3 & 1/3 \\ & & 1/2 \end{matrix}$$

Требуется решить систему:

$$Ax = u_k^*,$$

$$\sum_{j=1}^{k} x_j = 1,$$

$$x \geq 0,$$

Частное решение этой системы:

$$x^* = \left(0,0,\frac{1}{3},\frac{1}{6},\frac{1}{6},\frac{1}{6},0,\frac{1}{6}\right)$$

Если такие решения найдутся для каждого u_k^*, точка $u \in M_{4,3}$.

Отметим, что описанный метод не требует построения выпуклой оболочки BQP_k и требует решения не более чем $C_{n,k}$ СЛАУ (ранее было отмечено, что число гиперграней растет экспоненциально быстро), что в значительной степени ускоряет проверку принадлежности релаксации и делает возможным проведение этой проверки для $n > 8$.

На основании этого метода и приложения lp_solve [7] для решения задачи линейного программирования было доказано, что точка из работы [8] удовлетворяет ограничениям многогранника $M_{n,6}$.

Теорема 1. *Для любого* $n \geq 195$ *найдутся точки многогранника* $M_{n,6}$ *в любом разложении которых в выпуклую комбинацию вершин метрического многогранника* $M_{n,3}$ *нет ни одной целой.*

Таким образом, ограничений $M_{n,6}$ также недостаточно для решения задачи распознавания целочисленности на метрическом многограннике. Кроме того, было установлено, что все описанные в [4; 8] точки не принадлежат следующей релаксации

$M_{n,7}$, что оставляет открытым вопрос об отсечении нецелочисленных граней $M_{n,3}$ произвольной релаксацией $M_{n,k}$ [1].

Ссылки

1. Padberg M. V. The boolean quadric polytope: some characteristics, facets and relatives // Mathematical Programming. 1989. Vol. 45. P. 139–172.

2. Деза М. М., Лоран М. Геометрия разрезов и метрик. М.: МЦНМО, 2001. 736 с.

3. Бондаренко В. А., Урываев Б. В. Об одной задаче целочисленной оптимизации // Автоматика и телемеханика. 2007. № 6. С.18–23.

4. Об одной задаче распознавания на релаксациях разрезного многогранника / В. А. Бондаренко, А. В. Николаев, М. Э. Сыманович, Р. О. Шемякин // Автоматика и телемеханика. 2014. № 9. С. 108–121.

5. Barahona F. On cuts and matchings in planar graphs // Mathematical Programming. 1993. Vol. 60. P. 53–68.

6. Бондаренко В. А., Николаев А. В. Об одном классе гиперграфов и о вершинах релаксаций разрезного многогранника // Доклады академии наук. 2012. Т. 442. № 3. С. 300–302.

7. Berkelaar M., Eikland K., Notebaert P. lp_solve 5.5, Open source (Mixed-Integer) Linear Programming system. URL: http://lpsolve.sourceforge.net/5.5/

8. Николаев А. В. Гиперграфы специального вида и анализ свойств релаксаций разрезного многогранника // Моделирование и анализ информационных систем. 2011. Т. 18. № 3. С. 82–1.

[1] Работа выполнена при поддержке гранта РФФИ № 14-01-00333 и гранта Президента Российской Федерации МК-5400.2015.1.

УДК 004

Сетевое web-приложение для мониторинга и анализа выполнения норм рабочего времени локомотивными бригадами ОАО «РЖД» в грузовом движении

Д. В. Опарин

Ярославский государственный университет им. П. Г. Демидова
E-mail: mpdima90@gmail.com

В статье рассматривается сетевое web-приложение, которое используется для построения анализа выполнения локомотивными бригадами ОАО «РЖД» норм рабочего времени в грузовом движении, а также учета потерь от их невыполнения.

Ключевые слова: автоматизированная система, сетевое приложение, формирование аналитической отчетности, хранимая процедура, SAS-портал, СУБД DB2 ЦОММ, локомотивные бригады, маршрут машиниста, анализ выполнения норм, УТО-9, непроизводительные затраты.

Постановка задачи

Рассматриваемая в данной статье проблема заключается в отсутствии единого инструмента для анализа выполнения локомотивными бригадами норм рабочего времени в грузовом движении, а также учета потерь от их невыполнения.

Для получения аналитических данных по работникам локомотивных бригад сотрудникам необходимо было использовать множество вспомогательных программ, а также выполнять расчет оценки времени вручную.

Решение проблемы. Описание приложения

Разработанное сетевое web-приложение выступает единым универсальным инструментом для построения на инфраструктуре ОАО «РЖД» отчетности о выполнении локомотивными бригадами Дирекции тяги — филиала ОАО «РЖД» — норм рабочего времени («пробежных норм») по элементам затрат в грузовом движении, а также учета потерь от их невыполнения.

Программное обеспечение блока «Пробежные нормы» включает в себя модуль ввода пробежных норм и модуль формирования отчета о выполнении локомотивными бригадами норм рабочего времени в грузовом движении. Модуль ввода позволяет вводить нормы по элементам затрат рабочего времени по участкам работы локомотивных бригад. Аналитический отчет предназначен для определения потерь рабочего времени по элементам затрат по участкам обслуживания, локомотивным эксплуатационным депо, территориальным дирекциям тяги и Дирекции тяги в целом относительно введенных норм.

Элементы затрат включают время:

- от явки до начала приемки ТПС (при выезде из депо);
- от начала приемки ТПС до окончания приемки ТПС (при выезде из депо);
- от окончания приемки ТПС до прохода КП (при выезде из депо);
- от явки до прохода КП (при выезде из депо);
- от прохода КП до отправления поезда (при выезде из депо);
- от явки до приемки ТПС на станции;
- от начала приемки ТПС на станции до окончания приемки;
- от окончания приемки ТПС на станции до отправления;
- все накладное время при отправлении;
- время в движении (время от отправления с начальной станции участка до прибытия на конечную станцию участка, без учета времени стоянок на промежуточных станциях);
- стоянки на промежуточных станциях;
- от прибытия до КП (при заезде в депо);
- от прохода КП до начала сдачи ТПС (при заезде в депо);
- от начала сдачи ТПС до окончания сдачи (при заезде в депо);
- от окончания сдачи ТПС до окончания работы (при заезде в депо);
- от прибытия до начала сдачи ТПС (при сдаче на станции);
- от начала сдачи ТПС до окончания сдачи (при сдаче на станции);
- от окончания сдачи ТПС до окончания работы (при сдаче на станции);
- все накладное время по прибытии;

- оборот по станции (время от прибытия на станцию до отправления со станции без отдыха в оборотном депо).

- итого за поездку;

- всего за поездку в оба конца с отдыхом в ПО;

- всего за поездку в оба конца с оборота.

В отчете ф. УТО-9 по элементам затрат рабочего времени отображаются следующие показатели:

- график отчетного периода;

- норма отчетного периода;

- факт отчетного периода;

- факт аналогичного периода прошлого года;

- % выполнения графика;

- % выполнения нормы;

- разница к графику;

- разница к нормативу;

- разница факта текущего и прошлого года;

- количество случаев отчетного периода;

- количество случаев прошлого года;

- графиковое время;

- нормативное время;

- фактическое отработанное;

- потери времени к графику;

- потери времени к нормативу;

- потери человек к графику;

- потери человек к нормативу.

На основе отчетных данных ф. УТО-9 строится аналитический отчет по непроизводительным затратам от превышения установленных нормативов оборота локомотивной бригады за отчетный период. Дополнительно в форме предусмотрена возможность просмотра фактических данных отдельно по маршрутам с разбивкой по элементам затрат.

Общая информация об объекте автоматизации

Общая схема функционирования системы приведена на рис. 1.

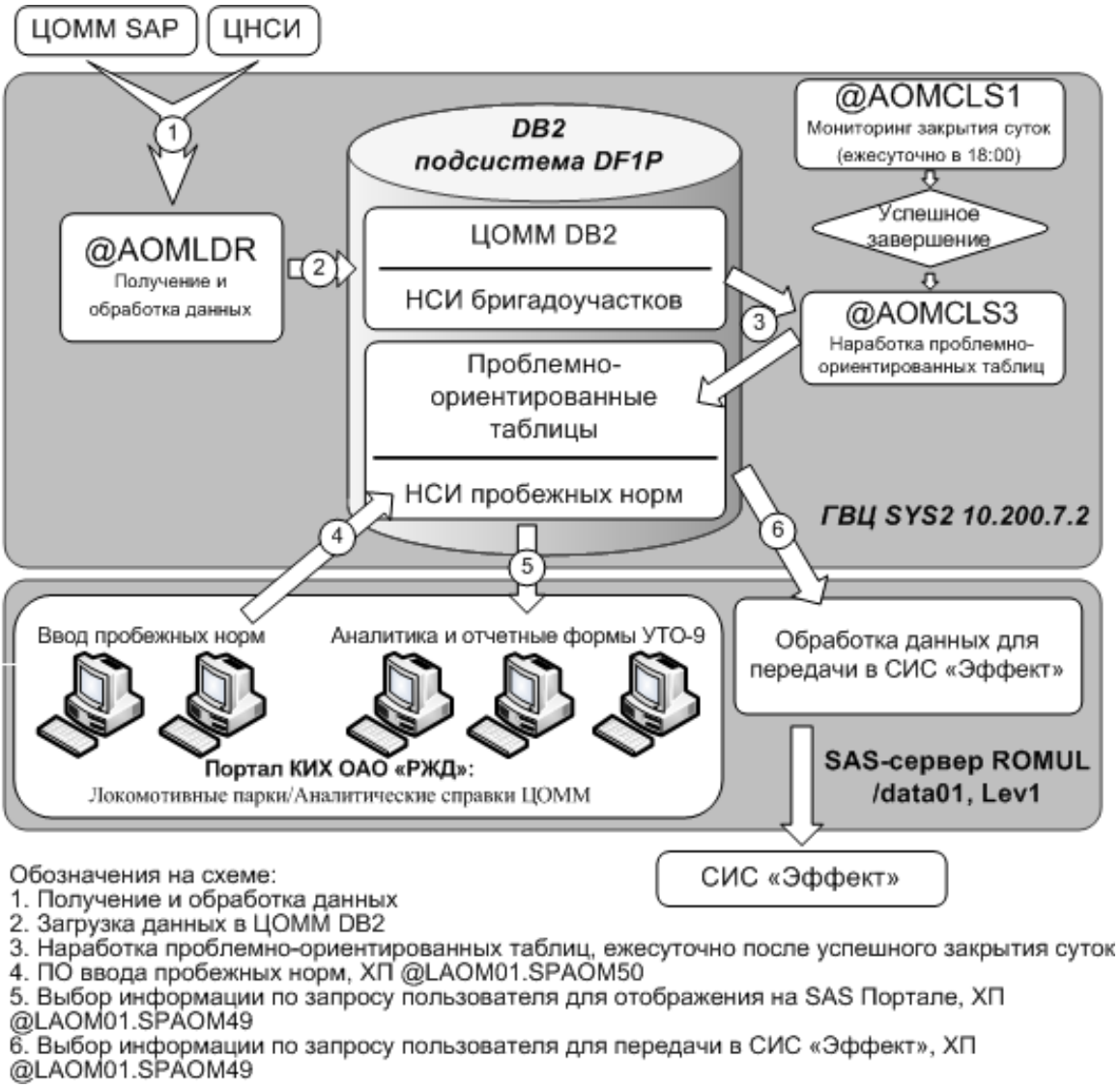
Объектами автоматизации в рамках проекта модификации Системы ЦОММ являются:

- ввод норм по элементам затрат работников локомотивных бригад;

- расчет фактических показателей отчета ф. УТО-9;

- формирование отчета ф. УТО-9;

- выгрузка данных отчета ф. УТО-9 в систему информационного сервиса «Эффект» (СИС «Эффект»).



Обозначения на схеме:
1. Получение и обработка данных
2. Загрузка данных в ЦОММ DB2
3. Наработка проблемно-ориентированных таблиц, ежесуточно после успешного закрытия суток
4. ПО ввода пробежных норм, ХП @LAOM01.SPAOM50
5. Выбор информации по запросу пользователя для отображения на SAS Портале, ХП @LAOM01.SPAOM49
6. Выбор информации по запросу пользователя для передачи в СИС «Эффект», ХП @LAOM01.SPAOM49

Рис. 1. Общая схема функционирования системы

Технологически задача расчета показателей формы УТО-9 тесно связана с системой автоматизированного формирования дорожной и аналитической отчетности ЦОММ (Централизованная обработка маршрутов машиниста) на сетевом уровне: для отчета необходимо использовать информацию базы данных ЦОММ DB2, время расчета коррелирует с фактом закрытия отчетных суток системы. Следовательно, программный комплекс формирования отчетности УТО-9 является функциональной частью системы формирования дорожной и аналитической отчетности ЦОММ на сетевом уровне.

Для формирования отчетности требуется информация:

• по фактической работе бригады — берется из маршрутных листов машиниста, БД ЦОММ DB2;

• по бригадоучасткам — берется из АС ЦНСИ (Автоматизированная система централизованного ведения нормативно-справочной информации);

• по пробежным нормам — из НСИ (Нормативно-справочная информация полученная из ЦНСИ) пробежных норм.

Информация по маршрутным листам машиниста формируется в БД ЦОММ DB2 в автоматическом режиме по оперативным запросам задания @AOMLDR из системы SAP ЦОММ. Подробно передача данных описана в документации к системе. Ежесуточно в 18:00 автоматически стартует задание @AOMCLS1 (мониторинг закрытия отчетных суток по задаче «Отчетность ЦОММ»), которое отслеживает факт корректного закрытия отчетных суток для определения возможности продолжения технологической цепочки формирования отчетности. По завершении задания @AOMCLS1 с кодом 0 автоматически стартует задание @AOMCLS3, выполняет расчеты по предрасчету показателей отчетности, в том числе отчета УТО-9. Входная информация для задания — данные маршрутных листов машиниста (ЦОММ) и НСИ бригадоучастков, результатом работы задания @AOMCLS3 являются данные в проблемно-ориентированных таблицах. Формирование данных для аналитических форм ИХ «Локомотивные парки» и отчетных форм УТО-9 с целью публикации в СИС ЭФФЕКТ происходит посредством вызова хранимой процедуры DB2. Входная информация для хранимой процедуры — данные проблемно-ориентированного сегмента и НСИ пробежных норм. Хранимая процедура вызывается по запросу пользователей CAC-портала и из задания @AOMCLS3.

Информация из АС ЦНСИ в автоматическом режиме подкачивается заданием @AOMLDR в момент запуска и через каждые двенадцать часов непрерывной работы. Данные о пробежных нормах вводятся специалистами дорог посредством формы ввода, расположенной на CAC-портале в разделе «Локомотивные парки».

Результаты

Результатами выполненной работы являются разработанное сетевое web-приложение, предоставляющее пользователю по-

дробный анализ выполнения локомотивными бригадами ОАО «РЖД» норм рабочего времени в грузовом движении, а также учета потерь от их невыполнения.

Данное сетевое приложение (форма) внутри ОАО «РЖД» получило кодовое название УТО-9. Построение отчета ф. УТО-9 может быть выполнено в одном из трех разрезов: по участкам обслуживания, локомотивным эксплуатационным депо и территориальным дирекциям тяги. Для облегчения проведения детального анализа, рабочая смена бригад разбивается на элементы затрат, начиная от явки на работу и заканчивая окончанием работы. На основе полученных данных производится наблюдение и анализ рабочего времени, принятие управленческих решений, премирование или депремирование локомотивных бригад.

Web-приложение включает программные модули, написанные на языках SAS Base, IBM DB2 SQL, HTML, JavaScript, jQuery.

Ссылки

1. Программа инновационного развития ОАО «Российские железные дороги» на период до 2015 года (Белая книга ОАО «РЖД») / сост.: В. И. Якунин, В. Е. Фортов, В. А. Гапанович. 2010. URL: http://www.up-pro.ru/imgs/library/innovations/strategiya-razvitiya-rzhd.pdf

2. Распоряжение ОАО «РЖД» от 30 апреля 2015 г. № 1267р об утверждении внутренней статистической отчетности ОАО «РЖД» о выполнении норм рабочего времени локомотивных бригад грузового движения.

3. Скотт Митчелл. Секреты Web-дизайна. СПб.: НТ Пресс, 2007. 224 с.

4. Кастро Э. HTML и CSS для создания Web-страниц. СПб.: НТ Пресс, 2006. 144 с.

5. Программирование на языке SAS: Основы. SAS Institute, “SAS”, 2008. 260 с.

УДК 004

Среда для построения и экспорта моделей топологий программно-конфигурируемых сетей

Д. А. Рязанцев, Д. Ю. Чалый

Ярославский государственный университет им. П. Г. Демидова
E-mail: ryazansev.dmitry@yandex.ru, chaly@uniyar.ac.ru

Программно-конфигурируемые сети — одно из самых перспективных направлений развития компьютерных сетей. В статье рассматривается среда для построения сетей, особенностью которой является возможность экспорта топологии в виде формальной модели на языке NetKat [2].

Ключевые слова: программно-конфигурируемые сети, SDN, NetKat, топология, Django, Mininet.

Введение

Рост сетевого трафика, связанный с увеличением количества мобильных устройств и формированием крупных высокопроизводительных кластеров для обработки данных привел к изменению требований к сетевым средам. Существующие сетевые архитектуры имеют ряд ограничений, таких как сложность масштабирования, несогласованность политик безопасности, сложность в обслуживании и внедрении новых технологий, зависимость от производителя. Попытки решений этих проблем в рамках существующей архитектуры приводят к появлению все более сложных протоколов передачи данных, что еще сильнее увеличивает нагрузку на сеть и усложняет ее поддержку.

Для решения этих проблем была разработана новая сетевая парадигма, получившая название программно-конфигурируемая сеть (software-definednetworking, SDN) [1]. Ключевым принципом SDN является разделение процессов передачи и управления данными. При таком подходе сетевая инфраструктура используются только для передачи данных, что делает сетевые устройства более

простыми и надежными. А логика управления переносится в так называемые контроллеры — программы, способные отслеживать состояние сети и динамически управлять распределением трафика по узлам. Использование глобального контроллера позволяет быстро добавлять новые узлы сети, переконфигурировать уже существующие и внедрять новые политики безопасности. В таком случае контроллер должен иметь один интерфейс для взаимодействия с сетевыми устройствами, что позволит решить проблему зависимости от производителя оборудования. Таким интерфейсом должен стать протокол OpenFlow [6], который позволяет управлять коммутатором с внешнего устройства.

Формальные методы моделирования управления трафиком в программно-конфигурируемых сетях

Для корректного управления движением сетевого трафика необходим язык, позволяющий формально описывать топологии сетей и политики управления ими. NetKat [2] — это язык для описания процесса передачи сетевого трафика и построения сетевых топологий, основанный на алгебре Клини. Использование строго формализованной модели позволяет проводить формальные доказательства свойств сетей, а также доказывать их эквивалентность. Ключевым элементом является пакет, это объект с полями: адрес отправления (src), адрес получателя (dst), тип протокола (typ), коммутатор (sw) и порт (pt), — которые определяют текущее положения пакета в сети. Политиками называются функции, которые преобразуют одно множество пакетов в другое. Существуют два вида политик: фильтры и модификаторы. Фильтр — это функция, которая возвращает множество пакетов, удовлетворяющих некоторому требованию. Фильтр ($f=n$) для любого пакета pk возвращает множество $\{pk\}$, в случае если поле f пакета pk эквивалентно n . Модификаторы — это функции, которые изменяют свойства пакета на заданное значение. Модификатор $(f \leftarrow n)$ для любого пакета pk устанавливает значение поля f, равное n. Для комбинации применения модификаторов и фильтров вводятся операторы объединения и композиции. Объединение обозначается знаком «+» и является объединением результатов применения политик к входному множеству. Политика $(p+q)$ означает, что к входному множеству пакетов будут применены политики p и q и результатом будет объединение этих множеств.

Композиция пакетов обозначается знаком «*» и является пересечением множеств результатов применения политик. (*p*q*) означает, что к входному множеству пакетов будет применена сначала политика *p*, а затем к полученному множеству — политика *q*. NetKat позволяет моделировать топологию с помощью объединения и композиции фильтров, которые описывают положение пакета в сети, и модификаторов, которые определяют пункт назначения. Простейшая топология имеет вид:

$$t = (\mathsf{sw} = A \cdot \mathsf{pt} = 2 \cdot \mathsf{sw} \leftarrow B \cdot \mathsf{pt} \leftarrow 1)$$

— и означает, что если значения полей пакета *sw=A*, *pt=2*, то этот пакет должен быть отправлен на коммутатор *B*, порт *1*.

Алгоритм построения топологии

Топологию можно представить в виде связного ориентированного графа, где вершинами будут являться коммутаторы (идентифицируемые именем), а ребрами — соединения между ними. Соединение характеризуется четырьмя параметрами: коммутатор отправления, порт отправления, коммутатор получения и порт получения. Порты задаются числовыми значениями начиная с единицы. Пусть дан связный ориентированный граф *D=(V, E)*, где *V* — множество вершин, а *E* — множество ребер. Для построения топологии по такому графу необходимо для каждого ребра из множества *E*, построить следующую строку вида *“(sw=начальная вершина•pt=port_1•sw→конечная вершина•pt→ port_2)”*. Необходимо определить *port_1* и *port_2*. Для этого для каждой вершины графа *v*, создадим счетчик *c(v)*, который будет определять номер последнего свободного порта, установив первоначальное значение равным 1. Будем подразумевать, что для каждого соединения выделятся новый порт. Таким образом, для каждого ребра *v[a,b]* (где *a* — начальная вершина, *b* — конечная вершина) из множества *V* необходимо:

1) для вершин *a* и *b* получить значение счетчиков *port_1=v(a)* и *port_2=v(b)*;

2) построить строку: *s=“(sw=начальная вершина•pt=port_1 •sw→конечная вершина•pt→port_2)”*;

3) увеличить значение счетчиков вершин *v(a)++* и *v(b)++*.

Для всех ребер из множества V можно построить строку s_i, которая определяет это ребро как политику языка NetKat. Иско-

мой топологией будет объединение политик S_i для каждого из ребер графа D.

Визуальная среда для экспорта топологии

Среда представляет собой клиент-серверное приложение, использующее фреймворк Django [3]. Клиент представляет графический интерфейс для взаимодействия с пользователем. Главное окно приложения изображено на рис. 1 и состоит из двух блоков: статического и динамичного.

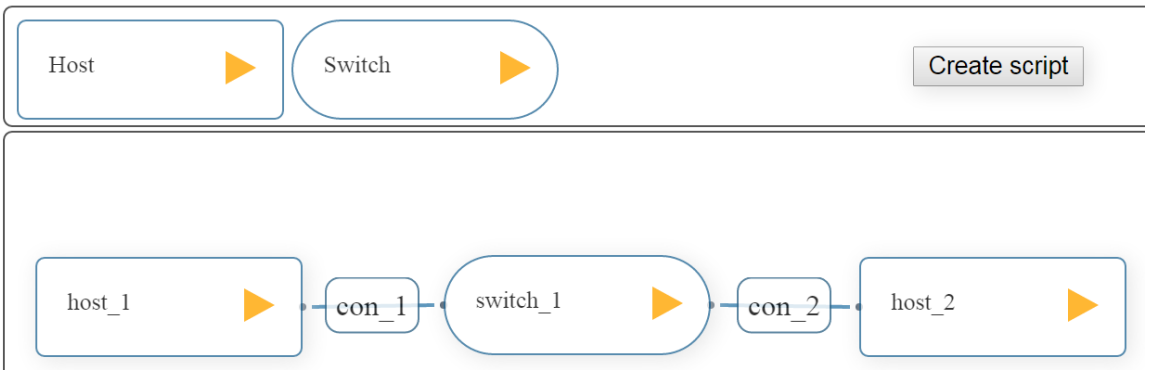


Рис. 1. Главное окно приложения

В статическом блоке находятся элементы, из которых состоит сеть, и кнопка для создания скрипта. Динамический блок представляет собой бесконечное полотно, на котором пользователь может создавать топологию, перетягивая с помощью мышки элементы из статического блока. Серверная часть отвечает за обработку данных и их хранение, а также за генерацию модели топологии. При добавлении, удалении или редактировании элементов клиент отправляет запрос на соответствующий адрес. Сервер принимает запрос, определяет метод для его обработки и вызывает этот метод. Фреймворк Django предоставляет гибкий механизм шаблонов для определения соответствия между адресами и методами. При вызове метод проверяет полученные данные и при необходимости вносит изменения в базу данных. При нажатии на кнопку «Createscript» сервер преобразует данные из хранилища в модель на языке NetKat. Для генерации топологии используется модифицированный алгоритм преобразования, который был описан ранее. Так как среда создает двунаправленную связь между компонентами, то для ее описания необходимо использовать две строки в спецификации NetKat. Другой особенностью является необходимость описания портов коммутаторов, используе-

мых для связи с хостами. Для описания порта используется следующая строка: “(*sw=название коммутатора*pt=номер порта*)”. В модифицированном алгоритме на шаге 2 будет сгенерированы или строки, описывающие соединения между двумя коммутаторами, или строка, описывающая порт для соединения с хостом. Таким образом, итоговая топология будет содержать строки, описывающие соединения типа «коммутатор — коммутатор», а также порты коммутаторов из связи типа «коммутатор — хост». Еще одной возможностью среды является экспорт топологии в эмулятор сетей Mininet [4], что особенно полезно при обучении основам создания сетей.

Заключение

Предложенная среда способна генерировать произвольные топологии, а также экспортировать их в формальную модель сети на языке NetKat и в эмулятор сетей Mininet. Приложение может быть использовано для обучения основным принципам работы и построения программно-конфигурируемых сетей. Исходный код находится в открытом доступе [5].

Ссылки

1. Software-Defined Networking: The New Norm for Networks (PDF). White paper. Open Networking Foundation. 2012. April 13.

2. Netkat: Semantic foundations for networks / C. J. Anderson, N. Foster, A. Guha, J.-B. Jeannin, D. Kozen, C. Schlesinger and D. Walke // In Proc. of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2014), ACM, 2014.

3. The Web framework for perfectionists with deadlines. URL: https://www.djangoproject.com/ (дата обращения: 24.06.2015).

4. An Instant Virtual Network on your Laptop (or other PC). URL: http://mininet.org/ (дата обращения: 24.06.2015).

5. Среда для построения сетевых топологий. URL: https://github.com/onfiro/micronet (дата обращения: 24.06.2015).

6. OpenFlow: Enabling Innovation in Campus Networks / N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner // SIGCOMM Computer Communications Review. Vol. 38. No 2.

УДК 338.33

Особенности функционирования генерирующих компаний оптового рынка электроэнергии

А. В. Уланова, А. Е. Чистяков

Ярославский государственный университет им. П. Г. Демидова
E-mail: ulanova111@ya.ru, alex-ch7@mail.ru

В статье рассматриваются вопросы функционирования компаний ОГК. Выявляются преимущества и недостатки данных структур. Делается сравнительный анализ по ключевым показателям.

Ключевые слова: ОГК, эффективность, рентабельность, волатильность.

В результате проведения мероприятий, связанных с реформированием отрасли электроэнергетики, ее структура стала достаточно сложной. Отрасль состоит из нескольких групп компаний и организаций, каждая из которых выполняет отведенную ей отдельную функцию.

Остановим свое внимание на генерирующих компаниях оптового рынка. Электроэнергия по своим принципам создает закономерность: время производства совпадает со временем потребления. ОГК должна быть готова к выработке, передаче и поставке электроэнергии при возникновении спроса в большом объеме, имея необходимую мощность и запас топлива. ОГК имеет значительный масштаб, и ее мощность находится в пределах от 9 до 18 ГВт. В данной статье под компаниями ОГК понимаются только ОГК-2, ОГК-4 и ОГК-5.

В результате анализа компаний ОГК были выявлены следующие преимущества.

1. Большие размеры электростанций позволяют получать выгоды от эффекта масштаба.

2. Региональная дифференциация позволяет минимизировать риски связанные с конкретным регионом, топливной базой и пр.

3. Низкий КПД компенсируется вводом установок ПГУ с КПД около 60 %.

4. При создании ОГК в каждую компанию было включено по одной крупной прибыльной ГРЭС, которые в основном формировали прибыль всей ОГК.

5. Отсутствие значительных доходов от теплоэнергии позволяет диверсифицировать каждой ГРЭС поставки в различные регионы и не зависеть от районных рынков.

6. Как правило, имеется значительный запас по КИУМ и способность работать в маневровом режиме.

7. Меньше социальных обязательств, чем у ТГК.

8. Отсутствуют неэффективные водогрейные котельные и тепловые сети.

Недостатки:

1. Выработка осуществляется в режиме конденсационных электростанций, т. е. при прочих равных условиях КПД меньше, чем в режиме когенерации.

2. Продажи тепловой энергии более стабильны, но их доля очень незначительная в структуре ОГК.

3. Низкая доля на данном региональном рынке.

4. Загрузка по остаточному принципу.

5. При создании ОГК в каждую компанию наряду с прибыльными ГРЭС были включены устаревшие, затратные ГРЭС.

6. Поставки в различные регионы на дальние расстояния определяются сетевыми ограничениями.

В табл. 1 можно на примере ОГК видеть, что в 2013 г. было падение выработки, как и в компаниях ТГК. Здесь надо отметить и тот факт, что в 2009 г. падение выработки на ТЭС страны было большим, чем в целом по России, т. к. гидро- и атомные электростанции увеличили выработку. В результате изменение выработки ОГК с 2008 г. по 2013 г. включительно составляло 3 %, что является незначительной величиной для генерирующих компаний.

Проведя сравнительный анализ трех предприятий: ОГК-2, ОГК-4,ОГК-5, — следует заметить, что компания ОГК-2 в отличие от ОГК-4 и ОГК-5, имела серьезные изменения в показателях, т. к. в 2011 г. объединилась с ОГК-6. Это очень важное замечание, т. к. изменчивость (волантильность) является важнейшим финансовым показателем и понятием в *управлении финансовыми рисками*.

Таблица 1

Структура производства электроэнергии в России по группам генерирующих компаний

(выр. электроэнергии, млрд кВ*ч)

	2009	2010	2011	2012	2013
ТГК	250	251	253	255	243
ОГК	171	183	186	185	177
Доля ОГК от ТГК (в процентах)	68	73	75	73	73
Доля ОГК от ЕЭС России (в процентах)	17	18	18	18	17
Всего ТГК и ОГК	421	434	439	440	420
Всего в системе ЕЭС России	957	1 004	1 019	1 032	1 023

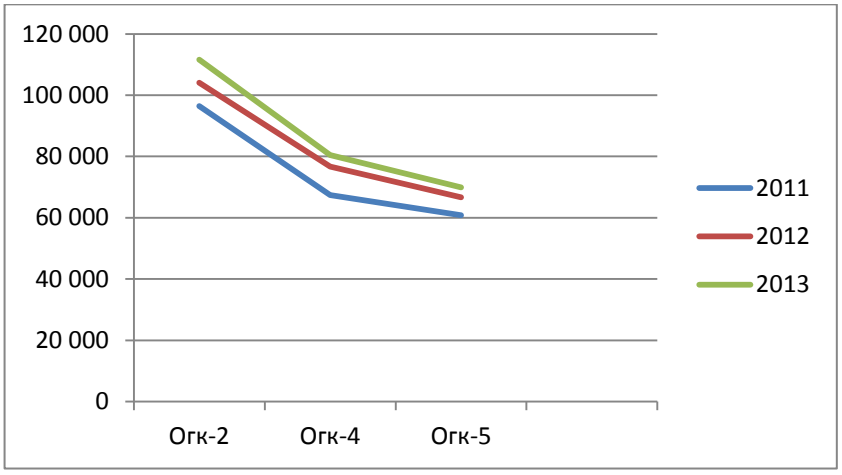


Рис.1. Показатели выручки

Показатели выручки и у компаний в целом синхронны, что обусловлено однородной спецификой их деятельности.

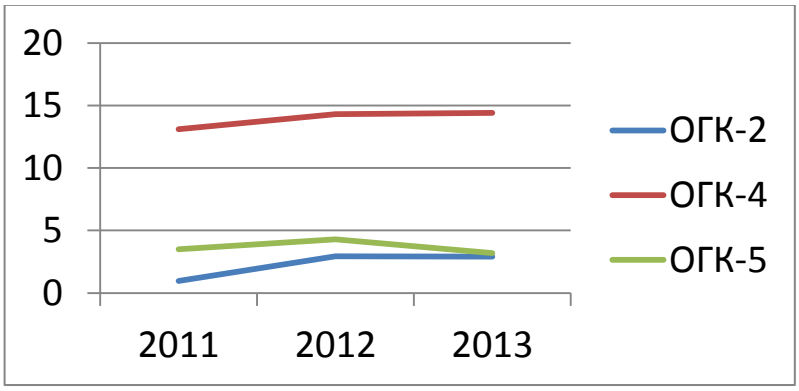


Рис. 2. Показатели рентабельности активов

Если рассмотрим рентабельность активов, то наиболее высокую отдачу демонстрирует ОГК-4. Это не только говорит об эффективности деятельности компании, очищенной от влияния объема заемных средств, но и показывает эффективность использования всего имущества предприятия. Более низкие показатели у ОГК-2 и ОГК-5, что говорит о некоем падающем спросе на продукцию и о перенакоплении активов. У ОГК-2 при мощности, почти в два раза большей, стоимость активов тем не менее находится практически на одном уровне с ОГК-5. Причем у ОГК-4 рентабельность наиболее высокая по меркам всей электроэнергетической отрасли — 15 %.

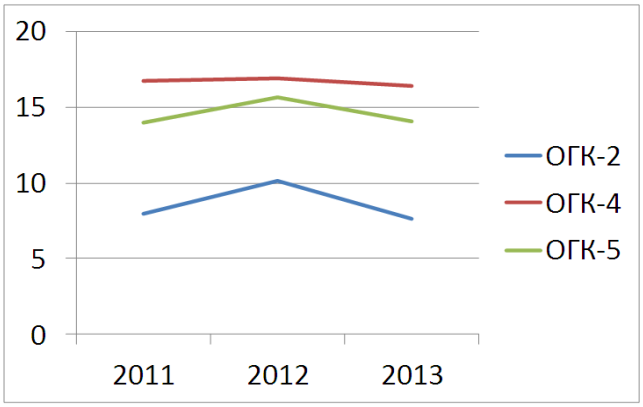


Рис. 3. Показатели рентабельности продаж

Рентабельность продаж у ОГК-4 показывает наиболее прибыльную деятельность предприятия и составляет около 17 %, что также является одним из самых высоких показателей в отрасли.

Дополнением к анализу финансово-хозяйственной деятельности компаний ОГК рассмотрим положение этих компаний на фондовом рынке. Сделать оценку инвестиционной выгодности финансовых вложений в компании ОГК -2, ОГК-3 и ОГК-5 можно через анализ динамики курсов акций.

Будем рассматривать период около 9 лет, его будет достаточно для оценки инвестиционной выгоды финансовых вложений. На основе оценки финансово-хозяйственной деятельности можно сделать вывод, что рассматриваемые электрогенерирующие компании считаются стабильно функционирующими, а производственные и финансовые показатели устойчивы.

В 2008 г. наблюдалось значительное снижение цены акций, обусловленное кризисными явлениями в экономике.

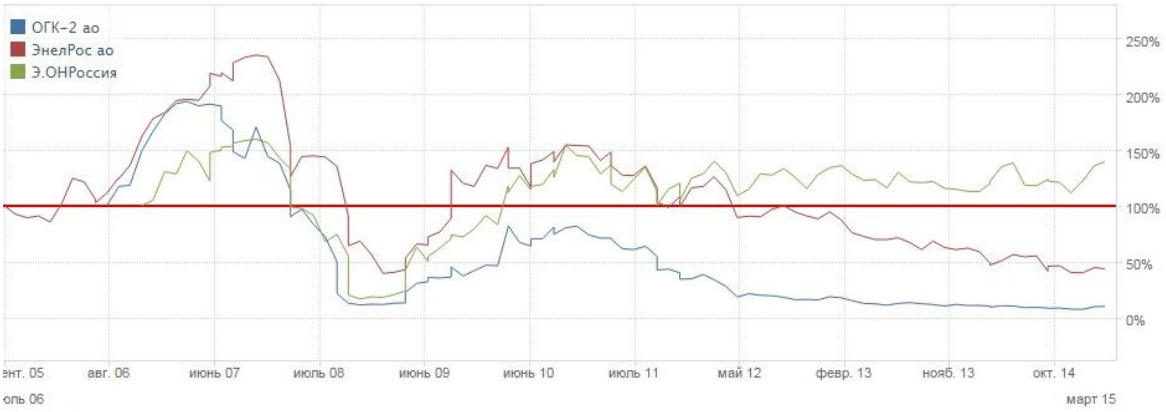


Рис. 4. Динамика изменения цены акций

На графике (рис.4) видно, что компания ОГК-2 имела более стремительное падение в кризис на июль 2008 г. После кризиса восстановление было более медленным. В то же время ОГК-4, восстановившись, продолжает стабильное движение на фоне падения ОГК-2 и ОГК-5.

Ссылки

1. Годовые отчеты ОАО «ОГК-2». URL: www.ogk2.ru
2. Годовые отчеты ОАО «ОГК-4». URL: http://www.eon-russia.ru
3. Годовые отчеты ОАО «ОГК-5». URL: http://www.ogk-5.com/

УДК 519

Разработка стратегий для игры «Погоня за очками»

О. А. Шалаева, А. В. Смирнов

Ярославский государственный университет им. П. Г. Демидова
E-mail: olga.shalaeva.93@mail.ru, alexander_sm@mail.ru

В статье описаны основные подходы к решению антагонистических игр и эффективные стратегии для игры «Погоня за очками». Охарактеризована разработанная программа, реализующая игру и все стратегии. Проводится сравнитель-

ный анализ разработанных стратегий на основе вычислительных экспериментов.

Ключевые слова: антагонистическая игра, игровой процесс, оптимальное заполнение, подходы к решению игр, разработка стратегий, слабая позиция, теория игр.

Введение

Целью работы являлось создание стратегий для игры «Погоня за очками» и их реализация. Были поставлены следующие задачи:

1. Исследование литературы по теории игр, поиск подходов и методов решения антагонистических игр, поиск подходящих приемов для рассматриваемой задачи.
2. Поиск похожих задач и способов их решения.
3. Разработка конкретных стратегий для игры.
4. Программная реализация разработанных стратегий на языке программирования C#.
5. Сравнительный анализ разработанных стратегий.

Постановка задачи

Игровое поле разбито на 64 клетки (по 8 в каждом ряду). У каждого из двух игроков есть кубик, на гранях которого указаны числа от 1 до 6.

Пространственная ориентация кубиков одинакова:

- верхняя грань — 1;
- нижняя грань — 2;
- левая грань — 5;
- правая грань — 6;
- передняя грань — 4;
- задняя грань — 3.

Игроки по очереди совершают ходы. Каждый ход представляет собой перемещение кубика в одну из смежных ячеек, при этом кубик поворачивается таким образом, чтобы грань, бывшая смежной с новой ячейкой, стала нижней (например, первый игрок совершает первый ход вниз; в этом случае грань «4» станет нижней, грань «1» — передней, грань «3» — верхней, грань «2» — задней, а ориентация граней «5» и «6» не изменится).

За каждый ход игроку начисляются очки, их количество совпадает с числом на верхней грани для нового положения кубика.

При этом ни одна клетка игрового поля не может быть пройдена дважды (т. е. игрок не может переместить кубик в клетку, если эта клетка уже посещалась им самим или его соперником).

Для каждого игрока игра заканчивается тогда, когда новый ход невозможен. Если у одного игрока ходы закончились, а у другого — нет, то второй игрок продолжает ходы до тех пор, пока это возможно.

Победителем после подсчета очков становится тот игрок, чья сумма очков больше.

Цель: создать игру «Погоня за очками» и реализовать в ней несколько стратегий поведения компьютера.

Алгоритмы стратегий

Стратегия 1. Антисимметрия

Данная стратегия заключается в следующем.

- Компьютер следующий свой ход совершает антисимметрично относительно хода игрока. Например: если игрок движется вниз, компьютер движется влево; если игрок движется вправо, компьютер движется вверх.

- При этом компьютер должен учитывать возможные ошибки соперника. Когда «Игрок» попадает в слабую позицию, «Компьютер», заметив это, отклоняется от предложенной ему стратегии и совершает ход в ином направлении.

Стратегия 2. Полная симметрия

Данная стратегия заключается в следующем.

- Компьютер следующий свой ход совершает симметрично относительно хода игрока. Например: если игрок движется вниз, компьютер движется вверх; если игрок движется вправо, компьютер движется влево, и наоборот.

Эта стратегия примечательна тем, что при любом развитии игры игровое поле между «Игроком» и «Компьютером» будет разделено поровну.

Стратегия 3. Симметрия

Данная стратегия заключается в следующем.

Компьютер симметрично повторяет действия игрока. Например: если игрок движется вниз, компьютер движется вверх; если игрок движется вправо, компьютер движется влево.

Если в какой-то момент игрок попал в слабую позицию, компьютер пытается «отсечь» игрока, т. е. выполняет свои шаги так, чтобы максимизировать свою территорию.

Сравнительный анализ стратегий

Проанализировав все три алгоритма разработанных стратегий, можно заметить:

1. Игрок, последний ход которого привел к разделению территории и позволил ему занять большую часть игрового поля, выигрывает чаще другого.

2. Большее количество очков, набранных игроком до разделения территории, не приводит к последующему выигрышу.

3. В момент разделения игрового поля стоит заметить, что преимущество по фактору территории получает «Игрок» при безошибочных действиях, но как только «Игрок» допускает ошибки, то часть занятой территории значительно уменьшается. При достаточно большом количестве ошибок преимущество в территории для «Игрока» сводится к нулю.

4. Получение преимущества по фактору очков в момент разделения игрового поля происходит по тому же принципу, как преимущество по фактору территории. «Игрок» набирает больше очков при правильных действиях. А «Компьютер» — наоборот.

Заключение

В рамках работы осуществлено следующее.

1. Проведен аналитический обзор нескольких основных подходов и методов решения задачи, проведен поиск схожих задач и их решения.

2. Написаны алгоритмы трех стратегий поведения компьютера в игре «Погоня за очками» в режиме «игрок — компьютер».

3. Разработанные стратегии реализованы на языке программирования C# в среде разработки Microsoft Visual Studio 2012.

4. Проведен сравнительный анализ разработанных стратегий.

Ссылки

1. Жарков В. А. Компьютерная графика, мультимедиа и игры на Visual C# 2005. М.: Жарков Пресс, 2005. 812 с.
2. Культин Н. Б. Microsoft Visual C# в задачах и примерах. СПб.: БХВ-Петербург, 2007. 241 с.
3. Бишоп Дж., Хорспул Н. C# в кратком изложении. М.: БИНОМ; Лаборатория знаний, 2005. 467 с.
4. Колобашкина Л. В., Алюшин М. В. Основы теории игр. М.: НИЯУ МИФИ, 2010. 164 с.
5. Еремеев А. П. Теоретико-игровые методы принятия решений: учебное пособие. М.: Издательство МЭИ, 2006. 50 с.

УДК 519

Разработка стратегий для игры «Сборщики мусора»

С. А. Шапошникова, А. В. Смирнов

Ярославский государственный университет им. П. Г. Демидова
E-mail: svetashaposhnikova666@gmail.com, alexander_sm@mail.ru

В статье представлены результаты исследования нескольких подходов и методов решения задачи построения оптимального пути для компьютера, показаны схожие задачи и их решения.

Описаны эффективные стратегии для игры «Сборщики мусора», программа, реализующая игру. Проводится сравнительный анализ разработанных стратегий на основе вычислительных экспериментов.

Ключевые слова: разработка стратегий, транспортная задача, оптимальный маршрут, антагонистическая игра, теория игр, подходы к решению игр, сравнение стратегий.

Введение

Целью является создание игры «Сборщики мусора» и реализация в ней нескольких стратегий поведения компьютера.

Задачи работы:

1. Поиск и анализ литературы по теории игр: подходы и методы реализации стратегий, методы их оптимизации, поиск подходящих приемов для рассматриваемой задачи.

2. Разработка конкретных игровых стратегий, в том числе с привлечением подходов, рассмотренных в аналитическом обзоре.

3. Программная реализация разработанных стратегий на одном из языков программирования.

4. Сравнительный анализ разработанных стратегий. Проведение экспериментов и анализ получаемых статистических данных с точки зрения сравнения реализованных стратегий.

Постановка задачи

Требуется разработать стратегии для игры «Сборщики мусора». Игра ведется на поле размера 8 х 8 клеток двумя игроками.

Условия следующие.

• В случайно выбранных клетках игрового поля размещаются шесть контейнеров — три зеленых и три красных.

• Игрокам требуется собрать как можно больше контейнеров. За красный контейнер дается два очка, за зеленый — одно.

• Начальное положение первого игрока (синий мусоровоз) — верхний левый угол игрового поля, второго игрока (желтый мусоровоз) — нижний правый угол игрового поля. Первый ход всегда делает первый игрок.

• Кто наберет больше очков, тот и становится победителем.

Похожие задачи

Транспортная задача (см. [4]) возникает в своем простейшем варианте, когда речь идет о рациональной перевозке некоторого однородного продукта от производителей к потребителю. Поэтому здесь естественно возникает задача о наиболее рациональном прикреплении транспорта, правильном направлении перевозок груза, при котором полностью удовлетворяются потребности при минимальных затратах на транспортировку.

Игра «Лабиринт». Кошка и мышь попадают в глухой лабиринт (см. [5]), показанный на рис. 1. Они могут огибать углы, но не имеют права возвращаться назад. Они передвигаются с одинаковой скоростью, и им дано время, за которое они могут обежать четвертую часть лабиринта. После этого, если мышь

еще уцелела, можно считать, что она спаслась. Решение игры сводится к нахождению выигрышных стратегий игроков.

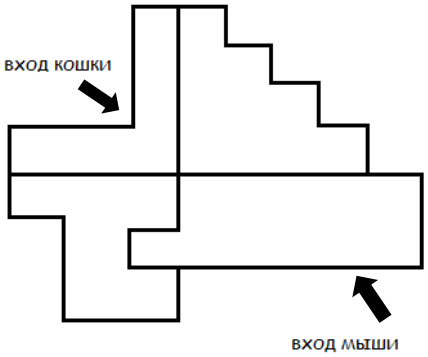


Рис. 1. Лабиринт

Разработка стратегий

Идея первой стратегии:

- считаем расстояния от положения второго игрока до каждого контейнера на игровом поле;
- заносим это в массив как стоимость пути;
- далее отдельно рассматриваем каждый контейнер и, если рядом с ним находятся другие контейнеры, увеличиваем стоимость этого контейнера;
- просматриваем массив и выбираем контейнер с наибольшей стоимостью и начинаем движение к нему (если же первый игрок «съедает» выбранный нами контейнер раньше второго игрока, то заново пересчитываем весь путь);
- после уничтожения контейнера повторяем весь алгоритм заново.

Идея второй стратегии:

- составляем матрицу стоимостей: считаем расстояния от второго игрока до каждого контейнера, затем считаем расстояния для всех пар контейнеров и все это заносим в таблицу;
- далее по таблице строим путь, выбирая контейнер с наименьшей стоимостью;
- движемся к первому в построенном пути контейнеру (если же первый игрок «съедает» выбранный нами контейнер раньше второго игрока, то заново пересчитываем весь путь);
- после уничтожения контейнера повторяем весь алгоритм заново.

Идея третьей стратегии:

- находим все контейнеры, до которых второй игрок доберется быстрее, чем первый игрок;
- считаем все перестановки этих контейнеров;
- для каждой перестановки считаем стоимость пути и выбираем перестановку с наименьшей стоимостью;
- «съедаем» все контейнеры из построенного пути;
- после предыдущего шага повторяем весь алгоритм заново.

Сравнение стратегий

Теперь мы должны сравнить стратегии и выяснить, какая из них является приоритетной для второго игрока, т. е. для компьютера. Для первого игрока мы рассмотрим три возможных модели поведения:

1) высокий уровень — профессиональный игрок — действует всегда оптимальным образом;

2) средний уровень — хорошо играет, но допускает одну — две ошибки за игру;

3) низкий уровень — допускает много ошибок.

Для большей наглядности для каждого уровня будем проводить по 100 экспериментов. В итоге получается 300 экспериментов для каждой стратегии. Количество выигранных игр второго игрока и будет являться вероятностью его победы в процентном соотношении.

Таблица 1

Результаты проведения экспериментов для стратегий

Номер стратегии	1-й уровень		2-й уровень		3-й уровень	
	Победил 1-ый игрок	Победил 2-ой игрок	Победил 1-ый игрок	Победил 2-ой игрок	Победил 1-ый игрок	Победил 2-ой игрок
1	62	38	35	65	20	80
2	66	34	35	65	32	68
3	70	30	43	57	38	62

В ходе проведения экспериментов можно выделить некоторые достоинства и недостатки разработанных стратегий.

• Для первой стратегии несущественным недостатком является то, что в «погоне» за контейнером с наибольшей стоимо-

стью, компьютер может не заметить стоящий поблизости «одинокий» контейнер и пройти мимо. Достоинством же является то, что в большинстве случаев, если на поле стоят рядом несколько контейнеров, то второй игрок успевает съесть их все до того, как это успеет сделать первый игрок.

• Для второй стратегии достоинством является то, что компьютер в первую очередь выбирает контейнер, стоимость которого минимальна, и таким образом при удачном расположении контейнеров может выиграть у первого игрока.

• Для третьей стратегии достоинством является то, что компьютер сначала съедает контейнеры, до которых он доберется раньше первого игрока.

• Также одним из основных факторов, влияющих на исход игры, является расположение контейнеров на игровом поле.

В результате проведения экспериментов получаем следующее:

1. Если компьютер играет с игроком высокого уровня, то наиболее выигрышной стратегией для него является первая. Вероятность победы здесь составляет 38 %, тогда как для второй и третьей стратегий вероятности равны 34 % и 30 % соответственно.

2. Если компьютер играет с игроком среднего уровня, то наиболее выигрышными стратегиями являются первая и вторая. Здесь процент победы равен 65 % в обоих случаях. Для третьей же стратегии это число равно 57 %.

3. Если же компьютер играет с игроком низкого уровня, то наилучшим выбором является опять-таки первая стратегия. Вероятность выигрыша здесь возрастает уже до 80 %. Для второй и третьей стратегии эти значения будут равны соответственно 68 % и 62 %.

Подводя итог, можно утверждать, что наиболее выигрышной стратегией для второго игрока (для компьютера) является первая стратегия. Вторая стратегия тоже является неплохим выбором, вероятность выигрышей в ней достаточно высока.

Заключение

Был проведен аналитический обзор нескольких различных подходов и методов решения задачи построения оптимального пути для компьютера, а также поиск схожих задач и их решения.

Разработаны алгоритмы трех стратегий поведения компьютера в игре «Сборщики мусора» в режиме «человек — компьютер».

Разработанные стратегии реализованы на языке программирования C# в среде разработки Microsoft Visual Studio .NET 2010.

Проведен сравнительный анализ разработанных стратегий.

Ссылки

1. Вентцель Е. С. Исследование операций. М.: Советское радио, 1972. 552 с.

2. Вильямс Дж. Д. Совершенный стратег, или Букварь по теории стратегических игр. М.: Советское радио, 1960. 270 с.

3. Петросян Л. А., Зенкевич Н. А., Семина Е. А. Теория игр: учебное пособие. М.: Высшая школа, 1998. 304 с.

Содержание

==========

Научное издание

Заметки
по информатике
и математике

Сборник научных статей

Выпуск 7

Редактор, корректор М. Э. Левакова
Верстка М. Э. Леваковой

Подписано в печать 08.12.15. Формат 60×84 1/16.
Усл. печ. л. 7,21. Уч.-изд. л. 5,66.
Тираж 27 экз. Заказ

Оригинал-макет подготовлен в редакционно-издательском отделе ЯрГУ

Ярославский государственный университет им. П. Г. Демидова.
150000, Ярославль, ул. Советская, 14.