# ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 004.771

*В.С. Гуров, А.М. Гостин, С.А. Батуркин, Д.В. Суворов*

## МЕЖДУНАРОДНЫЕ СТАНДАРТЫ И СПЕЦИФИКАЦИИ В ДИСТАНЦИОННОМ ОБУЧЕНИИ И ИССЛЕДОВАНИЯХ

*Приводятся основные стандарты и спецификации, применяемые для программного обеспечения и сервисов, используемых для дистанционного обучения и научных исследований. Дано описание наиболее успешных проектов организации удаленного доступа к лабораторному оборудованию как в России, так и за рубежом. Приведена обобщенная структурная схема организации удаленного доступа к лабораторному оборудованию.*

***Ключевые слова**: дистанционное обучение, стандарт, спецификация, удаленный доступ, лаборатория, исследования.*

**Введение.** Развитие международного сотрудничества в области e-learning, масштабные работы по развитию научных сетей, глубокая степень интеграции производственной, научной и учебной областей невозможны без стандартизации обмена данными.

Использование стандартов и спецификаций в процессах дистанционного обучения позволяет организациям планировать политику корпоративного обучения и сотрудничества по обмену опытом. Реализация большинства современных проектов по предоставлению удаленного доступа к лабораторному оборудованию строится как на стандартах e-learning, так и на основе стандартов и спецификаций, присущих конкретной области технического знания.

Разработка типового комплексного решения обмена учебным контентом или удаленного доступа к лабораторному оборудованию позволяет повысить академическую мобильность, эффективность научных исследований путем расширения географии и повышения доступности используемого оборудования.

**Международные стандарты и спецификации.** Среди всех стандартов, относящихся к области электронного обучения (e-learning), в качестве международных можно выделить следующие.

1) IMS – Instructional Management Systems (Системы организации обучения) [1];

2) IEEE – Institute of Electrical and Electronic Engineers (Институт электротехники и электроники) [2];

3) AICC – Airline Industry Computer Based Training Committee (Международный комитет по компьютерному обучению в авиации) [3];

4) SCORM – Sharable Content Object Reference Model (Модель обмена учебными материалами) [4];

5) ARIADNE (Консорциум АРИАДНА) [5];

6) JTC 1/SC 36 - Joint ISO/IEC Technical CommitteeISO (первый совместный комитет ISO – IEC, подкомитет 36)[6], [7];

7) CEN/ISSS – Centre Europeén de Normalisation / the Information Society Standardisation System (Европейский центр нормализации/ Информационное общество систем стандартизации) [8].

Наиболее широко сейчас распространены стандарты AICC, IMS, ISO-IEC, SCORM.

**AICC** - Aviation Industry CBT Committee самый первый стандарт электронного обучения. Его первая версия вышла в 1993 году. Изначально разрабатывался как стандарт СДО транснациональных предприятий авиационной отрасли. Благодаря своему стажу, поддерживается многими системами дистанционного обучения и средствами разработки электронных курсов. Текущее развитие стандарта происходит в области формализации взаимодействия систем управления обучением и систем создания

учебных модулей, что в будущем позволит использовать значительное большее количество учебных курсов. Учебные курсы, совместимые со стандартом, могут использоваться любой совместимой системой дистанционного обучения независимо от того, кем, где и с помощью каких средств были созданы. Подробное описание дано в следующих документах: CMI001 CMI Guidelines for Interoperability (сам стандарт) и последний нормативный документ, регламентирующий системы управления обучением (PENS - Package Exchange Notification Services) - AGR-011 CBT Package Exchange Notification.

**IMS**. Стандарт IMS разработан и поддерживается IMS Global Learning Consortium Работа по его разработке ведется с 1997 года. Он с самого начала создавался для применения в высших учебных заведениях, в отличие от других стандартов (AICC – авиационное транскорпорации, SCORM - военные). Основные направления разработки IMS – метаданные, упаковка содержания, совместимость вопросов и тестов, а также управление содержанием.

Отличительной частью IMS является стандартизация способов описания тестовых заданий и результатов тестирования. Этот раздел в стандарте называется Question and Test Interoperability (QTI) и впервые появился в 2000 году. Стандарт состоит из двух основных частей: Assessment, Section, Item (вопросы и их организация в тесте), QTI Results Reporting (формат записи результатов). Они описывают модели данных и способы описания вопросов и тестов, а также результатов, полученных после того, как тестирвемый ответил на тест или вопрос. Отдельное описание вопросов от средств воспроизведения этих вопросов ( "плееров" тестов) позволило бы осуществлять свободный обмен тестами и их результатами между средствами создания учебного материала, специализированными проигрывателями тестов, системами тестирования и/или системами управления обучением.

Данные о тестах в формате IMS QTI описываются на основе языка XML, корневым элементом которого является элемент questestinterop. Он представляет собой своеобразный контейнер для элементов: assessment (тест), section (тема, секция) и item (тестовое задание). Корневой элемент содержит один или несколько элементов assessment (тест), внутри тестов содержится одна или более тем, и уже темы содержат задания теста.

К сожалению, в данном стандарте не указываются способы обмена этой информацией и как можно, например, организовать механизмы передачи этих даных [9].

Среди стандартов **JTC 1/SC 36** (36 подкомитет занимается вопросами стандартизации информационных технологий в области обучения, образования и тренинга) можно выделить следующие.

1. 2382-36:2008 ISO/IEC Информационные технологии - Словарь - Часть 36: Тренинг, образование и обучение.
2. 12785-1:2009 ISO/IEC Информационные технологии - Тренинг, образование, и обучение - упаковка содержания - Часть 1: Информационная модель.
3. 19778-1:2008 ISO/IEC Информационные технологии - Тренинг, образование и обучение - Совместная технология - Совместное рабочее место - Часть 1: Совместная модель данных рабочего места. Часть 2: Совместная модель данных окружающей среды. Часть 3: Совместная модель данных группы.
4. 19780-1:2008 ISO/IEC Информационные технологии - Тренинг, образование и обучение - Совместная технология - Совместная коммуникация изучения - Часть 1: Основанная на тексте коммуникация.
5. 19796-1:2005 ISO/IEC Информационные технологии - Тренинг, образование и обучение - Качественное управление, гарантия и метрики - Часть 1: Общий подход.
6. 19796-3:2009 ISO/IEC Информационные технологии - Тренинг, образование и обучение - Качественное управление, гарантия и метрики - Часть 3: методы, ссылки и метрики.
7. ISO/IEC 23988:2007 Информационные технологии - свод правил для использования информационной технологии (ИТ) в определении оценок.
8. 24751-1:2008 ISO/IEC Информационные технологии - Индивидуализированная адаптируемость и доступность в e-learning, образовании и обучении - Часть 1: Структура и модель ссылки. Часть 2: "Доступ для всех" личных нужд и предпочтение цифровой поставке. Часть 3: "Доступ для всех" цифрового описания ресурса.

Необходимо отметить, что в разработке стандартов с данным подкомитетом сотрудничают такие организации, как ADL, AICC, AUF, DCMI, IMS Global, Infoterm, LETSI, LTSC. Подобный масштаб совместной работы говорит о желании всех участников современного пространства e-learning выработать унифицированный подход.

В настоящее время подкомитет разрабатывает порядка 26 стандартов и других нормативных документов.

**Стандарт SCORM.** Данный стандарт счи-

тается одной из самых современных версий стандартов обмена учебными материалами. Технологические принципы построения отличаются от описанных выше стандартов.

SCORM - Sharable Content Object Reference Model. Стандарт разрабатывается инициативной группой ADL (Advanced Distributed Learning) с 1997 года при поддержке министерства обороны США. Основан на стандарте XML. В его основе также лежат наработки, предложенные другими разработчиками. В их числе модель данных Computer Managed Instruction (AICC) и стандарт формирования метаданных IEEE P1484 (IMS). Последняя версия стандарта официально выпущена в октябре 2006 года, и в настоящий момент доступно 4-е издание SCORM 2004 [4]. SCORM 2004 содержит следующие разделы.

1. Overview - вводная часть. Общие положения и нормы SCORM.

2. Content Aggregation Model (CAM) - эта часть стандарта описывает структуру учебных блоков и пакетов учебного материала. Пакет может содержать курс, урок, модуль и т. п. В пакет входят xml-файл (Manifest), где описана структура пакета, и файлы, составляющие учебный блок. Manifest включает:

- метаданные (свойства компонентов учебного материала);
- организацию учебного материала (в каком порядке расположены компоненты);
- ресурсы (ссылки на файлы, содержащиеся в пакете);
- sub-Manifest (xml-файл может содержать под-Manifest).

Блоки учебного материала, входящие в пакет, могут быть двух типов: Asset и Sharable Content Object (SCO). Asset — элемент, который не взаимодействует с LMS-сервером (СДО), это может быть html-страница, XML-объект, просто картинка, слайды презентации, звуковой файл или flash-объект и т.п.; SCO — это элемент, который взаимодействует с LMS-сервером: сообщает о ходе и результатах обучения, получает и передает дополнительные данные и т.п. (как минимум SCO сообщает о своем запуске и завершении), кроме того SCO может быть использован для построения совершенно другого обучающего курса.

3. Run-Time Environment (RTE) - эта часть стандарта описывает взаимодействие SCO и системы обучения (Learning Management System, СДО) через программный интерфейс приложения (Application Program Interface, API). Требования SCORM RTE позволяют обеспечить совместимость SCO и LMS, чтобы каждая система дистанционного обучения могла взаимодействовать со SCO таким же образом, как и любая другая, соответствующая стандарту SCORM. LMS должна обеспечивать доставку требуемых ресурсов пользователю, запуск SCO, отслеживание и обработку информации о действиях учащегося.

4. Sequencing and Navigation (SN) - эта часть стандарта описывает, как должны быть организованы навигация и предоставление компонентов учебного материала в зависимости от действий учащегося. Требования SCORM SN позволяют упорядочивать учебный материал в соответствии с индивидуальными особенностями.

5. Conformance Requirements - эта часть содержит полный список требований, проверяемых ADL на соответствие стандарту SCO.

Стандарт SCORM получил широкое распространение и активно используется в следующих системах обучения:

- WebTutor (плаформы Lotus и Microsoft);
- СДО «Прометей»;
- Adobe Connect Pro;
- Oracle Enterprise Learning Management System;
- Competentum;
- IBM Workplace Collaborative Learning;
- SAP Learning Solution;
- Moodle;
- TotalLMS;
- REDCLASS;
- СДО «ДОЦЕНТ»;
- e-Learning Server;

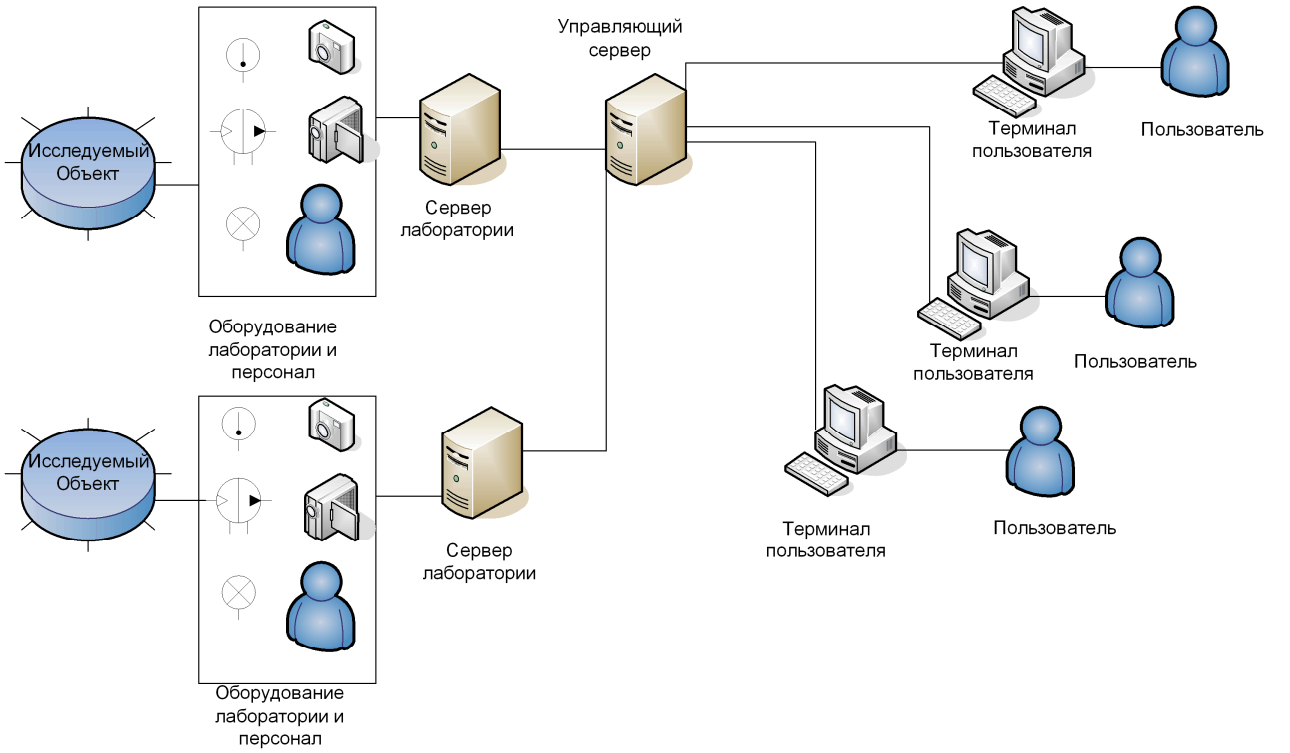
и пр.

Благодаря своей универсальности стандарт применяется не только в процессах электронного обучения, но также и при организации удаленного доступа к оборудованию научных центров. Можно отметить несколько успешных отечественных проектов организации удаленного доступа к лабораторному оборудованию, в основе которых лежал обмен данными на основе SCORM:

1) НОЦ «Нанотехнологические системы» кафедры ИУ4 МГТУ им.Н.Э.Баумана - www.nanotech.iu4.bmstu.ru/onLine.lab - доступ к установкам, реализующим методы сканирующей зондовой микроскопии и молекулярной рамандиагностики;

2) Рязанский государственный радиотехнический университет - http://nanocent.rsreu.ru и http://nanocaml.rsreu.ru – доступ к комплексу зондовых, атомно - силовых и электронных микроскопов для диагностики и комплексных испытаний наночастиц, наноструктурированных и наномодифицированных материалов.

В основе указанных проектов лежит типовая архитектура, представленная на рисунке.




**Архитектура типового решения для организации удаленного доступа к научным лабораториям**

Частичная поддержка стандарта SCORM наблюдается в следующих отечественных проектах:

1) РНЦ «Курчатовский институт» - http://ud.kcsr.kiae.ru – доступ к рабочим станциям синхротронного излучения;

2) НИЯУ «МИФИ» - www.2.mephi.ru/science/iaL/cmm - доступ к сверхвысоковакуумной системе формирования и исследования морфологии, состава и электронной структуры наноматериалов методами сканирующей зондовой микроскопии и рентгеновской фотоэлектронной спектроскопии.

Из зарубежных проектов лабораторий удаленного доступа следует выделить:

1) проект iLab Массачусетсткого технологического института. Первоначально развивался как решение для локальной сети, но довольно быстро приобрел популярность за пределами университета [10];

2. проект удаленной лаборатории государственного университета Среднего Теннеси (Middle Tennessee State University) [11];

3) NIL Remote Lab – комплексное коммерческое решение, включающее в себя целый ряд технологий удаленного доступа, применяемых для различных типов оборудования [12].

Следует отметить тот факт, что архитектура отечественных систем идентична iLabs и различается лишь техническими и программными решениями. В настоящее время, указанные выше отечественные центры включены в сеть ННС, что позволяет широкому кругу пользователей как сети ННС, так и сторонних получать доступ к дорогостоящему оборудованию, что создает неплохой потенциал для коммерциализации проектов.

**Заключение.** В заключение стоит упомянуть ещё раз о широкой географии распространения стандарта SCORM и масштабах его внедрения как в образовательных платформах, так и в проектах удаленного доступа к лабораториям научных центров.

С учетом нынешних тенденций перехода большинства многопользовательских систем к веб-ориентированному администрированию и разработке, прежде всего, на технологической базе тонкого клиента, можно говорить о том, что веб-интерфейсные системы со временем займут значительный сегмент рынка. Стандарт SCORM развивается согласно требованиям времени и на сегодня является единственно верным решением в области стандартизации обмена учебным контентом и данными в научных исследованиях посредством удаленных лабораторий.

### *Библиографический список*

1. www.imsglobal.org/community/index.html – официальный сайт IMS.

2. www.ieee.org - официальный сайт IEEE.

3. www.aicc.org - официальный сайт AICC.

4.http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/2004%204th%20Edition/Documentation.

aspx - официальный сайт ADL инициативы.
5. www.ariadne-eu.org - официальный сайт проекта ARIADNE.
6. www.iso.org – официальный сайт ISO.
7. www.iec.ch - официальный сайт IEC.
8. www.cenorm.be/isss - официальный сайт Европейского центра нормализации.
9. www.scormcourse.ru – русскоязычный сайт посвященный стандарту SCORM.
10. ilab.mit.edu - официальный сайт проекта iLabs Массачуссетского технологического института.
11. Sang Yoo Remote access internetworking laboratory. ISBN:1-58113-798-2 doi > 10.1145/971300.971409, 2004.
12. www.nil.com/faq - официальный сайт NIL Remote Labs.

УДК 681.3.01

*А.И. Баранчиков, Е.А. Асташина*

# МОДИФИЦИРОВАННЫЙ АЛГОРИТМ РЕИНЖИНИРИНГА ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ

*Предлагается алгоритм поиска функциональных зависимостей без применения полного перебора, исключающий избыточность выходных данных и позволяющий получить требуемый результат за приемлемое число шагов. Проведена оценка временной сложности предложенного алгоритма. На примере рассмотрена эффективность поиска. Показана целесообразность исследований в данной области.*

***Ключевые слова**: реинжиниринг, реляционная база данных, отношение, атрибут, функциональная зависимость.*

**Введение.** При проектировании баз данных информационных систем целесообразно использовать информацию о семантике систем, ранее разработанных для той же предметной области, как средство получения дополнительной информации, коррекции ошибок и валидации схем баз данных. При этом могут возникнуть трудности, вызванные недостаточностью документации, недостатками схемы базы данных, неявными структурами, избыточностью, устаревшими конструкциями, кросс-модельным влиянием, некорректным проектированием. В этом случае следует использовать реинжиниринг, который позволит выявить сущности предметной области и отношения между ними и создать формальное описание предметной области на более высоком уровне абстракции [1].

**Целью работы** является разработка эффективного алгоритма поиска функциональных зависимостей, простого для понимания и перевода в программный код, исключающего избыточность выходного множества и выполняемого с приемлемыми временными затратами, позволяющего эффективно использовать компьютерные ресурсы и уменьшить время анализа семантики существующих информационных систем для получения знаний, используемых при проектировании новой системы в той же предметной области.

В работе рассматривается алгоритм поиска функциональных зависимостей, который является одним из первых этапов реинжиниринга информационных систем.

## Теоретические исследования

**Поиск функциональных зависимостей.** Функциональную зависимость (F-зависимость) можно считать расширением понятия ключа отношения. Нестрого говоря, F-зависимость имеет место тогда, когда значения кортежа на одном множестве атрибутов единственным образом определяют эти значения на другом множестве атрибутов [2].

Для поиска функциональной зависимости необходимо применение операций выбора и проекции.

В системах баз данных F-зависимости способствуют обеспечению согласованности и целостности базы данных. Меньшее число F-зависимостей означает меньший объем используемой памяти и меньшее количество проверок. Таким образом, меньшее множество F-зависимостей гарантирует более быстрое исполнение алгоритмов.

Для каждого отношения *r(R)* всегда можно найти все F-зависимости, которым *r* удовлетворяет, перебором всех возможностей с помощью известных алгоритмов [2]. Однако этот подход требует больших временных затрат. Сократить затраты позволит предварительный поиск ключей и атрибутов, на множестве которых

значения кортежа определяются единственным образом значениями на любых других допустимых множествах атрибутов из заданного отношения $r$ , т.е. поиск атрибутов с единственным значением во всех кортежах.

Для поиска атрибутов с единственным значением необходима проверка условия

$$|\pi_x(r)| \le 1, \qquad (1)$$

т.е. проекция $r$ на заданный атрибут $X$ должна содержать не более одного кортежа.

Поскольку указанный атрибут будет функционально определяться любым другим атрибутом из $R$ и их сочетаниями, имеет место F-зависимость вида $R - E \to E$ , где $E$ – атрибут с единственным значением.

При поиске ключей необходима проверка условия

$$|\pi_x(r)| = |r|, \qquad (2)$$

т.е. проекция $r$ на $X$ должна иметь то же число кортежей, что и $r$. В данном случае $X$ - либо атрибут (для простого ключа), либо сочетания атрибутов (для составного ключа).

Определение ключа на языке функциональных зависимостей дает право выделить зависимость вида $K \to R$, где $K$ – ключ отношения.

Выполнение данных условий фиксируется, и указанные атрибуты (сочетания атрибутов) исключаются из дальнейшего поиска F-зависимостей, поскольку, если известны некоторые зависимости из $F$, можно вывести остальные с использованием аксиом вывода (в данном случае аксиомы пополнения) и нет необходимости поиска всех возможных зависимостей с указанными атрибутами.

Для исключения посторонних атрибутов и избыточности выходного множества $F$ применяется проверка рассматриваемых атрибутов на вхождение в ранее найденные зависимости. Если $F$ – неизбыточное множество F-зависимостей, то в нем нет «лишних» зависимостей, удаление которых позволит уменьшить $F$.

Пусть дано отношение $r$ со схемой $R = \{A_1, A_2, ..., A_n\}, a_{ij} = t_j(A_i), j = 1,2,...,m$ . Тогда заданное отношение $r$ будет удовлетворять некоторой функциональной зависимости $A_i \to A_k$ , если $\pi_{A_k}(\sigma_{A_i = a_{ij}}(r))$ содержит не более чем один кортеж для каждого $A_i$-значения $a$, т.е.

$$|\pi_{A_k}(\sigma_{A_i = a_{ij}}(r))| \le 1. \qquad (3)$$

Далее представлен алгоритм поиска F-зависимостей с использованием введенных понятий.

**Экспериментальные исследования**

**Алгоритм FindFunctionalDependence.**

Вход: отношение $r$ со схемой $R = \{A_1, A_2, ..., A_n\}$ .

Выход: $F$ – множество функциональных зависимостей, которым удовлетворяет исходное отношение.

Begin

(1) $F := \varnothing$;

1) Поиск атрибутов с единственным значением

(2) $E := \varnothing$; - вспомогательное множество для найденных атрибутов;

(3) for $A_i \in R, i = 1,2,...,n$ do

(4) if $|\pi_{A_i}(r)| \le 1$ then

(5) $E := E \cup A_i$;

(6) if $E \neq \varnothing$ then

begin

(7) $R := R - E$ ;

(8) $F := F \cup (R \to E)$ ;

end;

2) Поиск ключей

(9) $K := \varnothing$ - множество найденных ключей;

(10) Определить множество всех возможных сочетаний атрибутов $B = \{B_1, B_2, ..., B_p\}$ ;

(11) for $B_i \in B, i = 1,2,..., p$ do

(12) if $|\pi_{B_i}(r)| = |r|$ then

(13) $K := K \cup B_i$;

(14) if $K \neq \varnothing$ then

(15) for $K_j \in K, j = 1,2,...,m$ do

(16) $F := F \cup (K_j \to R - K_j)$ ;

(17) Исключить сочетания, содержащие ключевые элементы, из множества B;

3) Поиск функциональных зависимостей среди оставшихся атрибутов

(18) $F' := \varnothing$ - вспомогательное множество F-зависимостей;

(19) for $B_i \in B, i = 1,2,..., p$ do

begin

(20) Добавить к $F$" все зависимости, содержащие $B_i$ в левой части;

(21) for $A_k \in R, k = 1,2,...,n$ do

(22) for для каждой зависимости вида $Left \to Right$ из $F$" do

(23) if $A_k \notin Right$ then

begin

(24) Ввести признак нахождения зависимости $isFind := true$ ;

(25) if $A_k \notin B_i$ then

begin

(26) $j := 1$ – счетчик кортежей отношения;
(27) do
(28) if $|\pi_{A_k}(\sigma_{B_i=b_{ij}}(r))| \geq 1$ then
(29) $isFind := false$;
(30) $j := j + 1$;
(31) while $(j \leq |r|)$ or $(isFind := true)$;
(32) if $isFind := true$ then
(33) $F := F \cup (B_i \to A_k)$;
end;
end;
(34)$F' := \varnothing$;
end;
end.

В качестве примера рассмотрим следующее отношение:

| $A$ | $B$ | $C$ | $D$ | $E$ |
|---|---|---|---|---|
| $a_1$ | $b_1$ | $c_1$ | $d_1$ | $e_1$ |
| $a_1$ | $b_2$ | $c_2$ | $d_2$ | $e_1$ |
| $a_2$ | $b_1$ | $c_3$ | $d_3$ | $e_1$ |
| $a_2$ | $b_1$ | $c_4$ | $d_3$ | $e_1$ |
| $a_3$ | $b_2$ | $c_5$ | $d_1$ | $e_1$ |

Для каждого атрибута вычисляем мощность проекции данного отношения на каждый из этих атрибутов:

$|\pi_A| = 3$, $|\pi_B| = 2$, $|\pi_C| = 5$, $|\pi_D| = 3$, $|\pi_E| = 1$.

На первом этапе алгоритма выполняется поиск атрибутов с единственным значением. Для этого необходимо найти атрибут, мощность проекции на который меньше или равна 1. В данном случае это атрибут $E$. Ему соответствует F-зависимость $ABCD \to E$. Добавляем ее в выходное множество и исключаем атрибут из дальнейшего поиска.

На следующем этапе ищутся ключи. В данном случае число кортежей отношения $|r| = 5$. Требуется определить атрибут, число кортежей в проекции на который равно 5. Таковым атрибутом является C. Ему будет соответствовать зависимость $C \to ABD$.

Дальнейший поиск будет осуществляться среди следующего множества сочетаний: {$A$, $B$, $D$, $AB$, $AD$, $BD$}. Чтобы установить зависимости между атрибутами и данными сочетаниями, необходимо проверить условие (3). Как только заданное условие не будет выполняться, поиск для заданного атрибута прекращаем.

$|\pi_A(\sigma_{B=b_1}(r))| = 2$,
$|\pi_A(\sigma_{D=d_1}(r))| = 2$,
$|\pi_A(\sigma_{BD=b_1d_1}(r))| = 1$,
$|\pi_A(\sigma_{BD=b_2d_2}(r))| = 1$,
$|\pi_A(\sigma_{BD=b_1d_3}(r))| = 1$,
$|\pi_A(\sigma_{BD=b_2d_1}(r))| = 1$,
$|\pi_B(\sigma_{A=a_1}(r))| = 2$,
$|\pi_B(\sigma_{D=d_1}(r))| = 2$,
$|\pi_B(\sigma_{AD=a_1d_1}(r))| = 1$,
$|\pi_B(\sigma_{AD=a_1d_2}(r))| = 1$,
$|\pi_B(\sigma_{AD=a_2d_3}(r))| = 1$,
$|\pi_B(\sigma_{AD=a_3d_1}(r))| = 1$,
$|\pi_D(\sigma_{A=a_1}(r))| = 2$,
$|\pi_D(\sigma_{B=b_1}(r))| = 2$,
$|\pi_D(\sigma_{AB=a_1b_1}(r))| = 1$,
$|\pi_D(\sigma_{AB=a_1b_2}(r))| = 1$,
$|\pi_D(\sigma_{AB=a_2b_1}(r))| = 1$,
$|\pi_D(\sigma_{AB=a_3b_2}(r))| = 1$.

Таким образом, среди оставшихся атрибутов были найдены следующие зависимости: $BD \to A, AD \to B, AB \to D$.

Результирующее множество $F$ имеет вид:
$F = \{ABCD \to E, C \to ABD, BD \to A, AD \to B, AB \to D\}$.

**Оценка временной сложности алгоритма.** Под временной сложностью алгоритма понимается «время» выполнения алгоритма, измеряемое элементарными операциями, которые необходимо выполнить алгоритму для достижения запланированного результата.

Поскольку время выполнения программы зависит от исходных данных, его можно определить как функцию от исходных данных. Но зачастую время выполнения программы зависит не от исходных данных, а от их «размера»[3].

Естественной мерой объема входной информации для данного алгоритма будет число элементов, подлежащих обработке, т.е. размер входного множества. В этом случае порядок времени выполнения программы $O(n)$ будет определяться как время выполнения в наихудшем случае, т.е. как максимум времени выполнения по всем входным данным размера $n$.

В общем случае время выполнения конечной последовательности программных фрагментов имеет порядок фрагмента с наибольшим временем выполнения $O(max(f(n), g(n)))$.

Чаще всего операторы присваивания имеют некоторое постоянное время выполнения, независящее от размера входных данных, имеющее

порядок $O(1)$, т.е. время, равнозначное некоторой константе.

Для данного алгоритма время выполнения, имеющее порядок, отличный от $O(1)$, имеют операторы (8), (10), (17), а также условные операторы, содержащие вычисление проекции, а именно (4), (12), (28), поскольку вычисление проекции подразумевает проход по всем кортежам дважды для исключения повторяющихся значений и имеет порядок для (4) и (12) $O(|r|^2)$. Оператор (28) содержит проекцию операции выбора, которая также подразумевает проход по всем кортежам один раз. Таким образом, время выполнения оператора (28) составляет $O(|r|^3)$.

Каждый из этих операторов выполняется внутри цикла. Поскольку данные операторы имеют наибольшее время выполнения, время выполнения циклов, их содержащих, будет определяться суммированием полученных значений по числу итераций. При поиске атрибутов с единственным значением оператор (4) выполняется $n$ раз, где $n$ – количество атрибутов в $R$. Следовательно, сложность оператора (3) имеет порядок $O(n \cdot |r|^2)$. При поиске ключей оператор (12) выполняется $p = 2^n - 2$ раз, где $p$ – число возможных сочетаний атрибутов. Тогда время выполнения оператора (11) – $O(|r|^2 \cdot (2^n - 2))$. При дальнейшем поиске зависимостей оператор (28) выполняется в цикле, имеющем 4 уровня вложения: цикл (27) имеет число итераций $|r|$, тело цикла (22) выполняется $l$ раз, где $l$ – количество F-зависимостей, найденных на первых двух этапах алгоритма, цикл (21) подразумевает $n$ итераций, а цикл (19) – $p$ итераций. Таким образом, сложность оператора (19) имеет порядок $O(|r|^4 \cdot (2^n - 2) \cdot n \cdot l)$.

Оператор (17) требует просмотра каждого элемента из множества возможных сочетаний $B$ и имеет сложность $O(2^n - 2)$. Оператор (8), в худшем случае, имеет сложность $O(n - 1)$.

Определение множества возможных сочетаний атрибутов (10) подразумевает перевод чисел от 1 до $2^n - 2$ в двоичную систему и замену единиц на имена атрибутов, имеющих соответствующие порядковые номера в исходном множестве. Данная процедура имеет сложность $O((2^n - 2) \cdot n)$.

Поскольку данные циклы и указанные операторы выполняются последовательно, общая сложность алгоритма имеет наибольшее из полученных значений:

$$O(FindFunctionalDependence) = O(|r|^4 \cdot (2^n - 2) \cdot n \cdot l). \quad (4)$$

**Заключение.** Предложенный алгоритм позволяет обнаружить неявные ограничения и структуры данных, выявить сущности предметной области за приемлемое число шагов и исключить избыточность результатов, что дает возможность использовать его для качественной оценки систем и приобретения знаний для разработки новых систем на основе существующих.

***Библиографический список***

1. Методы и технологии реинжиниринга ИС [электронный ресурс]: труды института системного программирования РАН // Ахтырченко К.В., Сорокваша Т.П. – 2003. – http://www.citforum.ru/SE/project/isr/

2. *Мейер Д.* Теория реляционных баз данных: пер. с англ. – М.: Мир, 1987. – 608 с., ил.

3. *Ахо Альфред В., Хопкрофт Джон, Ульман Джеффри Д.* Структуры данных и алгоритмы: пер. с англ. – М.: Вильямс, 2003. – 384 с., ил.

УДК 510

***Г.С. Орлов***

# ПРОСТРАНСТВЕННО НЕСТАЦИОНАРНЫЕ ПРОСТРАНСТВЕННО – ВРЕМЕННЫЕ ДИНАМИЧЕСКИЕ ВЕРОЯТНОСТНЫЕ НАГРУЖЕННЫЕ ГРАФЫ

*Результаты данной статьи являются дальнейшим обобщением введённых автором в работах [1], [2] понятий динамического вероятностного нагруженного графа и пространственно – временного динамического вероятностного нагруженного графа. Вводится понятие пространственно нестационарного пространственно-временного динамического вероятност-*

*ного нагруженного графа. Анализируются его свойства, даются определения сопутствующих понятий.*

***Ключевые слова:*** *динамический вероятностный нагруженный граф, пространственно – временной динамический вероятностный нагруженный граф, случайный процесс, случайный граф, математическое моделирование.*

**Введение.** Цель работы - дальнейшее обобщение введённых ранее понятий. Изучение и описание свойств новых объектов, анализ их основных характеристик.

В работе [1] были введены пространственно - временные динамические вероятностные нагруженные графы, которые, по сути, являются логическим расширением понятия динамического вероятностного нагруженного графа, также введённого в [1]. В работе [2] на их основе был построен случайный процесс и проанализированы его основные свойства. Напомним, о чём идёт речь.

**Определение_1.** ***Динамическим вероятностным нагруженным графом (ДВНГ) будем называть ориентированный граф*** $G = (X_G, P_G), (|X_G| = n, |P_G| = m)$***, у которого в любой момент времени*** $t \in T$***, для любых вершин*** $x_i, x_j$ ***из*** $X_G$ $l-я$ ***дуга*** $U_{ij}^{(l)} \in P_G$ ***существует с вероятностью*** $p_{ij}^{(l)}(t)$***. Если в данный момент времени*** $t$ ***дуга*** $U_{ij}^{(l)}$ ***существует, то она характеризуется известной априори величиной*** $C_t\left(U_{ij}^{(l)}\right)$ ***- нагрузкой дуги в момент времени*** $t$***. Если*** $t$ ***принимает только дискретные значения, то граф*** $G$ ***будем называть динамическим вероятностным нагруженным графом с дискретным временем (ДВНГсДВ). Если*** $t$ ***меняется непрерывно, то*** $G$ ***- динамический вероятностный нагруженный граф с непрерывным временем (ДВНГсНВ).***

**Определение_2.** ***ДВНГ*** $G = (X_G, P_G), \left(|X_G| = n, |P_G| = m\right)$ ***будем называть пространственно - временным ДВНГ (П-В ДВНГ), если задана некоторая система координат (в соответствующем пространстве*** $\Re^s$***), в которой однозначно определены координаты любой из вершин*** $x_j \in X_G \left(j = \overline{1,n}\right)$ ***этого ДВНГ:*** $x_j = x_j\left(\xi_1^{(j)}, \xi_2^{(j)}, ..., \xi_s^{(j)}\right)$.

Проведём дальнейшее обобщение понятия П-В ДВНГ. Если для любой вершины $x_i \in X_G \left(i = \overline{1,n}\right)$ её координаты не зависят от времени $t \in T$, то такой П-В ДВНГ будем называть пространственно стационарным. Если же хотя бы для какой-то одной вершины $x_j$ из $X_G$ её координаты зависят от времени, то такой П-В ДВНГ будем называть пространственно нестационарным П-В ДВНГ. Очевидно, что в случае пространственно нестационарного П-В ДВНГ координаты его вершин могут быть заданы посредством некоторых детерминированных или даже стохастических (*случайных)* функций, зависящих от времени $x_j = x_j\left(\xi_1^{(j)}(t), \xi_2^{(j)}(t), ..., \xi_s^{(j)}(t)\right)$, которые могут быть как дискретными (*принимать конечное или счётное множество значений*), так и непрерывными (*мощность множества значений которых – континуум*).

**Определение_3.** ***Пространственно нестационарный П-В ДВНГ будем называть графом с дискретным числом состояний, если все функции координат всех его вершин являются дискретными. Если существует хотя бы одна координатная функция*** $\xi_i^{(j)}(t)$***, мощность множества значений которой – континуум, то граф будем называть П-В ДВНГ с континуальным числом состояний.***

Рассмотрим дискретный случай. Пусть $x_j = x_j\left(\xi_1^{(j)}(t), \xi_2^{(j)}(t), ..., \xi_s^{(j)}(t)\right)$ - любая из вершин графа $G$. Предположим, что в некоторой пространственно – временной системе координат $O_{\zeta_1, \zeta_2, ..., \zeta_s, t} \overset{def}{=} O_{\bar{\zeta}, t}$ определена конечная совокупность точек $M_k\left(z_1^{(k)}, z_2^{(k)}, ..., z_s^{(k)}, t\right), \left(k = \overline{1,m}\right)$, жёстко закреплённых в пространстве (*то есть не меняющих своего положения относительно системы координат* $O_{\bar{\zeta}, t}$ *в зависимости от временного сечения*). Будем считать, что в любой момент времени $t \in T$ рассматриваемая вершина $x_j$ может совпадать с одной из этих точек (*и только с ними*). Обозначим через $p_{\hat{t}}^{(j=k)}$ вероятность события: «в момент времени $\hat{t} \in T$

вершина $x_j$ П-В ДВНГ $G$ находится в точке $M_k\left(z_1^{(k)}, z_2^{(k)}, ..., z_s^{(k)}, \hat{t}\right)$», то есть $p_t^{(j=k)} \overset{def}{=} P\left\{\bigcap_{i=1}^{s}\left(\xi_i^{(j)}(t) = z_i^{(k)}\right)\right\} = P\left\{x_j \equiv M_k \,\middle|\, t = \hat{t}\right\}$.

Ясно, что в том случае, когда все координатные функции $\xi_i^{(j)}(t)$ являются детерминированными, $p_t^{(j=r)} \in \{0,1\}$ $\left(r = \overline{1,m}\right)$.

Отметим, что в общем случае допустима ситуация, когда несколько вершин пространственно нестационарного П-В ДВНГ могут в момент $t \in T$ оказаться в одной и той же точке пространства (*то есть совпасть*). Пусть $\left(x_1, ..., x_j, ..., x_n\right)$ - кортеж вершин $G = (X_G, P_G)$, а $\left(M_1, ..., M_k, ..., M_m\right)$ - кортеж фиксированных точек пространства $\Re^s$. Обозначим через $\left(\overline{P_1}(t), ..., \overline{P_j}(t), ..., \overline{P_n}(t)\right)$ - кортеж векторов вероятностей отдельных вершин рассматриваемого графа [здесь $\overline{P_j}(t) \overset{def}{=} \left(p_t^{(j=1)}, ..., p_t^{(j=k)}, ..., p_t^{(j=m)}\right)$]. По условию для конкретной (одной из $X_G$) вершины графа $x_j$ в любом временном сечении $t \in T$ совокупность событий $\left\{x_j \equiv M_k\right\}$ $\left(k = \overline{1,m}\right)$ образует полную группу событий, поэтому $\left(\forall\ j = \overline{1,n}\right)$: $\sum_{i=1}^{m} p_t^{(j=i)} = 1$. Таким образом, можно считать, что в любой момент времени $t \in T$ определена $n$ - мерная векторная случайная величина $\overline{\Psi}(t) \overset{def}{=} \left(\psi_1(t), ..., \psi_j(t), ..., \psi_n(t)\right)$, координаты $\psi_j(t)$ которой равны номеру точки пространства [одной из последовательности точек $\left(M_1, ..., M_k, ..., M_m\right)$], с которой в данном временном сечении совпадает вершина $x_j$.

Если для любого момента времени $t \in T$ и каждой вершины графа $x_j$ определены вероятности $p_t^{(j=r)}$, то функция (аргумента $t$) математического ожидания для $\psi_j(t)$ определяется как $m_t\left[\psi_j(t)\right] = \sum_{k=1}^{m}\left\{k \cdot p_t^{(j=k)}\right\}$. Таким образом, логично считать, что в момент времени $t \in T$ вершина $x_j$ графа, *скорее всего,* будет находиться в точке пространства, номер которой *ближе всего* к значению $m_t[\psi_j(t)]$. Ясно, что для $\overline{\psi}(t)$ математическое ожидание $m_t\left\{\overline{\psi}(t)\right\} = \left(m_t[\psi_1(t)], ..., m_t[\psi_j(t)], ..., m_t[\psi_n(t)]\right)$.

Из вышеизложенного следует, что на множестве вершин $X_G$ определён <u>случайный процесс</u> $\Im(\omega, t)$ $\left(\omega \in \Omega, t \in T\right)$, позволяющий в каждое временное сечение получить распределение вершин пространственно нестационарного графа с дискретным числом состояний по точкам $\left(M_1, ..., M_k, ..., M_m\right)$ (*здесь, как общепринято, буквой* $\Omega$ *обозначено пространство элементарных событий, соответствующее данному случайному процессу*). Если множество $T$ содержит конечное или счётное число элементов, то будем говорить, что имеется случайный процесс $\Im(\omega, t)$ с дискретным носителем. Если $T$ имеет мощность континуум, то будем говорить, что задан случайный процесс $\Im(\omega, t)$ с непрерывным носителем.

Введём **<u>метрические характеристики для пространственно нестационарных П-В ДВНГ</u>**. Рассмотрим произвольный момент времени $t \in T$. Пусть для каждой вершины $x_j$ графа $G = (X_G, P_G)$ известны её координаты в этом временном сечении, то есть $\left(\forall\ j = \overline{1,n}\right)$: $x_j(t) = x_j\left(\xi_1^{(j)}(t), \xi_2^{(j)}(t), ..., \xi_s^{(j)}(t)\right)$.

**<u>Определение 4.</u>** ***Расстоянием между вершинами*** $x_j\left(\xi_1^{(j)}(t), \xi_2^{(j)}(t), ..., \xi_s^{(j)}(t)\right)$ ***и*** $x_l\left(\xi_1^{(l)}(t), \xi_2^{(l)}(t), ..., \xi_s^{(l)}(t)\right)$ ***графа*** $G = (X_G, P_G)$ ***в момент времени*** $t \in T$ ***будем называть величину***

$$r_t\left(x_j, x_l\right) \overset{def}{=} \sqrt{\sum_{i=1}^{s}\left(\xi_i^{(j)}(t) - \xi_i^{(l)}(t)\right)^2}.$$

Ясно, что если в сечении $\hat{t} \in T$ имеет место: $\left(x_j \equiv M_k \,\middle|\, t = \hat{t}\right)$ и $\left(x_l \equiv M_q \,\middle|\, t = \hat{t}\right)$, то $r_t\left(x_j, x_l\right) = r_t\left(M_k, M_q\right) = \sqrt{\sum_{i=1}^{s}\left(z_i^{(k)} - z_i^{(q)}\right)^2}$.

**<u>Определение 5.</u>** ***Диаметром графа*** $G = (X_G, P_G), \left(|X_G| = n, |P_G| = m\right)$ ***в момент времени*** $t \in T$ ***будем называть величину*** $D_G(t) = \max_{\{x_j, x_l\}} r_t\left(x_j, x_l\right)$.

**<u>Определение 6.</u>** ***Максимальным удалением от вершины*** $x_j$ ***графа*** $G = (X_G, P_G)$ ***в момент времени*** $t \in T$ ***будем называть величину*** $\chi_t\left(x_j\right) = \max_{x_l}\left\{r_t\left(x_j, x_l\right)\right\}$.

**Определение 7.** ***Радиусом графа*** $G = (X_G, P_G), (|X_G| = n, |P_G| = m)$ ***в момент времени*** $t \in T$ ***будем называть величину*** $R_G(t) = \min\limits_{x_j} \{\chi_t(x_j)\}$.

Отметим, что в случае стохастических координатных функций все введённые метрические характеристики будут случайными функциями.

Рассмотрим два произвольных временных сечения $t_1$, $t_2$ $(t_1 < t_2)$ из множества $T$, на котором анализируется данный процесс. Для каждой вершины $x_j$ $\left(\forall\, j = \overline{1, n}\right)$ графа найдём её координаты в моменты времени $t_1, t_2$: $x_j(t_1) = x_j\left(\xi_1^{(j)}(t_1), \xi_2^{(j)}(t_1), ..., \xi_s^{(j)}(t_1)\right)$ и $x_j(t_2) = x_j\left(\xi_1^{(j)}(t_2), \xi_2^{(j)}(t_2), ..., \xi_s^{(j)}(t_2)\right)$. Так как в данном случае анализируется граф с дискретным числом состояний, то, зная координаты всех вершин (*значения всех координатных функций*), мы знаем с какими конкретно точками пространства эти вершины совпадают в сечениях $t_1, t_2$. Пусть в момент $t_1$ вершина $x_j$ совпадает с точкой $M_{j_1} \in \Re^s$, а в момент $t_2$ - с точкой $M_{j_2} \in \Re^s$ $(t_1 < t_2)$.

**Определение 8.** ***Величиной смещения вершины*** $x_j$ ***за период*** $\Delta t = t_2 - t_1$ ***будем называть величину***

$$\rho_{x_j}(t_1, t_2) \overset{def}{=} \sqrt{\sum_{i=1}^{s}\left(z_i^{(j_2)} - z_i^{(j_1)}\right)^2}\,.$$

**Определение 9.** ***Величиной смещения графа*** $G = (X_G, P_G)$ ***за период*** $\Delta t = t_2 - t_1$ ***будем называть величину***

$$\rho_G(t_1, t_2) \overset{def}{=} \sum_{j=1}^{n} \rho_{x_j}(t_1, t_2).$$

Рассмотрим случайный процесс $\Im(\omega, t)$ с непрерывным носителем $T = [t_н, t_к]$. Проведём дискретизацию носителя $T$. Пусть $t_0 = t_н$, $t_1 = t_0 + \Delta t$, ..., $t_w = t_к = t_{w-1} + \Delta t$, где $\Delta t$ - выбранный шаг дискретизации.

**Определение 10.** ***Если существует конечный предел***

$$\lim_{\substack{\Delta t \to 0 \\ (w \to \infty)}} \left( \frac{\max\limits_{j = \overline{0, (w-1)}} \{D_G(t_j)\}}{\min\limits_{j = \overline{0, (w-1)}} \{D_G(t_j)\}} \right) \overset{def}{=} K_{раст}(G),$$

***то будем называть его коэффициентом растяжения графа*** $G = (X_G, P_G)$.

Отметим, для случайного процесса с дискретным носителем ($T = \{t_0, t_1, ..., t_w\}$)

$$K_{раст}(G) = \left( \frac{\max\limits_{j = \overline{0, w}} \{D_G(t_j)\}}{\min\limits_{j = \overline{0, w}} \{D_G(t_j)\}} \right).$$

Подчеркнём, что в случае стохастических координатных функций все введённые ранее метрические характеристики пространственно нестационарного П-В ДВНГ будут случайными функциями.

**Определение 11.** ***Если существует такая точка*** $M_k \in \Re^s$, ***что событие*** $\left\{ \bigcap\limits_{j=1}^{n} \left(x_j \equiv M_k\right) \middle| t \in T \right\}$ ***не является невозможным событием, то будем называть её точкой сбора вершин графа*** $G = (X_G, P_G)$.

Отметим, что в том случае, когда все координатные функции $\xi_i^{(j)}(t)$ являются детерминированными, поиск ответа на вопрос «имеет ли данный граф точку сбора вершин?» не вызывает принципиальных затруднений.

**Определение 12.** ***Если для какой-либо реализации случайного процесса*** $\Im(\omega, t)$ ***было зафиксировано такое временное сечение*** $\hat{t} \in T$, ***что событие*** $\left\{ \bigcap\limits_{j=1}^{n} \left(x_j \equiv M_k\right) \middle| \hat{t} \in T \right\}$ ***произошло, то момент времени*** $\hat{t} \in T$ ***будем называть моментом (временем) сбора вершин графа*** $G = (X_G, P_G)$.

В том случае, когда все координатные функции $\xi_i^{(j)}(t)$ являются детерминированными мы, очевидно, всегда можем определить момент сбора вершин графа, если он существует. Если анализируется стохастический процесс $\Im(\omega, t)$, то вопрос существования (*а уж тем более нахождения*) времени сбора вершин графа требует дополнительных исследований.

Рассмотрим некоторое временное сечение $\hat{t} \in T$. Предположим, что $\left(x_j \equiv M_k \middle| t = \hat{t}\right)$ (*в момент времени* $\hat{t}$ *вершина* $x_j$ *графа* $G$ *находится в точке* $M_k$). Пусть $M_l$ - какая-то другая точка из рассматриваемой совокупности $(M_1, ..., M_k, ..., M_m)$.

**Определение 13.** ***Будем говорить, что в***

***момент*** $\hat{t}$ ***точка*** $M_l$ ***достижима вершиной*** $x_j$ ***графа*** $G$, ***если для некоторой величины*** $\Delta t = const > 0$ ***вероятность события*** $\left(x_j \equiv M_l \,\middle|\, t = \hat{t} + \Delta t\right)$ ***отлична от нуля [при условии, что*** $\left(\hat{t} + \Delta t \in T\right)$***].***

**Определение 14.** ***Пусть зафиксированы некоторая вершина*** $x_j$ ***графа*** $G$ ***и момент времени*** $\hat{t} \in T$. ***Множество точек*** $\left\{ M_{j_1}, ..., M_{j_i}, ..., M_{j_r} \right\}$, ***достижимых вершиной*** $x_j$ ***в момент времени*** $\hat{t}$, ***будем называть множеством достижимости для вершины*** $x_j$ ***в момент*** $\hat{t}$ ***и обозначать*** $V_{x_j}\left(\hat{t}\right)$.

Заметим, что $V_{x_j}\left(\hat{t}\right)$ не может быть пустым множеством, так как оно как минимум содержит одну вершину.

Докажем почти очевидный, но важный с точки зрения приложений факт.

**Теорема 1. Если в момент времени** $\hat{t} \in T$ **для вершины** $x_j \in X_G$ $(x_j \equiv M_k \,|\, t = \hat{t})$ **имеет место** $V_{x_j}\left(\hat{t}\right) = \{M_k\}$, **то ни при каком** $t \in T\left(t > \hat{t}\right)$ **вершина** $x_j$ **не может попасть ни в одну из точек множества** $\left\{ M_1, ..., M_k, ..., M_m \right\}$ **за исключением точки** $M_k$.

Доказательство. Предположим противное. Пусть в условиях теоремы существует такой момент времени $\tilde{t} \in T\left(\tilde{t} > \hat{t}\right)$, что $P\left(x_j \equiv M_l \,\middle|\, t = \tilde{t}\right) \neq 0$, где $l \neq k$. То есть точка $M_l \in \left\{ M_1, ..., M_k, ..., M_m \right\}$, $M_l \neq M_k$. Но тогда для приращения $\Delta t = \tilde{t} - \hat{t} > 0 : P\left(x_j \equiv M_l \,\middle|\, t = \hat{t} + \Delta t\right) \neq 0$ и, следовательно, $M_l \in V_{x_j}\left(\hat{t}\right)$. Получили противоречие.

**Следствие. Пусть** $x_j \in X_G, T = [t_н, t_к]$ **и** $V_{x_j}\left(t_н\right) = \{M_k\}$. **Тогда для любого** $\hat{t} \in T$ **выполняется** $V_{x_j}\left(\hat{t}\right) = \{M_k\}$.

**Определение 15.** ***Пусть*** $M_k \in V_{x_j}\left(\hat{t}\right)$. ***Наименьшее значение величины*** $\Delta t$, ***при котором*** $P\left\{ x_j \equiv M_k \,\middle|\, t = \hat{t} + \Delta t \right\} \neq 0$ ***[при условии, что*** $\left(\hat{t} + \Delta t \in T\right)$***], будем называть временем доступа вершины*** $x_j$ ***графа*** $G$ ***в точку*** $M_k$ ***в момент*** $\hat{t}$ ***и обозначать следующим образом*** $t_{дост}^{x_j}\left(M_k; \hat{t}\right)$.

Ясно, что для любого временного сечения $\hat{t} \in T$ и каждой вершины $x_j$ графа $G$ определено $V_{x_j}\left(\hat{t}\right)$.

**Определение 16.** ***Индексом доступности -*** $I_{дост}\left(M_k; \hat{t}\right)$ ***точки*** $M_k$ ***в момент времени*** $\hat{t} \in T$ ***будем называть количество множеств*** $V_{x_j}\left(\hat{t}\right)$, ***каждое из которых содержит*** $M_k$.

Отметим очевидный факт. Если для какого-либо временного сечения $\hat{t} \in T$ точка $M_k$ является точкой сбора вершин графа $G$, то $I_{дост}\left(M_k; \hat{t}\right) = \left|X_G\right| = n$.

**Определение 17.** ***Модой совокупности точек*** $\left\{ M_1, ..., M_k, ..., M_m \right\}$ ***в момент времени*** $\hat{t} \in T$ ***будем называть множество*** $Mod_{\hat{t}}\left\{M_1, ..., M_m\right\} \overset{def}{=} \left\{ M_{i_1}, M_{i_2}, ..., M_{i_l} \right\}$ ***точек данной совокупности, у которых в данном временном сечении*** $\hat{t} \in T$ ***индекс доступности имеет наибольшее значение.***

Из определения моды совокупности точек непосредственно следует, что она является элементом булеана $B\left\{ M_1, ..., M_k, ..., M_m \right\}$.

Заметим, что если для какого-либо временного сечения $\hat{t} \in T$ точка $M_k$ является точкой сбора вершин графа $G$, то $M_k \in Mod_{\hat{t}}\left\{M_1, ..., M_m\right\}$.

Далее рассмотрим пространственно нестационарные П-В ДВНГ с континуальным числом состояний. Пусть вершина $x_j = x_j\left(\xi_1^{(j)}(t), \xi_2^{(j)}(t), ..., \xi_s^{(j)}(t)\right)$ - любая из вершин графа $G$. Будем предполагать, что всевозможные точки пространства $\Re^s$, в которые *априори* может попасть вершина $x_j$ в

некоторый момент времени $t \in T$, образуют ограниченное множество $\Lambda(x_j) \subset \Re^s$. Если точка $M_k^{(j)} \in \Lambda(x_j)$, то, очевидно, что вероятность события «в момент $\hat{t} \in T$ вершина $x_j$ будет находиться в данной конкретной точке $M_k^{(j)}$ области $\Lambda(x_j) \subset \Re^s$» равна нулю (то есть $p_t^{(j=k)} \overset{def}{=} P\left\{x_j \equiv M_k^{(j)} \,\middle|\, t = \hat{t}\right\} = 0$). Поэтому введём для каждого временного сечения $\hat{t} \in T$ вероятность попадания вершины $x_j$ в некоторую малую $\varepsilon$- окрестность $O_\varepsilon(M_k^{(j)})$ ($s$ - мерный гиперкуб, с ребром $2 \cdot \varepsilon$) точки $M_k^{(j)}$ из $\Lambda(x_j) \subset \Re^s$:

$$\widetilde{p}_{\hat{t}}^{(j=k)} \overset{def}{=} P\left\{x_j \in O_\varepsilon(M_k^{(j)}) \,\middle|\, t = \hat{t}\right\}.$$

Представим область $\Lambda(x_j) \subset \Re^s$ в виде объединения непересекающихся подобластей: $\Lambda(x_j) = \bigcup_{i=1}^{w} O_\varepsilon(M_i^{(j)})$ (*заметим, что эта операция осуществима в принципе, так как по условию* $\Lambda(x_j) \subset \Re^s$ *- ограниченное множество*). Теперь условие нормировки будет иметь вид $\sum_{i=1}^{w} \widetilde{p}_t^{(j=i)} = 1$. Если $X_G = \{x_1, ..., x_j, ..., x_n\}$, то область $\bigcup_{j=1}^{n} \Lambda(x_j) \subset \Re^s$ будем называть **областью функционирования графа** $G$ и обозначать $D(X_G)$ (*область определения для множества вершин графа*).

Отметим, что область функционирования конкретного графа $D(X_G)$ можно рассматривать как некоторое ***топологическое пространство***, в котором множества $\Lambda(x_j)$ соответствуют ***открытым множествам*** (*задают топологию пространства*). Ясно (см. например [3]), что совокупность этих областей образует и ***базу*** данного топологического пространства. По построению множество точек $\{M_1^{(1)}, ..., M_k^{(j)}, ..., M_l^{(n)}\}$ является ***всюду плотным*** в $D(X_G)$. Откуда сразу же следует, что $D(X_G)$ - ***сепарабельное*** пространство.

**Определение 18.** ***Пусть*** $D(X_G) = \bigcup_{j=1}^{n} \Lambda(x_j)$ ***и*** $\forall\ i \neq j : \Lambda(x_i) \cap \Lambda(x_j) = \varnothing$. ***Тогда области*** $\Lambda(x_j)\left(j = \overline{1, n}\right)$ ***будем называть зонами ответственности (вершины*** $x_j$***), а пространство*** $D(X_G)$ ***- пространством, допускающим разбиение на зоны ответственности вершин.***

Как и раньше считаем, что на $D(X_G)$ определена метрика, позволяющая для любого $t \in T$ находить расстояние $r_t(M_i^{(j)}, M_k^{(l)})$ между любыми двумя фиксированными точками $M_i^{(j)}$ и $M_k^{(i)}$ из $D(X_G)$. Поэтому, если при $\hat{t} \in T$ имели место события $\left\{x_j \in O_\varepsilon(M_k^{(j)}) \,\middle|\, t = \hat{t}\right\}$ и $\left\{x_l \in O_\varepsilon(M_q^{(l)}) \,\middle|\, t = \hat{t}\right\}$, то по аналогии с рассмотренным выше дискретным случаем $r_{\hat{t}}(x_j, x_l) = r_{\hat{t}}(M_k, M_q) = \sqrt{\sum_{i=1}^{s} \left(z_i^{(k)} - z_i^{(q)}\right)^2}$.

Подчеркнём, что в общем случае $r_t(x_j, x_l)$ - случайная функция.

**Определение 19.** ***Пусть*** $D(X_G) = \bigcup_{j=1}^{n} \Lambda(x_j)$ ***и*** $\bigcap_{j=1}^{n} \Lambda(x_j) \overset{def}{=} V(X_G) \neq \varnothing$. ***Тогда область*** $V(X_G) \subseteq D(X_G)$ ***будем называть зоной общей ответственности (всех вершин графа).***

**Определение 20.** ***Если в*** $V(X_G) \subseteq D(X_G)$ ***существует такая точка*** $M_k$***, что для некоторого момента времени*** $\hat{t} \in T$ ***выполнено условие:*** $\prod_{j=1}^{n} \widetilde{p}_{\hat{t}}^{(j=k)} \neq 0$***, то соответствующую окрестность точки*** $M_k$ ***-*** $O_\varepsilon(M_k)$ ***будем называть областью сбора вершин графа. Если для какой-то реализации случайного процесса в момент времени*** $\hat{t} \in T$ ***имело место событие***

$$\left(\forall\ j = \overline{1, n}\right): \left\{x_j \in O_\varepsilon(M_k^{(j)}) \,\middle|\, t = \hat{t}\right\},$$ ***то*** $\hat{t}$ ***будем называть временем сбора вершин графа (в рамках данной реализации).***

Сформулируем очевидные, но важные для приложений факты.

**Теорема 2. Если пространственно нестационарный граф с континуальным числом состояний имеет зону общей ответственности, то $D(X_G)$ не допускает разбиения на зоны ответственности вершин.**

**Теорема 3. Если $D(X_G)$ графа $G$ допускает разбиение на зоны ответственности, то $G$ не имеет области сбора вершин.**

**Определение 21.** ***Пусть область*** $\Theta^{\langle q\rangle} \subseteq \bigcap_{i=1}^{q} \Lambda\left(x_{j_i}\right)$, ***где*** $\Theta^{\langle q\rangle} \neq \varnothing$, ***а*** $1 \le q \le n$. ***Тогда область*** $\Theta^{\langle q\rangle}$ ***будем называть зоной ответственности вершин*** $x_{j_i}$ $\left(i = \overline{1, q}\right)$, ***а значение*** $q$ ***- рангом зоны ответственности.***

**Определение 22.** ***Пусть*** $M_k^{(j)} \in \Lambda\left(x_j\right)$. ***Наименьшее значение величины*** $\Delta t$, ***при котором*** $P\left\{ x_j \in O_\varepsilon(M_k^{(j)}) \middle| t = \hat{t} + \Delta t \right\} \neq 0$ ***[при условии, что*** $\left(\hat{t} + \Delta t \in T\right)$***], будем называть временем доступа вершины*** $x_j$ ***графа*** $G$ ***в точку*** $M_k^{(j)}$ ***в момент*** $\hat{t}$ ***и обозначать следующим образом*** $\tilde{t}_{дост}^{x_j}\left(M_k^{(j)}; \hat{t}\right)$.

Ясно, что в том случае, когда $x_j \in O_\varepsilon(M_k^{(j)}) \middle| t = \hat{t}$, имеем $\tilde{t}_{дост}^{x_j}\left(M_k^{(j)}; \hat{t}\right) = 0$.

**Определение 23.** ***Будем говорить, что*** $\tilde{t}_{дост}^{x_j}\left(M_k^{(j)}; \hat{t}\right) = \infty$, ***если не существует такого значения величины*** $\Delta t$, ***при котором*** $P\left\{ x_j \in O_\varepsilon(M_k^{(j)}) \middle| t = \hat{t} + \Delta t \right\} \neq 0$ ***[при условии, что*** $\left(\hat{t} + \Delta t \in T\right)$***].***

**Определение 24.** ***Количество вершин*** $x_j$ ***графа*** $G$, ***у которых в момент времени*** $\hat{t} \in T$ $\tilde{t}_{дост}^{x_j}\left(M_k^{(j)}; \hat{t}\right) \neq \infty$ ***будем называть индексом доступности точки*** $M_k^{(j)}$ ***в сечении*** $\hat{t}$ ***и обозначать*** $\tilde{I}_{дост}\left(M_k^{(j)}; \hat{t}\right)$.

Определение моды совокупности точек для случая пространственно нестационарного П-В ДВНГ с континуальным числом состояний формально не отличается от данного ранее (см. определ. 17).

В заключение отметим важную особенность, относящуюся к дугам таких графов. Так как по построению в любом временном сечении $\hat{t} \in T$ каждая отдельная вершина $x_j$ графа $G$ идентифицируется с точностью до соответствующей окрестности $O_\varepsilon\left(M_k^{(j)}\right)$ некоторой точки $M_k^{(j)}$ [то есть $x_j \in O_\varepsilon\left(M_k^{(j)}\right)$ $\left(t = \hat{t}\right)$], то дуги $U_{ij}^{(l)}$, проведённые из какой-либо вершины $x_i$ в какую-либо вершину $x_j$, по сути являются множеством (*классом эквивалентности*) дуг, имеющих одинаковые характеристики и проведённых из любой точки множества $O_\varepsilon\left(M_k^{(i)}\right)$ в любую точку множества $O_\varepsilon\left(M_k^{(j)}\right)$.

**Заключение.** В данной статье проведено дальнейшее обобщение понятий динамического вероятностного нагруженного графа и пространственно – временного динамического вероятностного нагруженного графа. Введён новый объект – пространственно нестационарный пространственно – временной динамический вероятностный нагруженный граф. Рассмотрены особенности этого объекта. Приведена методика построения случайного процесса на базе таких графов. Даны определения сопутствующих понятий, проведен анализ их свойств.

***Библиографический список***

1. *Орлов Г.С.* Динамические вероятностные нагруженные графы. Определения, свойства, области применения. – Вестник РГРТУ. №1 (Выпуск 31). Рязань, 2010. – С. 48 – 57.

2. *Орлов Г.С.* Случайные процессы на базе обобщённых динамических вероятностных нагруженных графов. – Вестник РГРТУ. №3 (Выпуск 33). Рязань, 2010. – С. 60 – 63.

3. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. – М.: Наука. Главная редакция физико – математической литературы, 1984.

УДК 681.317.75:519.2

*Д.А. Перепелкин*

# АЛГОРИТМ АДАПТИВНОЙ УСКОРЕННОЙ МАРШРУТИЗАЦИИ НА БАЗЕ ПРОТОКОЛА OSPF ПРИ ДИНАМИЧЕСКОМ ДОБАВЛЕНИИ ЭЛЕМЕНТОВ КОРПОРАТИВНОЙ СЕТИ

*Предложен алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети, повышающий качество ее функционирования.*

***Ключевые слова:*** *адаптивная ускоренная маршрутизация, протокол OSPF, динамические изменения, алгоритмы маршрутизации, корпоративные сети.*

**Введение.** Цель работы – разработка нового эффективного алгоритма поиска оптимальных маршрутов на базе протокола OSPF при динамическом добавлении элементов корпоративной сети, повышающего качество ее функционирования. На современном этапе развития и использования корпоративных сетей наиболее актуальное значение приобрели оценка производительности сети и повышение качества ее функционирования. Также особое внимание уделяется оптимизации уже существующих или планируемых к созданию сетей. Корпоративные сети стали определяющим компонентом в информационной стратегии большинства организаций, и недостаточное внимание к оценке ее мощности и планированию привело к тому, что для поддержки современных приложений многие сети необходимо заново проектировать, а во многих случаях и заменять.

Совершенствование сетевых технологий требует обеспечения качественного обслуживания передаваемого трафика [1]. Изменения характеристик каналов связи, модификация структуры сети, включение в нее новых узлов и линий связи приводят к полному пересчету таблиц маршрутизации. Одним из решений повышения качества функционирования корпоративных сетей является точное определение оптимальных маршрутов передачи данных и быстрое переключение более загруженных каналов связи на другие – свободные каналы, при динамическом добавлении элементов корпоративной сети.

**Теоретическая часть.** Для повышения качества функционирования корпоративных сетей наиболее важной задачей является выбор эффективного алгоритма маршрутизации, который будет обеспечивать поиск оптимальных маршрутов с учетом различных свойств той или иной корпоративной сети. Для эффективной передачи по сети пакетов данных необходимо, чтобы алгоритм маршрутизации учитывал требования к уровню качества обслуживания (Quality of Service, QoS). Алгоритмы должны обеспечивать рациональность маршрута, быть простыми и гибкими с точки зрения возможных изменений в структуре корпоративной сети и нагрузках на линиях связи, к тому же они не должны порождать интенсивный служебный трафик.

В настоящее время широкое применение получили алгоритмы адаптивной маршрутизации. Они обеспечивают автоматическое обновление таблиц маршрутизации после изменения конфигурации сети. Используя алгоритмы адаптивной маршрутизации, маршрутизаторы могут собирать информацию о топологии связей в сети и оперативно реагировать на все изменения в ее конфигурации.

В современных условиях корпоративная сеть рассматривается как отдельная автономная система, внутри которой используется свой протокол маршрутизации. Такой протокол принято называть протоколом внутренней маршрутизации.

Протоколы внутренней маршрутизации делятся на две группы, каждая из которых связана с одним из следующих типов алгоритмов:

- дистанционно-векторные алгоритмы (Distance Vector Algorithm, DVA);
- алгоритмы состояния каналов (Link State Algorithm, LSA).

В алгоритмах дистанционно-векторного типа каждый маршрутизатор периодически и широковещательно рассылает по сети вектор, компонентами которого являются расстояния от

данного маршрутизатора до всех известных ему сетей. Под расстоянием обычно понимается число транзитных узлов, однако могут быть использованы и другие показатели метрики. Дистанционно-векторные алгоритмы хорошо работают только в небольших сетях. В крупных же сетях они загружают линии связи интенсивным широковещательным трафиком [2].

Алгоритм состояния каналов обеспечивает каждый маршрутизатор информацией, достаточной для построения точного графа связей сети. Все маршрутизаторы работают на основании одинаковых графов, что делает процесс маршрутизации более устойчивым к изменениям конфигурации. «Широковещательная» рассылка (то есть передача пакета всем непосредственным соседям маршрутизатора) используется здесь только при изменениях состояния связей. Вершинами графа являются как маршрутизаторы, так и объединяемые ими сети. Распространяемая по сети информация состоит из описания связей различных типов: маршрутизатор - маршрутизатор, маршрутизатор – сеть.

Протоколом, основанным на алгоритме состояния каналов стека TCP/IP, является протокол OSPF (Open Shortest Path First).

**В протоколе OSPF** каждый маршрутизатор самостоятельно решает задачу оптимизации маршрутов. В процессе выбора оптимального маршрута анализируется неориентированный граф сети. Метрики выбранного пути могут характеризоваться следующими параметрами качества обслуживания (QoS):

1) пропускной способностью канала;
2) задержкой (время распространения пакета);
3) числом дейтограмм, стоящих в очереди для передачи;
4) загрузкой канала;
5) требованиями безопасности;
6) типом трафика;
7) числом шагов до цели;
8) возможностями промежуточных связей.

Протокол OSPF производит расчет метрики отдельно для каждого вида сервиса TOS (type of service). Число TOS определяет многообразие метрик, соответствующих видам сервиса, для данного канала. Последовательность описания метрик задается величиной кода TOS. Значения кодов TOS, принятых в протоколе OSPF, приведены в таблице 1.

В протоколе OSPF используется принцип контроля состояния канала (link-state protocol), а метрика представляет собой оценку эффективности связи в этом канале: чем меньше метрика, тем эффективнее организация связи. В простейшем случае метрика маршрута может равняться его длине в пересылках (hops). Но в общем случае значения метрики могут определяться в гораздо более широком диапазоне.

**Таблица 1**

| OSPF - код | TOS - коды | TOS (RFC 1349) |
|---|---|---|
| 0 | 0000 | Обычный сервис |
| 2 | 0001 | Минимизация денежной стоимости |
| 4 | 0010 | Максимальная надежность |
| 8 | 0100 | Максимальная пропускная способность |
| 16 | 1000 | Минимальная задержка |

Выбор оптимального маршрута в протоколе OSPF определяется по алгоритму Дейкстры. Трудоемкость построения таблиц маршрутизации с использованием этого алгоритма составляет порядка $O(N^2)$, где $N$ – число маршрутизаторов в корпоративной сети.

Метрика, оценивающая пропускную способность канала, определяется, например, компанией Cisco, как количество секунд, нужное для передачи 100 Мбит. Имеется следующая формула для вычисления метрики доставки информации через каналы сети OSPF:

метрика = $10^8$ / скорость передачи в битах в секунду.

Метрика протокола OSPF для используемых в корпоративных сетях каналов передачи данных приведена в таблице 2.

**Таблица 2**

| Наименование канала | Скорость канала | Метрика канала |
|---|---|---|
| Канал 1 | 100 Мбит/с | 1 |
| Сеть Ethernet / 802.3 | 10 Мбит/с | 10 |
| Тракт E1 | 2,048 Мбит/с | 48 |
| Тракт T1 | 1,544 Мбит/с | 65 |
| Канал 2 | 64 Кбит/с | 1562 |
| Канал 3 | 56 Кбит/с | 1785 |
| Канал 4 | 19,2 Кбит/с | 5208 |
| Канал 5 | 9,6 Кбит/с | 10416 |

В работе [3] предложен алгоритм парных переходов, позволяющий за счет сбора дополнительной информации учесть возможные изменения конфигурации корпоративной сети и не производить полный пересчет маршрутных таблиц. Это позволило снизить трудоемкость расчета таблиц маршрутизации до величины порядка $O(k \cdot N)$, где $k$ – число фактически выполненных парных переходов.

При динамическом добавлении элементов

корпоративной сети использование данного алгоритма оказывается неэффективным, так как трудоемкость расчета дополнительной информации для осуществления парного перехода составляет $O(N^2\log_2 N)$, что оказывается выше соответствующей оценки трудоемкости алгоритма Дейкстры.

Разработка новых, более эффективных алгоритмов адаптивной маршрутизации позволяет уменьшить трудоемкость построения таблиц маршрутизации в корпоративных сетях, использующих в своей работе протокол OSPF.

**Разработка алгоритма.** Для повышения качества функционирования корпоративных сетей на базе протокола OSPF предложен алгоритм адаптивной ускоренной маршрутизации, который позволяет уменьшить трудоемкость построения таблиц маршрутизации до величины $O(N)$ при динамическом добавлении элементов корпоративной сети.

Представим корпоративную сеть в виде неориентированного взвешенного связного графа $G = (V,E,W)$, где $V$ – множество вершин, $|V| = N$, $E$ – множество ребер, $|E| = M$, $W$ – множество весов ребер.

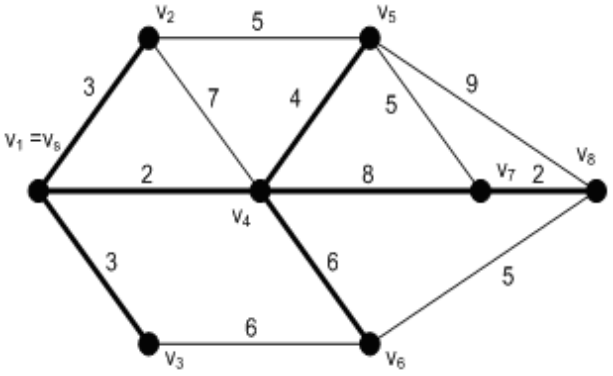



**Рисунок 1 – Граф G корпоративной сети**

Пусть на графе $G$ в некоторый момент времени уже решена задача поиска кратчайших путей до всех вершин множества $V_s = V\backslash\{v_s\}$ из начальной вершины $v_s$, т.е. построено дерево кратчайших путей с корнем в вершине $v_s$. Обозначим это дерево как $T_g$. На рисунке 1 жирными линиями обозначено построенное дерево кратчайших путей.

Рассмотрим множество ребер $E$ графа $G$. По признаку вхождения ребер в дерево $T_g$ можно разделить исходное множество $E$ на два подмножества: $E_T \in T_g$ и $E_R \notin T_g$, $E_T \cup E_R = E$.

**Множество ребер дерева $E_T$** – множество ребер дерева $T_g$ для графа $G$. Для заданного графа $G$, согласно свойству дерева, мощность множества $E_T$ будет равняться мощности множества $V$ минус единица $|E_T|=|V| - 1$.

**Множество ребер замены для дерева $E_R$** – множество ребер графа $G$, не вошедших в дерево $T_g$. При соответствующих условиях некоторое ребро $e_{i,j}\in E_R$, инцидентное вершинам $v_i$ и $v_j$, может перейти во множество ребер дерева $E_T$, заменив собой некоторое ребро $e_{k,p}\in E_T$. При этом инцидентность ребра $e_{k,p}$ вершине $v_i$ или $v_j$ является обязательным условием. В свою очередь ребро $e_{i,j}$ перейдет во множество $E_R$.

Будем называть такие переходы **парными переходами** и обозначать $e_{i,j} – e_{k,p}$.

Во множестве $E_R$ можно выделить 2 подмножества.

**Множество ребер замены $E_S$ для дерева** – это такое подмножество множества $E_R$, элементы-ребра которого участвуют, по крайней мере, в одном отношении парного перехода.

**Множество непарных ребер $E_P$** – это такое подмножество множества $E_R$, элементы-ребра которого не участвуют ни в одном отношении из множества $R$.

В общем случае множество $E_P$ может быть пустым $|E_P| = 0$. Множество $E_S$ будет пустым только при условии, что исходный связный граф $G$ является деревом, и задача поиска кратчайших путей в этом случае лишена смысла.

Для каждого ребра $e_{i,j}\in E$ на шкале значений весов определена точка вхождения в дерево $w_{i,j}^{\ t}$ и точка вхождения во множество замены $w_{i,j}^{\ s}$, причем $w_{i,j}^{\ t} \le w_{i,j}^{\ s}$.

Так для графа G, представленного на рисунке 1, множество ребер дерева составляет $E_T = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{4,5}, e_{4,6}, e_{4,7}, e_{7,8}\}$; множество ребер замены $E_S = \{e_{2,5}, e_{2,4}, e_{3,6}, e_{5,7}, e_{6,8}\}$; множество непарных ребер будет $E_P = \{e_{5,8}\}$. Если рассмотреть ребро $e_{4,5}$, то для него точка вхождения в дерево будет составлять 6, а точка вхождения во множество замены – 13. При этом данное ребро находится в отношении парного перехода с ребром $e_{2,5}$, а после попадания $e_{4,5}$ во множество непарных ребер эта парная перестановка примет вид $e_{2,5}$ - $e_{5,7}$.

Обозначим множество путей до вершины $v_i$ из исходной вершины $v_s$ через $\Pi_i$, где элемент множества $\pi_{i,k}\in\Pi_i$ будет множеством не повторяющихся ребер $e_{i,j}\in E$, образующих вместе путь, соединяющий $v_s$ и $v_i$. Для всех $\pi_{i,k}\in\Pi_i$ поставим в соответствие некоторое число, равное сумме весов, входящих в него ребер, т.е. длину пути $d_{i,k}\in D_i$, где $D_i$ представляет собой множество оценок кратчайших путей до вершины $v_i$ из исходной вершины $v_s$. Кратчайший путь до вершины $v_i$ будем обозначать $\pi_i$, а его оценку длины – $d_i$.

Для разработки алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов

корпоративной сети сформулируем следующие теоремы.

**Теорема 1.** При добавлении некоторой вершины $V_i$ для всех вершин не инцидентных вершине $i$ (т.е. не имеющих ребра $e_{i,k}$) кратчайшие пути и их оценки окажутся без изменения.

**Доказательство.** Новая вершина $V_i$ не входит в кратчайший путь к некоторой вершине $V_j$. Следовательно, и в пути к этой вершине $\pi_{v,j}$ отсутствует ребро $e_{i,k}$. Поэтому новый путь к вершине $V_j$, содержащий данное ребро, будет иметь оценку $d_{j,i} > = d_j$ и дерево кратчайших путей не изменится для всех вершин, не инцидентных вершине $V_i$. Теорема доказана.

**Следствие 1.** При добавлении некоторой вершины $V_k$, имеющей ребро с вершиной $V_i$ и $V_j$, если $d_i < d_j$, то дерево кратчайших путей до вершины $V_i$ не изменится.

**Следствие 2.** При добавлении некоторой вершины $V_k$, имеющей ребро с вершиной $V_i$ и $V_j$, если $d_i < d_j$, то необходимо определить новые кратчайшие пути для множества вершин, инцидентных вершине $V_k$, кроме вершины $V_i$.

Данному случаю соответствует граф, представленный на рисунке 2, при добавлении вершины $V_9$.

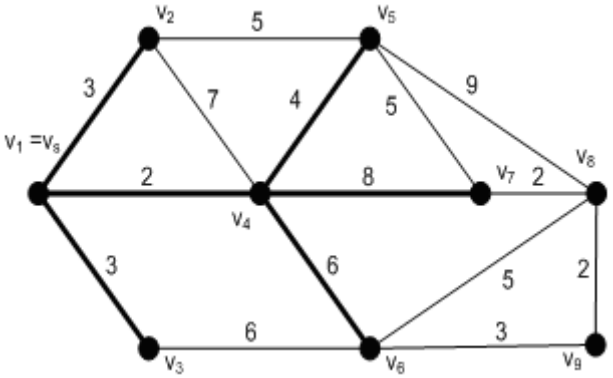



**Рисунок 2 - Добавление вершины $V_9$**

На рисунке 2 жирными линиями обозначено дерево кратчайших путей, которое не требует изменения.

То есть при добавлении вершины $V_9$ дерево кратчайших путей для вершин $V_2$, $V_3$, $V_4$, $V_5$, $V_6$, $V_7$ согласно теореме 1 не изменится. Для добавленной вершины $V_9$, а также вершины $V_8$ необходимо построить новые кратчайшие пути с учетом ребра $e_{6,9}$ с весом $w_{6,9} = 3$ и ребра $e_{9,8}$ с весом $w_{9,8} = 2$. Кратчайший путь до вершины $V_8$ не изменится и составит $\pi_8 = \{e_{1,4}, e_{4,7}, e_{7,8}\}$, а его оценка $d_8 = 12$. Кратчайший путь до вершины $V_9$ составит $\pi_9 = \{e_{1,4}, e_{4,6}, e_{6,9}\}$, а его оценка $d_9 = 11$.

**Теорема 2.** При добавлении нового ребра $e_{i,j}$, инцидентного вершинам $i$ и $j$, причем $i$ лежит выше чем $j$ по дереву иерархии, с весом $w_{i,j}$, то без изменения окажутся кратчайшие пути и их оценки для вершин множества $V^{(Vi)}$.

**Доказательство.** Пусть ребро $e_{i,j}$ входит в путь $\pi_{i,k} < > \pi_i$ и $\pi_{j,p} < > \pi_j$. Если данное ребро не уменьшает оценок обеих инцидентных ему вершин $v_i$ и $v_j$, то есть $d_{i,k} > = d_i$ и $d_{j,p} > = d_j$, дерево кратчайших путей не изменится, так как ребро $e_{i,j}$ оказывает влияние, прежде всего, на инцидентные ему вершины множества $V$. Так как существовавшие до изменения пути до вершин $i$ и $j$ имели меньшую длину, то ребро $e_{i,j}$ не включается, и дерево кратчайших путей не изменится. Если уменьшилась оценка, какой – либо инцидентной вершины, например $v_j$, то эта оценка $d_{j,p}$ будет оценкой кратчайшего пути до вершины $v_j$ и ребро $e_{i,j}$ войдет в новое дерево кратчайших путей, так как не существует другого кратчайшего пути $\pi_j$ до вершины $v_j$, кроме пути $\pi_{j,p}$, содержащего ребро $e_{i,j}$. Этот кратчайший путь $\pi_{j,p}$ не будет существовать, если не будут существовать кратчайшие пути до всех промежуточных вершин $v_k \in V^{(Vi)}$ этого пути. Невозможно будет сказать останутся ли неизмененными кратчайшие пути до остальных вершин графа. Теорема доказана.

**Следствие.** При добавлении нового ребра $e_{i,j}$, инцидентного вершинам $i$ и $j$, с весом $w_{i,j}$ при не изменении оценок $d_i$ и $d_j$ дерево не изменится. Если изменяется оценка какой–либо вершины $v_i$ или $v_j$, то необходимо определить новые кратчайшие пути для множества вершин, не принадлежащих множеству $V^{(Vi)}$.

Данному случаю соответствует граф, представленный на рисунке 3, при добавлении ребра $e_{6,7}$.

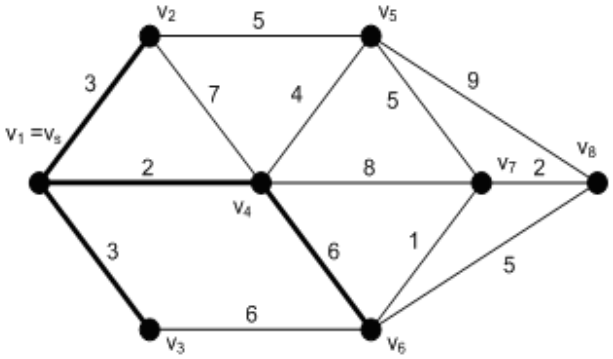



**Рисунок 3 - Добавление ребра $e_{6,7}$**

На рисунке 3 жирными линиями обозначено дерево кратчайших путей, которое не требует изменения.

То есть при добавлении ребра $e_{6,7}$ с весом $w_{6,7} = 1$ дерево кратчайших путей для вершин $V_2$, $V_3$, $V_4$, $V_6$ не изменится согласно теореме 2. Для вершин $V_5$, $V_7$, $V_8$ необходимо определить новые кратчайшие пути с учетом добавленного ребра $e_{6,7}$. Кратчайший путь до вершины $V_5$ составит $\pi_5 = \{e_{1,4}, e_{4,5}\}$, а его оценка $d_5 = 6$. Кратчайший

путь до вершины $V_7$ составит $\pi_7 = \{e_{1,4}, e_{4,6}, e_{6,7}\}$, а его оценка $d_7 = 9$. Кратчайший путь до вершины $V_8$ составит $\pi_8 = \{e_{1,4}, e_{4,6}, e_{6,7}, w_{7,8}\}$, а его оценка $d_8 = 11$.

Использование доказанных выше теорем позволяет разработать алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF, уменьшающий размерность задачи поиска кратчайших путей при динамическом добавлении элементов корпоративной сети.

Рассмотрим работу алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF в корпоративных сетях. Укрупненная схема алгоритма имеет следующий вид.

*Шаг 1.* Первоначальная инициализация исходных данных. Используя пакет HELLO протокола OSPF, определить веса линий связи $w_{i,j}$.

*Шаг 2.* Построить дерево оптимальных маршрутов корпоративной сети.

*Шаг 3.* Для вершины, являющейся листом дерева, производится поиск всех парных переходов без ограничений. Эти списки для удобства дальнейшей работы привязываются к вершине, инцидентной рассматриваемому ребру и расположенной ниже по иерархии.

*Шаг 4.* Если вершина не является листом дерева, то вычисляются парные переходы для этой вершины и выбираются лучшие значения потенциалов парных переходов для потомков вершины и собственных парных переходов. Подобная процедура выполняется для формирования списков парных переходов в случае динамического добавления новых элементов в структуру корпоративной сети.

*Шаг 5.* Для каждой вершины формируется полный список парных переходов. Число элементов в каждом из этих списков не превышает количества вершин графа. Такое решение позволяет отказаться от предварительной сортировки потенциалов или приращений для парных переходов без значительного усложнения алгоритма обработки изменения.

*Шаг 6.* Для каждого ребра графа корпоративной сети определить точку вхождения в дерево оптимальных маршрутов и точку вхождения во множество замены.

*Шаг 7.* Определить, есть ли пакеты на передачу:

а) если да, то перейти к шагу 8;

б) иначе – к шагу 16.

*Шаг 8.* Используя поля «Время жизни» и «Контрольная сумма заголовка» протокола IP определить, требуется ли уничтожить (отбросить) данный пакет:

а) если да, то перейти к шагу 22;

б) иначе - к шагу 9.

*Шаг 9.* Используя поле «Тип сервиса», определить, требуется ли создание виртуального соединения:

а) если да, то перейти к шагу 10;

б) иначе – к шагу 12.

*Шаг 10.* Организовать виртуальное соединение, послав первый пакет «запрос вызова» адресату:

а) если пакет согласия на соединение от адресата пришел, то перейти к шагу 13;

б) если адресат отклонил вызов, то перейти к пункту 12.

*Шаг 11.* а) установить флаг передачи;

б) послать адресату пакет ликвидации соединения;

в) получить пакет подтверждения рассоединения от адресата.

*Шаг 12.* Проверка флага передачи:

а) если флаг установлен, то перейти к шагу 22;

б) иначе – к шагу 7.

*Шаг 13.* Используя поле «Тип сервиса», определить необходимую таблицу маршрутизации с учетом желаемого уровня качества обслуживания:

а) если таблица 1, то перейти к шагу 14;

б) иначе – к шагу 15.

*Шаг 14.* а) передать пакет, используя первую таблицу маршрутизации;

б) перейти к шагу 11.

*Шаг 15.* а) передать пакет, используя вторую таблицу маршрутизации;

б) перейти к шагу 11.

*Шаг 16.* Анализируя полученную протоколом OSPF информацию, определить, произошло ли добавление новых элементов в структуру корпоративной сети:

а) если да, то перейти к шагу 17;

б) иначе - к шагу 7.

*Шаг 17.* а) *если добавилась новая вершина $V_k$, имеющая ребра с вершинами $V_i$ и $V_j$, причем $d_i < d_j$ (i-вершина расположена ниже по иерархии в дереве кратчайших путей), то:*

1) дерево кратчайших путей до вершины $V_i$ не изменится;

2) для всех вершин, не инцидентных вершине $V_i$, дерево кратчайших путей не изменится;

3) потенциал вершины $V_j$ (путь $d_j$ до вершины $V_j$) с учетом новых связей $d_{i,k}$ и $d_{k,j}$ не изменился. Первоначальное дерево кратчайших путей не изменится. Из возможных путей $d_i + d_{i,k}$ и $d_j + d_{k,j}$ выбрать путь минимальной длины и включить его в дерево кратчайших путей. Перейти к пункту 5;

4) потенциал вершины $V_j$ (путь $d_j$ до вершины $V_j$) с учетом новых связей $d_{i,k}$ и $d_{k,j}$ уменьшился (то есть $d_i + d_{i,k} + d_{k,j} < d_j$).

Включить в дерево кратчайших путей ребра $d_{i,k}$ и $d_{k,j}$. Для всех вершин $V_m$, инцидентных вершине $V_j$, определить их новые потенциалы. Если потенциал до вершины $V_m$ уменьшился, то ребро $d_{j,m}$ включить в дерево кратчайших путей, исключив первоначальное ребро до вершины $V_m$;

5) для всех ребер, включенных в дерево кратчайших путей, определить возможные ребра для парных переходов и рассчитать для них точки вхождения во множество ребер замены и в дерево кратчайших путей;

б) *если добавилось новое ребро $e_{i,j}$ с весом $w_{i,j}$, причем вершина $V_i$ располагается ниже по иерархии в дереве оптимальных маршрутов, чем вершина $V_j$, то:*

1) оценка длины маршрута для вершины $V_j$ не изменилась. Дерево оптимальных маршрутов не изменится. Определить, находится ли ребро $e_{i,j}$ в отношении парного перехода к какому-либо ребру для вершины $V_j$. Если да, то включить его в множество вершин, состоящих в отношении парного перехода к вершине $V_j$. Если нет, то оставить все без изменения;

2) оценка длины маршрута для вершины $V_j$ уменьшилась. Включить ребро $e_{i,j}$ в дерево оптимальных маршрутов. Для всех вершин, кроме $V_i$, инцидентных вершине $V_j$, рассчитать новые оценки маршрутов. Для тех вершин, у которых оценки уменьшились, соответствующее инцидентное ребро включить в дерево оптимальных маршрутов, удалив из дерева ребро, входящее в оптимальный маршрут до этой вершины.

*Шаг 18.* Используя список парных переходов, определить, требуется ли сделать парный переход:

а) если да, то перейти к шагу 19;

б) иначе – к шагу 20.

*Шаг 19.* Для каждой вершины, у которой в список возможных маршрутов входит ребро с изменившейся метрикой, определить путь минимальной длины и поместить его в дерево кратчайших путей.

*Шаг 20.* Построить новое дерево оптимальных маршрутов с учетом изменений.

*Шаг 21.* Сформировать таблицы маршрутизации.

*Шаг 22.* Проверка окончания работы маршрутизатора:

а) если да, то перейти к шагу 23;

б) иначе – сбросить флаг передачи и перейти к шагу 7.

*Шаг 23.* Завершение работы маршрутизатора.

**Результаты моделирования.** Для подтверждения правильности предложенного алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF при динамическом добавлении элементов корпоративной сети разработана программа имитационного моделирования процессов маршрутизации.

При разработке основное внимание уделялось корректности предлагаемого алгоритма и размерности решаемой задачи.

Для каждого испытания на множестве обработанных изменений выбирались минимальное, максимальное и среднее значения размерности задачи, выраженные через количество вершин, для которых необходим поиск кратчайшего пути. Для каждого эксперимента были найдены значения оценок математического ожидания и среднего квадратичного отклонения числа изменений. Для алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF определялось число фактически выполненных парных переходов при динамическом добавлении элементов корпоративной сети.

На рисунках 4 – 6 представлены результаты моделирования алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF.

В таблице 3 приведены обобщенные статистические характеристики для средней размерности задачи. В представленной таблице через СКО обозначено значение оценки среднего квадратичного отклонения.

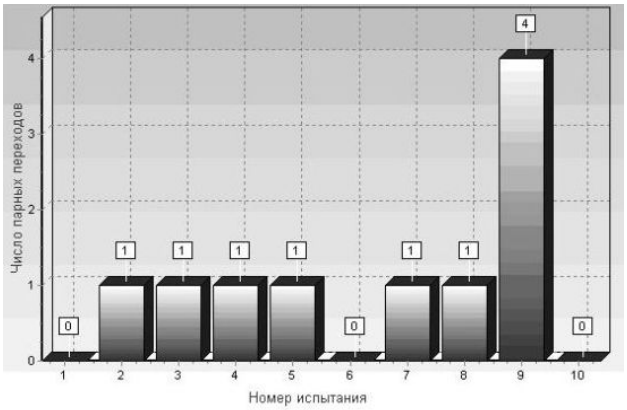



**Рисунок 4 - Число изменений дерева в графе из 10 вершин**

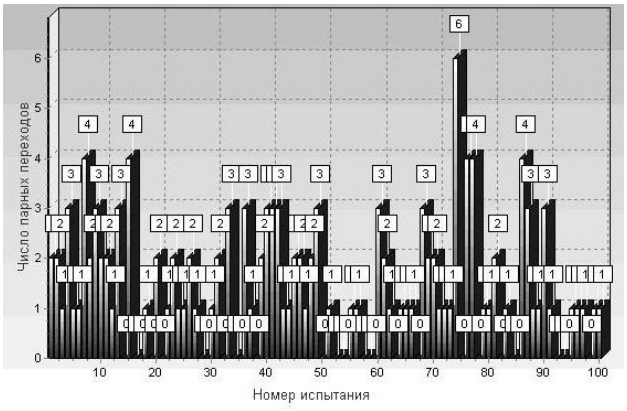



**Рисунок 5 - Число изменений дерева в графе из 100 вершин**

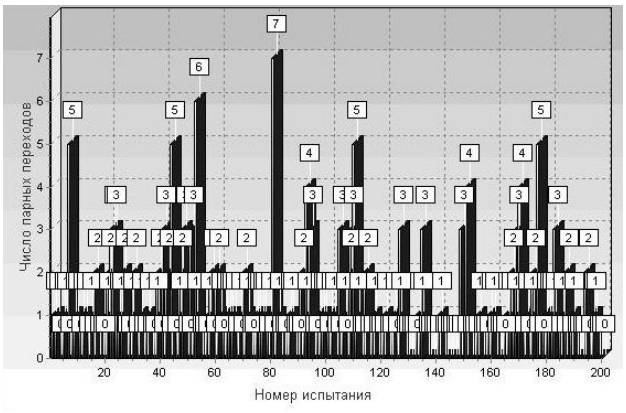



**Рисунок 6 - Число изменений дерева в графе из 200 вершин**

**Таблица 3**

| Число вершин графа | Min значение | Max значение | Значение оценки МО | Значение оценки СКО |
|---|---|---|---|---|
| 10 | 0 | 0,415 | 0,1960 | 0,0725 |
| 100 | 0,03 | 0,601 | 0,0262 | 0,1241 |
| 200 | 0 | 0,723 | 0,0179 | 0,1052 |

Были проведены исследования графов, состоящих из 10, 100 и 200 вершин. Исследование разработанного алгоритма адаптивной ускоренной маршрутизации на базе протокола OSPF показало, что максимальное значение числа изменений существенно ниже размерности для каждого из рассмотренных графов, а значение оценки математического ожидания не превышает единицы. Более того, обнаружена тенденция уменьшения значения оценки математического ожидания числа изменений дерева оптимальных маршрутов с увеличением количества вершин графа.

При динамическом добавлении элементов корпоративной сети разработанный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF в отличие от алгоритма Дейкстры позволяет производить перестроение таблиц маршрутизации не полностью, а только той ее части, в которой произошли изменения. При этом трудоемкость изменения дерева кратчайших путей является линейной функцией от числа вершин и определяется выражением $O(N)$. То есть при добавлении элементов корпоративной сети необходимо один раз просмотреть все вершины графа сети и на основе предварительно собранной информации о парных переходах выполнить изменения для вычисления новых оптимальных маршрутов. Если рассмотреть полученные результаты при моделировании предложенного алгоритма адаптивной ускоренной маршрутизации и сравнить их с алгоритмом Дейкстры, то видно:

1) для графа из 10 вершин максимальное число парных переходов составляет четыре изменения (= 4), при этом для алгоритма Дейкстры при каждом добавлении элементов трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 10^2 = 100$ элементарных операций;

2) для графа из 100 вершин максимальное число парных переходов составляет шесть изменений (= 6), при этом для алгоритма Дейкстры при каждом добавлении элементов трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 100^2 = 10000$ элементарных операций;

3) для графа из 200 вершин максимальное число парных переходов составляет семь изменений (=7), при этом для алгоритма Дейкстры при каждом добавлении элементов трудоемкость построения таблицы маршрутизации составляет $O(N^2) = 200^2 = 40000$ элементарных операций.

На основе этого можно сделать вывод, что предложенный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF является эффективным при поиске оптимальных маршрутов в условиях динамических изменений в структуре корпоративной сети и нагрузках на линиях связи за счет использования дополнительной информации о возможных парных переходах.

**Заключение.** Разработанный алгоритм адаптивной ускоренной маршрутизации на базе протокола OSPF позволяет повысить качество функционирования корпоративных сетей за счет уменьшения трудоемкости построения таблиц маршрутизации до величины порядка $O(N)$ при динамическом добавлении элементов корпоративной сети.

### *Библиографический список*

1. *Куракин Д.В.* Маршрутизаторы для глобальных телекоммуникационных сетей и реализуемые в них алгоритмы // Информационные технологии.– 1996. №2.

2. *Олифер В.Г., Олифер Н.А.* Основы компьютерных сетей – СПб.: Питер, 2009. – 352 с.

3. *Уваров Д.В., Перепелкин А.И.* Построение дерева кратчайших путей на основе данных о парных переходах // Системы управления и информационные технологии. №4 (16), Москва-Воронеж. 2004. С.93-96.

УДК 004.738.52

*Н.В. Неелова, А.А. Сычугов*

# СРАВНЕНИЕ РЕЗУЛЬТАТОВ ДЕТЕКТИРОВАНИЯ ДУБЛЕЙ МЕТОДОМ ШИНГЛОВ И МЕТОДОМ ДЖАККАРДА

*Разработана математическая модель и метод вычисления схожести текстовых объектов на основе коэффициента Джаккарда с учетом синонимических замен. Реализована программа на основе разработанного метода. Проведено сравнение на сформированных текстовых кластерах с заданной точностью результатов предложенной модели и модели, использующей метод шинглов. Рассмотрены преимущества и недостатки разработанного метода.*

***Ключевые слова:*** *нечеткие дубли, метод шинглов, схожесть Джаккарда, вычисление схожести строк, полнота вычисления.*

**Введение.** Цель работы – сравнить результаты определения нечетких дублей методом шинглов и методом Джаккарда без и с учетом синонимических замен по метрике «полнота», сделать выводы об использовании последнего в промышленных масштабах.

Одним из важных факторов, определяющих качество ответа поисковой системы, является количество дубликатов, показываемых на страницах результата поиска на запрос пользователя. Дубликаты не только «засоряют» выдачу, они также увеличивают базы поисковых архивов за счет избыточной информации, осложняют вычисления при объединении новостных сообщений в сюжеты, способствуют принятию решения о первоисточнике. Поэтому проблема обнаружения веб-копий является актуальной для поисковых систем и приложений, учитывающих схожесть документов.

Источники дублей в сети самые различные: от технических недоработок сайта до незаконного копирования информации или обмана поисковых систем. Более подробно классификация рассмотрена в [1], также в указанной статье выделены методы выявления схожих документов.

Кластеризация документов-дубликатов осуществляется на основе соотношения числовой меры сходства двух документов. Выделяют два подхода детектирования дублей: on-line и off-line. Если при online методах вычисление веб-копий осуществляется в процессе формирования ответа поисковыми системами на запрос пользователя, то off-line методы направлены на чистку репозиторий данных поисковых машин.

В настоящей работе рассматриваются два частных метода off-line кластеризации. Первый – метод «шинглов», является представителем синтаксического подхода (выбор последовательностей символов), второй – метод, основанный на схожести Джаккарда – лексический (выбор представительных слов).

Схема шинглирования хороша тем, что дает высокую точность определения дублей, но не высокую полноту, однако на данном подходе основано сравнение текстов в поисковых системах [2, 3, 4]. Он позволяет значительно уменьшить размер образа документа и упростить задачу кластеризации. Согласно исследованиям в [4] функция схожести Джаккарда также имеет незначительно худшие результаты полноты в сравнении с другими алгоритмами, однако он имеет лучшую производительность. Неустойчивость обоих методов проявляется при наличии орфографических ошибок и синонимических замен.

В связи с этим в исследовании [5] была предложена модель выявления схожести объектов на основе лексического метода Джаккарда с учетом синонимических замен (в том числе орфографические ошибки). Данная доработка, учитывающая возможность преднамеренного и случайного редактирования текста, не может быть реализована на методе шинглов из-за специфики алгоритма.

В данном исследовании была поставлена задача сравнить широко используемый метод шинглов и предложенный алгоритм, основанный на коэффициенте Джаккарда, с внедренной функцией учета синонимических замен. Сравнение производилось на совокупности текстов, полученных из оригиналов при помощи синонимических рядов. В результате были сделаны

выводы о возможности использования предложенного метода в промышленных масштабах вместо метода шинглов.

**Метод шинглов.** Шинглами называют последовательности слов документа, перекрывающие друг друга. Размер шингла не должен быть слишком маленьким, для того чтобы свести к минимуму случайные повторы, но и не должен быть слишком большим, чтобы минимальное изменение текста не изменяло значения контрольных сумм.

Таким образом, для каждого n-словия текста рассчитывается контрольная сумма. N-словия идут внахлест, с перекрытием (см. рисунок 1).

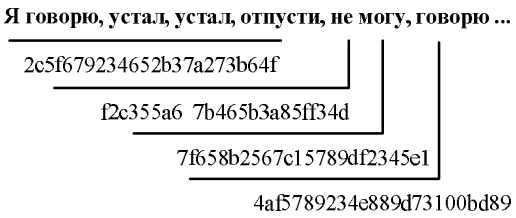



**Рисунок 1 – Формирование «шинглов»**

Далее из всего множества контрольных сумм выбирается подмножество фиксированного размера с использованием случайных перестановок, описанных в работе [6]. При этом вероятность, что минимальные элементы в перестановке контрольных сумм на множествах шинглов документов A и B (пусть эти множества обозначаются как $F_A$ и $F_B$ соответственно) совпадут, равна мере сходства этих документов *sim* (A, B):

$$sim(A,B) = P[\min\{\pi(F_A)\} = \min\{\pi(F_B)\}] = \frac{|F_A \cap F_B|}{|F_A \cup F_B|}. \quad (1)$$

Реализация перестановок шинглов возможна с помощью произведения векторов строк, представляющих множество контрольных сумм шинглов документа в двоичном виде, на случайные бинарные матрицы, соответствующие перестановкам (см. [7]).

Из формулы (1) следует, что мера сходства принадлежит отрезку [0;1], а равенство *sim*(A, A)=1, означающее 100% сходство двух идентичных документов, всегда верно. Таким образом, чем больше значение *sim* (A, B), тем больше сходство двух документов.

**Функция схожести Джаккарда.** Это функция, оценивающая меру сходства двух документов путем соотношения множеств термов документов. Мера Джаккарда определяется как:

$$sim\ (x, y) = \frac{|\{x_1,...,\ x_v\} \cap \{y_1,...,\ y_w\}|}{|\{x_1,...,\ x_v\} \cup \{y_1,...,\ y_w\}|}, \quad (2)$$

где *x, y* – документы, схожесть которых устанавливается; *v, w* – количество лексем в текстах документов *x* и *y* соответственно; $\{x_1,...,x_v\}$, $\{y_1,...,y_w\}$ – множество лексем текстов *x* и *y*.

Из формулы (2) следует, что область значений функции от 0 до 1 и достигает максимума при полном совпадении двух множеств. Мера Джаккарда чрезвычайно проста, однако согласно источнику [8] можно выделить два недостатка. Во-первых, не учитывается разница в размере сравниваемых документов, а, во-вторых, не используется информация о частоте употребления термов, составляющих документы. Также были указаны проблемы орфографических ошибок, коротких строк и синонимических замен. Однако согласно исследованию [4] функция Джаккарда более эффективна при использовании в промышленных масштабах сравнительно с другими методами online фильтрации дубликатов веб-документов, так как имеет достаточно высокую производительность при незначительно худших результатах полноты.

**Модель вычисления схожести текстовых объектов на основе функции Джаккарда с предварительной обработкой.** Если $\bar{x} = (x_1,...,x_d)$ – объект предметной области, определенный через вектор характеристик, где $x_1...x_d$ – характеристики предметной области, а *d* – их количество. В данном случае объектом предметной области выступает текст обрабатываемого документа, а характеристикой – его лексемы. $\bar{X} = (\bar{x}_1,...,\bar{x}_n)$ – множество объектов предметной области, i-й объект из $\bar{X}$ определяется как $\bar{x}_i = (x_{i1},...,x_{id})$.

Пусть также каждой *m*-й характеристике объекта $\bar{x} = (x_1,...,x_d)$ можно поставить в соответствие значение из множества $W = \{w_m \mid w_m = IDF(m)\}$ – множество IDF (инверсная частота термина) значений всех слов коллекции. Другими словами, каждое слово документа имеет свое IDF. Путем эмпирических исследований устанавливается отрезок значимых слов: $[w_{\min}; w_{\max}]$, где $w_{\min}$ и $w_{\max}$ – минимальная и максимальная пропускная граница характеристик объекта. Таким образом, образ объектов будут составлять только те термы, которые прошли значимый отрезок. Другими словами, устраняется шумовое влияние незначимых слов документа.

Каждой *q*-й характеристике объекта $\bar{x}$ можно поставить в соответствие набор $\bar{y} = (y_1,...,y_e)$ – вектор расширения характеристики объекта предметной области, где *e* – количество расширений характеристики объекта $\bar{x}$.

В данном случае под расширением понимается синонимичный ряд для каждого слова документа. При этом вектор $\bar{y}$ может не иметь элементов, т.е. количество расширений характеристики объекта $\bar{x}$ $e = 0$, а для характеристики $x_q$ расширение определяется как $\bar{y}_q = (y_{q1},...,y_{qe})$.

$\bar{S} = (\bar{s}_{11},...,\bar{s}_{nn})$ – множество матриц схожести двух объектов $\bar{X}$, где $n$ – количество объектов $\bar{X}$. Каждая матрица схожести представляет набор соотношения всех лексем двух документов: $\bar{s}_{i,j} = (s_{11},...,s_{1dj},..,s_{di1},..,s_{didj})$ – вектор схожести объектов $\bar{x}_i$ и $\bar{x}_j$ размерности *di* и *dj* соответственно, а $s_{ij} \in \{0,1\}$.

В случае если $y_{jgm} \equiv x_{ih}$, где $g \in [1,dj]$, а $m \in [1,e]$, тогда $x_{ih} \in (y_{jg1},...,y_{jge})$, где $h \in [1,di]$, а следовательно, $s_{ih,jg} = 1$, то есть если расширенный ряд одной из характеристик соответствует сравниваемой характеристике, то это говорит о схожести данных характеристик и отражается в матрице схожести двух объектов $\bar{x}_i$ и $\bar{x}_j$. Применив формулу (2) к матрице $\bar{s}_{i,j}$, получим коэффициент Джаккарда:

$$sim(\bar{x}_i,\bar{x}_j) = \frac{|\bar{x}_i \cap \bar{x}_j|}{|\bar{x}_i \cup \bar{x}_j|} = \frac{\sum\{s_{ih,jg} \mid s_{ih,jg} = 1\}}{d - \sum\{s_{ih,jg} \mid s_{ih,jg} = 1 \wedge x_{ih} \in (y_{jg1},..,y_{jge})\}}.$$

**Метод вычисления схожести текстовых объектов на основе функции Джаккарда с предварительной обработкой. О**пределение схожести двух документов с предварительным удалением стоп-слов и обработкой синонимических замен включает в себя следующие этапы (см. рисунок 2).

1. Представление каждого из объектов предметной области $\bar{X}$ в виде вектора $\bar{x} = (x_1,...,x_d)$. Удаление из характеристик объекта тех, которые не принадлежат $[w_{\min};w_{\max}]$, т.е. множество характеристик объекта составляют $x_m \in \bar{x} \mid w_m \in [w_{\min};w_{\max}]$. Определение мощности объединения $\bar{x}_i$ и $\bar{x}_j$, $d = |\bar{x}_i \cup \bar{x}_j|$.

2. Построение матрицы схожести $\bar{s}_{i,j}$ двух объектов $\bar{x}_i$ и $\bar{x}_j$.

3. Предварительная обработка – присвоение $s_{ih,jg} = 1$, при $y_{jgm} \equiv x_{ih}$ и $x_{ih} \in (y_{jg1},...,y_{jge})$.

4. Вычисление коэффициента Джаккарда с учетом предварительной обработки, используя формулу (2):

$$sim(\bar{x}_i,\bar{x}_j) = \frac{|\bar{x}_i \cap \bar{x}_j|}{|\bar{x}_i \cup \bar{x}_j|} = \frac{\sum\{s_{ih,jg} \mid s_{ih,jg} = 1\}}{d - \sum\{s_{ih,jg} \mid s_{ih,jg} = 1 \wedge x_{ih} \in (y_{jg1},..,y_{jge})\}}.$$

5. Принятие решения о схожести объектов на основе коэффициента Джаккарда $sim(\bar{x}_i,\bar{x}_j)$.

На рисунке 2 представлена подробная схема принятие решения о схожести объектов на основе коэффициента Джаккарда.

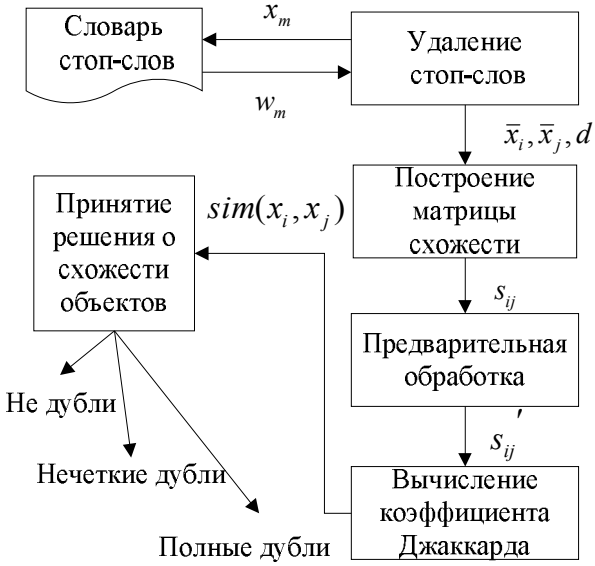



**Рисунок 2 - Схема определения схожести документов на основе коэффициента Джаккарда**

**Оценки эффективности методов поиска дубликатов.** Правильный выбор метрики, оценивающей эффективность работы систем в задачах информационного поиска, определяет полезность исследования. В данном случае был выбран стандартный набор метрик, оценивающих эффективность методов поиска дублей.

Точность (*precision*) характеризует способность системы выдавать в списке результатов только документы, действительно являющиеся дубликатами:

$$precision = \frac{a}{a+b},$$

где *a* – количество найденных пар дубликатов, совпадающих с «релевантными» парами; *b* – количество найденных пар дубликатов, не совпадающих с «релевантными» парами.

Полнота (*recall*) характеризует способность системы находить дубликаты, но не учитывает количество ошибочно определенных недублей:

$$recall = \frac{a}{a+c},$$

где *a* – количество найденных пар дубликатов,

совпадающих с «релевантными» парами; *c* – количество не найденных пар дубликатов, совпадающих с «релевантными» парами.

В данном исследовании оценивать точность нет необходимости, так как исходные данные уже кластеризованы с заданной точностью. Поэтому основной метрикой оценки исследования является полнота.

**Цель исследования.** В работе [4] было установлено, что алгоритм на основе схожести Джаккарда более предпочтителен в использовании, чем метод, основанный на дистанции с аффинными штрафами из-за меньших вычислительных затрат. В работе [5] было установлена эффективность использования предварительной обработки на предмет синонимических замен, что позволило увеличить полноту детектирования дублей. Однако при вычислении коэффициента Джаккарда с предварительной обработкой значительно увеличиваются временные затраты. В качестве варианта их сокращения было предложено сокращать количество обрабатываемых слов, устраняя стоп-слова [9]. При удалении стоп-слов также устраняются шумовые воздействия на меру схожести.

В данном исследовании стоит задача сравнить результат работы разработанной методики и алгоритма «шинглов». Этот алгоритм выбран из-за его популярности в программах поисковых систем [4]. Так как для алгоритма шинглов важен размер, необходимо установить наиболее подходящую для данного исследования n-словность.

Результаты сравнения работы метода «шинглов» и метода, основанного на сходстве Джаккарда, будут оценены с помощью метрики полноты. По итогам исследования необходимо выделить преимущества и недостатки разработанной модели, предложить пути усовершенствования алгоритма.

Таким образом, в данной работе можно выделить следующие цели:

– реализация алгоритма шинглов и определение наиболее подходящего для данного исследования размера шингла;

– сравнение эффективности алгоритма вычисления схожести документов без устранения синонимических замен на основе функции Джаккарда и алгоритма шинглов;

– сравнение эффективности алгоритма вычисления схожести документов с устранением синонимических замен на основе функции Джаккарда и алгоритма шинглов.

**Постановка задачи.** Согласно выделенным целям можно поставить следующие задачи.

1. Составить кластеры схожих документов с заданной точностью.
2. Организовать обработку кластеров алгоритмом шинглов и методом Джаккарда.
3. Обработав данные алгоритмом шинглов с разными размерами шингла, установить его наиболее подходящую длину.
4. Построить графики полноты для обоих методов и произвести их анализ.

**Экспериментальное исследование.** Для выполнения исследования была подобрана коллекция различных текстов более 100 вариантов. Размер текстов ограничивался 2000 знаков, чтобы устранить влияние данного параметра на ход эксперимента. Каждый текст обрабатывался несколькими способами: ручной рерайтинг статьи, когда человек перефразирует предложения текста, используя синонимические замены, входящие в собственный лексикон, и автоматический рерайтинг, когда изменения текста производят с помощью программы, меняющей в тексте слова на синонимы из словаря. Для автоматического рерайтинга использовалась сторонняя разработка Night ArticleGen и словарь синонимов на 14 тысяч слов, который в дальнейшем участвовал в расчете матрицы схожести с обработкой синонимических рядов. Таким образом, в обработке участвовало более 300 текстов и более 100 кластеров.

Для реализации устранения стоп-слов использовался файл с IDF значениями слов, отсортированных в порядке возрастания и критический отрезок значимых слов [500; 191703], полученный при анализе указанного выше файла IDF значений. Для получения этого файла была написана программа, на вход которой подавались тексты различного содержания для устранения тематического влияния общим объемом 500МБ. Для расчета IDF значений термов использовалась формула (3), которая из приведенных в [10] вариантов расчета IDF дала лучший результат в данном исследовании. Лемматизация (процесс привода словоформы к лемме — её нормальной словарной форме) слов была осуществлена с помощью парсера mystem, компании Яндекс.

$$\mathrm{ICF} = \frac{\mathrm{TotalLemms}}{\mathrm{CF}}, \qquad (3)$$

где *TotalLemms* – общее число вхождений всех лемм в коллекции, *CF* – число вхождений леммы в коллекцию.

Для расчета схожести текстов на основе алгоритма шинглов использовался скрипт [11]. Данный скрипт разбивает тексты на шинглы, а затем сравнивает полученные матрицы. Резуль-

тат выдается для шинглов из 1 слова, 2, 3 и 4. Что дает возможность отследить необходимые тенденции для выбора длины шингла. Представленный скрипт был доработан на предмет устранения стоп-слов и лемматизацию. Принципы указанных вставок описаны выше. Также данный скрипт доработан для пакетной обработки данных, позволяющей получить матрицу схожести по всем кластерам. Для подсчета временных затрат была внедрена соответствующая функция.

Для расчета коэффициента схожести Джаккарда была написана программа, которая сравнивает поступающие документы между собой, строя на выходе матрицу схожести. Программа работает в двух режимах с учетом синонимических рядов и без их учета. Для сокращения времени обработки и устранения шумового влияния на меру схожести в составлении образа документа участвуют только значимые слова, которые лежат в указанном выше отрезке значимых слов: [500; 191703].

Для лемматизации слов каждого документа, т.е. для решения задачи представления каждого из объектов предметной области $\overline{X}$ в виде вектора, был применен парсер mystem, предоставляемый компанией Яндекс. Программа mystem производит морфологический анализ текста на русском языке, представляя все слова в нормальной форме. Для слов, отсутствующих в словаре, порождаются гипотезы. Таким образом, вектор объекта $\overline{x}$ – набор слов в нормальном виде рассматриваемого документа.

Для определения синонимов использовался словарь на 14 тыс. слов. Он же участвовал в автоматическом рерайтинге начальных данных.

Для оценки временных затрат была реализована функция, которая рассчитывала время на получение матриц схожести в двух режимах.

Таким образом, для реализованной программы расчета схожести строк на основе коэффициента Джаккарда, входными данными являются: обрабатываемые документы, словарь обратных частот и словарь синонимов. Выходными данными являются 4 файла: общая матрица схожести Джаккарда всех документов между собой без обработки синонимических замен, общая матрица схожести Джаккарда всех документов между собой с обработкой синонимических рядов и 2 файла, соответствующих временных затрат на подсчет указанных выше матриц.

С помощью матриц рассчитывались показатели полноты для всех групп и для всех алгоритмов. При этом по условию количество нечетких дублей было известно заранее. В результате был построен график, оценивающий метрику полноты для разных длин шинглов при расчете схожести по данному алгоритму, и совмещенный график полноты для метода Джаккарда, для метода Джаккарда с учетом синонимических замен и для алгоритма шинглов, с выбранным оптимальным параметром размера шингла. Данные графики анализировались, по результатам анализа делались соответствующие выводы о предпочтительности использования разработанного метода в сравнении с методом шинглов.

**Результаты исследований.** Полученные кластеры, путем обработки текстов ручным и автоматическим рерайтингом, были обработаны алгоритмом шинглов и методом Джаккарда. Часть результатов можно видеть в таблице 1.

**Таблица 1. Полученная мера схожести**

| Метод | | Мера схожести | | |
|---|---|---|---|---|
| | | 1 с 2 | 1 с 3 | 2 с 3 |
| Первый кластер | | | | |
| Джаккард без учета синонимов | | 57.50% | 32.84% | 26.89% |
| Джаккард с учетом синонимов | | 62.20% | 53.22% | 40.41% |
| метод шинглов | 1 слово в шингле | 53.61% | 34.01% | 23.92% |
| | 2 слова в шингле | 36.49% | 12.27% | 7.25% |
| | 3 слова в шингле | 25.76% | 6.34% | 2.42% |
| | 4 слова в шингле | 19.17% | 3.09% | 1.03% |
| Второй кластер | | | | |
| Джаккард без учета синонимов | | 86.17% | 43.87% | 40.43% |
| Джаккард с учетом синонимов | | 90.36% | 79.85% | 77.44% |
| метод шинглов | 1 слово в шингле | 85.37% | 41.69% | 38.84% |
| | 2 слова в шингле | 72.78% | 21.29% | 19.87% |
| | 3 слова в шингле | 65.29% | 12.38% | 11.44% |
| | 4 слова в шингле | 58.19% | 7.88% | 7.44% |
| … | | | | |

Из таблицы 1 видно, что мера схожести первого со вторым текста одного кластера в 1-5,2 раза больше, чем схожесть с третьим и третьего со вторым. Под номером 2 текст, образованный автоматическим рерайтингом на базе словаря синонимов. Обычный процент замен синонимайзера (программа, размножающая

статьи при помощи синонимичных рядов) составляет от 20 до 40%, что объясняет такой высокий процент схожести.

Также согласно таблице 1, чем больше число слов в шингле, тем меньше мера схожести. Затухание меры с увеличением размера шингла составляет 7-10 позиций. Более сильное затухание прослеживается по горизонтали (зависимость от процента изменений текста). Мера схожести при сравнении первых двух текстов кластера 2-2,5 раза больше, чем при сравнении текстов ручного рерайтинга.

Рисунок 3, отражающий метрику полноты для каждого размера шингла, подтверждает описанные выше заключения.

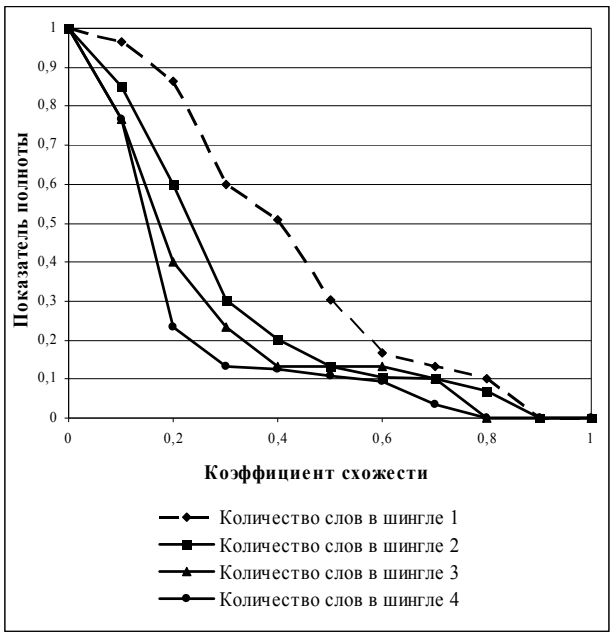



**Рисунок 3 - Алгоритм шинглов. График полноты для различных длин шингла**

Большинство поисковых систем используют более расширенные алгоритмы, основанные на супершинглах и мегашинглах. Согласно [2] показатель полноты улучшается на 30%. Поэтому полученные низкие показатели полноты можно объяснить использованием простой реализации алгоритма шинглов.

Однако и при данной реализации несомненно преимущество метода Джаккарда с учетом синонимов в определении нечетких дублей, созданных путем синонимических замен. Если показатель сходства метода Джаккарда с учетом синонимов увеличивается в 1,5-2 раза, то по сравнению с алгоритмом шинглов в 3-5 раз. За основу сравнения взяты коэффициенты, образованные при размере шингла в два слова, так как при используемом алгоритме шинглов в одно слово, данный алгоритм сводится к алгоритму Джаккарда. Этот факт и объясняет значительную схожесть полученных мер при расчете коэффициента Джаккарда и схожести по одинарному шинглу (см. таблицу 1 и рисунок 4).

На рисунке 4 представлены графики полноты двух методов. График полноты метода Джаккарда с учетом синонимических замен наиболее гладкий и показывает самый большой интервал полноты. Рисунок 4 подтверждает сделанные ранее выводы: об эффективности использования разработанного метода Джаккарда с учетом синонимов, о сходстве метода Джаккарда и алгоритма шинглов с размером в одно слово.

Таким образом, преимущество разработанного метода в сравнении с алгоритмом шинглов по метрике полноты очевидно.

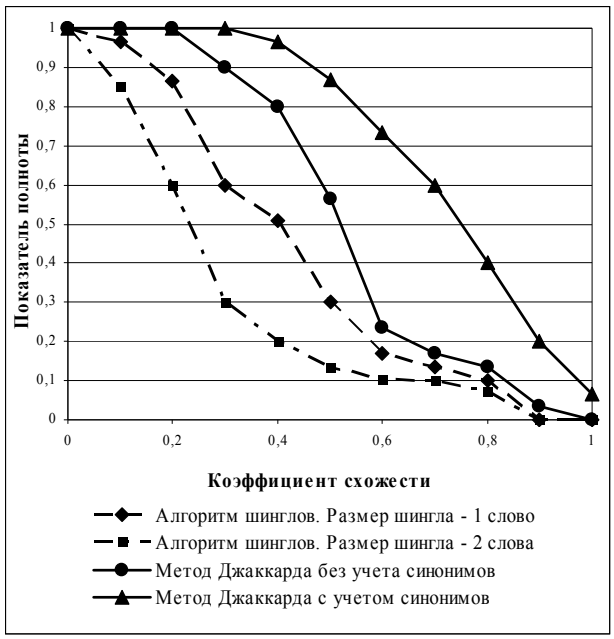



**Рисунок 4 - Соотношение графиков полноты**

Что касается временных затрат, то преимущество у метода Джаккарда, чуть дольше рассчитывается сходство с использованием алгоритма шинглов и совсем много времени уходит на учет синонимов. Ниже в таблице 2 приведены данные по временным затратам.

**Таблица 2. Время расчета схожести**

| Количество документов | Время обработки, *мс* | | |
|---|---|---|---|
| | Шинглы | Джаккард | Учет синонимов |
| 4 шт. по 2000 знаков | 12031 | 5042 | 2750469 |
| 4 шт. по 4000 знаков | 23964 | 11403 | 7715109 |

Временные показатели при расчете матрицы Джаккарда с учетом синонимов критически велики для использования этого метода в промышленных масштабах off-line фильтрации дублей. Однако данный способ будет полезен при

фильтрации на лету, когда объем выборки значительно ниже, чем вся поисковая база, а также можно ограничиться расчетами 30-50 результатов выдачи (согласно статистике [12], пользователь просматривает лишь первые страницы ответа поисковой системы).

Для сокращения временных ресурсов можно предложить усовершенствовать метод поиска синонимов по базе, распараллелить процессы расчета коэффициента Джаккарда, а также доработать алгоритм с учетом частоты слов в наборе.

**Заключение.** В статье описана разработанная модель и метод сравнения двух текстов на основе коэффициента Джаккарда с учетом синонимических замен. Данный метод был рассмотрен в сравнении с широко используемым алгоритмом шинглов. Исследование показало, что по метрике полноты метод Джаккарда с учетом синонимов дает хороший результат, выше результата метода шинглов в 3-5 раз. Однако использование разработки в промышленных масштабах off-line фильтрации web-дублей поисковыми системами затрудняется большими временными ресурсами.

За счет использования в разработке системы удаления стоп-слов удалось сократить время построения матрицы в 2 раза. Также предложено использовать метод Джаккарда с учетом синонимических замен в методах on-line фильтрации. Во-первых, это намного сужает множество обрабатываемых документов, во-вторых, так как посетители просматривают всего лишь первые 3-5 страниц поисковой выдачи, можно ограничиться обработкой документов этого подмножества.

Преимущества разработанного метода перед традиционным методом расчета коэффициента Джаккарда и алгоритмом шинглов наиболее заметны на паре документов, один из которых создан посредством автоматического рерайтинга (наиболее частый способ создания контента для обмана поисковых систем). Полнота метода Джаккарда с учетом синонимов выше примерно на 30%, чем у метода шинглов. В данной работе использовалась самая простая реализация метода шинглов, что дало не совсем четкую картину и увеличение полноты составило 50-60 %.

Дальнейшее усовершенствование метода будет затрагивать улучшение метода поиска синонимов по базе и распараллеливание процесса расчета меры схожести. Также доработка коснется использования частоты слов непосредственно в наборе документа, так как не всегда удаленное стоп-слово из общего словаря является нейтральным для конкретного документа. Это позволит более качественно построить образ документа, а следовательно получить более эффективные результаты при детектировании нечетких дублей.

### *Библиографический список*

1. *Неелова Н.В.* Методы кластеризации веб-дубликатов как инструмент повышения качества ответа поисковых систем // Инновационные подходы к применению информационных технологий в профессиональной деятельности: сб. науч. тр. – Белгород, 2009. – С. 305-309.

2. *Зеленков Ю.Г., Сегалович И.В.* Сравнительный анализ методов определения нечетких дубликатов для WEB-документов // Труды 9-ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» RCDL’2007: сб. работ участников конкурса –Переславль-Залесский, 2007. – Т. 1. – С. 166-174.

3. Некоторые автоматические методы детектирования спама, доступные большим почтовым системам [Электронный ресурс] // http://company.yandex.ru/public/articles/antispam.xml

4. *Цыганов Н.Л., Циканин М.А.* Исследование методов поиска дубликатов веб-документов с учетом запроса пользователя // Интернет-математика 2007: сб. работ участников конкурса – Екатеринбург: Изд-во Урал. ун-та, 2007. – С. 211-222.

5. *Неелова Н.В.* Исследование лексического метода вычисления схожести строк веб-документа с учетом предварительной обработки // XXXV Гагаринские чтения: Научные труды Международной молодежной научной конференции в 8 томах / М.: МАТИ, 2009. – Т. 4. – С. 31-32.

6. *Andrei Z. Broder.* On the resemblance and containment of documents. In Proceedings of Compression and Complexity of Sequences 1997, pages 21–29. IEEE Computer Society, 1997.

7. *Кузнецов С.О.* Порождение кластеров документов-дубликатов: подход, основанный на поиске частых замкнутых множеств признаков / С.О. Кузнецов, Д. И. Игнатов, С. А. Объедков, М. В. Самохин // Интернет-математика 2005. Автоматическая обработка веб-данных. – М., 2005. – С. 302-319.

8. *Кондратьев М.Е.* Анализ методов кластеризации новостного потока [Электронный ресурс] // Материал конференции RCDL’2006.

9. *Неелова Н.В.* Функция удаления нейтральных слов при вычислении нечетких дублей лексическим методом Джаккарда // Интеллектуальные и информационные системы: материалы Всероссийской научно-технической конференции / Тульский государственный университет. – Тула, 2009. – С. 149-151.

10. *Гулин А., Маслов М., Сегалович И.* Алгоритм текстового ранжирования Яндекса на РОМИП2006 // Труды РОМИП 2006. – Суздаль, 2006.

11. PHP код алгоритм шинглов [Электронный ресурс] // http://utext.rikuz.com/

12. *Ашманов И.* Оптимизация и продвижение сайтов в поисковых системах / И. Ашманов, А. Иванов – СПб.: Питер, 2008. – 400 с. : ил. – ISBN: 978-5-388-00008-8.

УДК 330.45 (075.8)

*А.А. Дунаев, В.Е. Лихачев*

# ОПТИМИЗАЦИЯ МАРШРУТОВ ОБСЛУЖИВАНИЯ ЗАЯВОК НА БАЗЕ АЛГОРИТМА БРУДНО - ЛУРЬЕ

*Разработана экономико-математическая модель оптимального планирования маршрутов движения обслуживающих элементов систем.*

***Ключевые слова:*** *транспортная задача, алгоритм, минимизация, обслуживание, заявка, поставщик, потребитель.*

**Введение.** Принятие решений – неотъемлемая часть повседневной деятельности человека. Как владелец тех или иных ресурсов производства, человек принимает решения о наиболее рациональном их применении. Всегда речь идёт о некотором обусловленном выборе. Концепция принятия решения в качестве первичного элемента деятельности рассматривает принятое решение как обоснованный сознательный выбор одной из ряда альтернатив. В качестве последних могут выступать стратегии, планы, варианты и т.д. Классическая транспортная задача как разновидность задачи линейного программирования ставится для однородного продукта. В реальной жизни представляет интерес решение транспортной задачи для неоднородных продуктов.

**Цель работы** исследовать алгоритм управления процессом обслуживания заявок потребителей услуг и разработать модифицированный алгоритм на основе решения транспортной задачи для минимизации числа рабочих циклов в системах массового обслуживания.

**Ключевая концепция.** Классическая транспортная задача явилась результатом стремления минимизировать затраты на поставки товаров от нескольких поставщиков к нескольким потребителям. Задача в общем виде формулируется следующим образом [1]:

1) имеется $M$ - поставщиков однородного $A_i$ товара $a_i$, расположенного в различных пунктах, где $i$ – номер поставщика, $i = \overline{1, M}$ ;

2) имеется $N$ потребителей $B_j$ товара $b_j$, также расположенных в различных пунктах, где $j$ – номер потребителя, $j = \overline{1, N}$ ;

3) стоимость перевозки одной единицы товара из пункта $A_i$ в пункт $B_j$ равна $C_{ij}$ денежных единиц.

Транспортная задача вычисления наилучших маршрутов доставки грузов рассматривается как задача линейного программирования с заданной стоимостной матрицей $\left\| C_{ij} \right\|, i = 1,2,...,M;\ \ j = 1,2,...,N$, где $C_{ij}$ - цена доставки груза от поставщика с номером $i$ к потребителю с номером $j$ [1]. Для объёмов поставки требуется выполнение условия баланса

$$\sum_{i=1}^{M} A_i = \sum_{j=1}^{N} B_j \ . \qquad (1)$$

Требуется вычислить такое множество маршрутов $\{I_k, J_k\}_{k=1}^{N^P}$ с объемами поставки $x_k$,

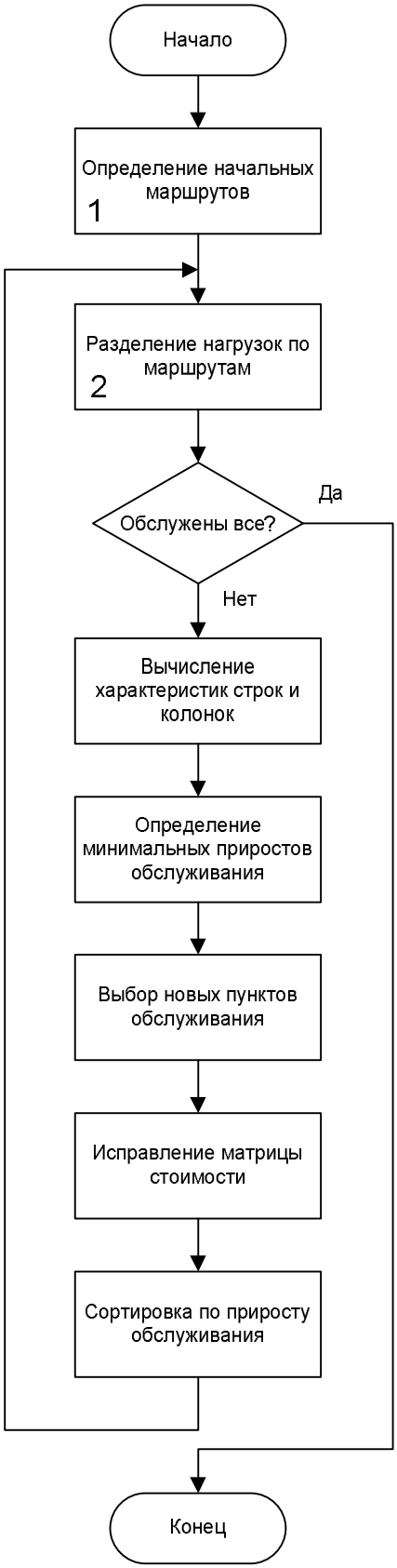
где $p$ – количество элементов, чтобы достигался минимум целевой функции

$$\Phi = \sum_{k=1}^{N^P} C_{I_k J_k} x_k \ . \qquad (2)$$

Представляет интерес разработка алгоритма решения транспортной задачи, минимизирующего число рабочих циклов поставщика услуг. Под рабочим циклом понимается непрерывный процесс обслуживания потребителей одним поставщиком услуг. Одной из таких задач является составление графика обслуживания заявок в системах массового обслуживания с целевой функцией числа свободных дней поставщиков услуг, максимум которой необходимо вычислить. Имеется в виду, что работа поставщиков услуг по выполнению заявок на обслуживание может выполняться как с выездом на место обслуживания, так и без выезда, т.е. стационарно, на рабочем месте. Обслуживающие элементы рассматриваются как поставщики услуг, объекты выполнения работ - потребители услуг.

**Постановка задачи.** Одним из методов

решения транспортной задачи является алгоритм Брудно - Лурье [2]. Для решения поставленной задачи предлагается использовать модификацию данного алгоритма. Блок-схема алгоритма представлена на рисунке.




**Блок-схема алгоритма Брудно - Лурье**

Модернизации подлежат первые два блока алгоритма, обозначенные цифрами 1 и 2.

Рассмотрим задачу выполнения поставок, когда поставщик представлен списками единичных услуг

$$\{A_{ipl}\}, i=\overline{1,M}; p=\overline{1,n_i^G}; l=\overline{1,N_{ip}^G}, A_{ipl}\in[1,N] \quad (3)$$

.

Для задачи минимизации числа рабочих дней поставщиков услуг $\{A_{ipl}\}$ представляет интерес список номеров объектов работ, связанных с исполнителями работ $i$-м исполнителем, $n_i^G$ - длина списка работ, $N_{ip}^G$ - количество заявителей в заявке, если $N_{ip}^G>1$. Для кодирования части однотипной заявки, например, при целесообразности бригадного обслуживания, можно использовать часть числового диапазона этого параметра. Полагаем, что число операций работы при этом соответствует единице, например, одному виду. Если $N_{ip}^G<1$, то для групповой заявки выбрана эта работа. Объём загруженности заказчика услуг представлен множеством

$$\{B_{dmj}\}, d=\overline{1,n^d};\ m=\overline{1,N^G};\ j=\overline{1,M},$$

$n^d$ - число дней работы на заказе; $N^G$ - число заказчиков услуг. Суммарную занятость на заказе с учетом параллельной занятости описывает множество $\{S_{dm}\}$, индексы аналогичны предыдущему множеству. Кроме этого, необходимо учитывать количество заказов для определения числа переездов. Бригады будут учитываться множеством $\{P_{jr}\}$, $j=\overline{1,N}$, $r$ – номер бригады. Занятость поставщика услуг учитывается в $\{Z_{id}\}$. Смысл индексов в каждой величине не меняется. На объём работы поставщика услуг наложено ограничение

$$S_{dm}\le S^{\max},\ Z_{id}\le Z^{\max}. \quad (4)$$

Матрица стоимости обслуживания в задаче подлежит определению, поскольку, вообще говоря, в рабочие дни загруженность поставщиков услуг является примерно постоянной.

Ставится задача вычисления такого множества $\{I_k, J_k\}$, что при полном обслуживании заказчика услуг и выполнении условий (4) достигался максимум целевой функции

$$\Phi=\sum_{i=1}^{M}\sum_{d=1}^{n^d} f_{id}, \quad (5)$$

где $f_{id} = \begin{cases} 0, & если\ Z_{id} \neq 0 \\ 1, & если\ Z_{id} = 0 \end{cases}$,

что, очевидно, соответствует задаче минимизации суммарного количества рабочих дней с выездом специалистов.

**Решение задачи.** Для решения поставленной задачи введём изменяющуюся в процессе итерационных шагов алгоритма стоимостную матрицу $\|C_{ij}\|$ таким образом, что при выборе обслуживания поставщиком услуг (бригадой) с номером *i* потребителей в день *d* соответствующие стоимости уменьшаются на заданную величину. Начальные значения элементов стоимостной матрицы выбираются постоянными, поскольку при отсутствии дополнительных ограничений порядок обслуживания потребителей поставщиком *i* не имеет значения, если $Z_{id} = 0$ для всех *d*. Общее число потребителей за $n^d$ дней $N = n^d \cdot N^G$, поэтому общий индекс потребителя можно определить как $j = N^G(d-1) + m$. Элементы стоимостной матрицы выбираются целочисленными, и динамическое уменьшение стоимостей осуществляется на единицу.

Таким образом, решение поставленной задачи как транспортной усложняется.

Рассмотрим схему алгоритма Брудно - Лурье решения транспортной задачи, представленную на рисунке. Если маршруты $\{I_k, J_k\}$ заданы, то возможно распределение нагрузок по маршрутам. Рассмотрим алгоритм распределения нагрузок для поставленной задачи. Для маршрута с номером *k* остаточная потребность определится как $B^o = S^{\max} - B_{[I_k J_k]}$, где мультииндекс $[I_k, J_k] = dmJ_k$. Выполнение потребности $J_k$ потребителя поставщиком $I_k$ осуществляется списком $n_i^G$ обслуживания, если выполняется условие

$$N_{ip}^G > 0 \wedge B^o > 0 \wedge S_{dm} < S^{\max} \qquad (6)$$

и группа списка $A_{ipl} = m$ для одного из номеров *l*.

В этом случае добавляется единица нагрузки поставщика услуг и заказчика в элементах $S_{dm}$, $B_{[I_k J_k]}$ и выключается выбранный элемент списка обслуживания изменением знака $N_{ip}^G$. Остаточная потребность $B^o$ при этом также уменьшается на единицу.

При одновременном обслуживании нескольких заказчиков услуг (случай $N_{ip}^G > 1$) элемент списка поставщика услуг включается в обслуживание, если одновременно для группового заказчика остаточная потребность $B_q^o > 0,\ q = N^G(d-1) + A_{I_k pl}$, иначе этот элемент списка в обслуживание текущего маршрута не включается.

Если рассматриваемый элемент списка работ поставщика услуг (управляющей компании) обслуживает часть потребителей – небольшую единичную группу заказчиков, то соответствующий элемент $S_{dm}$ увеличивается, если увеличивается максимум $P_{J_k r}$, иначе принимается условие одновременной занятости группового заказчика. В последовательном алгоритме вычисления заявок на обслуживание несложно учесть возможность разной загруженности бригад поставщика услуг и групп по дням. Для этого в логической строке (6) достаточно задать соответствующие ограничения $S^{\max} = S_{dm}^{\max}$, $Z^{\max} = Z_{id}^{\max}$. Для регулирования числа различных типов заявок на каждый день достаточно ввести счётчики типов заявок и добавить логику условий в (6). Если $J_k = N^G(d-1) + m$, то стоимости уменьшаются на единицу для всех $N^G$. Такой алгоритм осуществляет динамическое построение стоимостной матрицы для решения поставленной оптимизационной задачи.

Таким образом, входной информацией блока вычисления поставок - выполнения плана обслуживания потребителей услуг является описание загрузки исполнителя и $N^P$ маршрутов движения. Выходной информацией блока является множество $\{B_j^o\}$, которое далее используется для вычисления характеристик столбцов и строк алгоритма Брудно – Лурье, поэтому модификации подлежат только блоки, помеченные цифрами 1 и 2 на блок-схеме алгоритма. Поскольку при вычислении характеристик $B_j^o$ фактически являются логическими элементами, то величина $B_j^o$ значения не имеет.

**Дополнительные ограничения** Оптимальный по критерию минимизации числа рабочих циклов график движения (распределения работ)

должен учитывать наличие общего числа заявок по участку и его специализацию. Для этого необходимо учесть дополнительные ограничения при решении поставленной задачи. Решение задачи учёта наличия специализированных заявок и удаленных маршрутов можно осуществить одновременно с решением задачи оптимизации целевой функции (5). Для этого необходимо ввести множество $\{s_v^P\}$, $v = \overline{1, n^a}$, где $v$ - количество типов, $n^a$ - номера маршрутов, $s_v^P$ - количество заявок $v$-типа. Для каждой заявки должен быть однозначно определен характер проводимых работ, что может быть задано с помощью множества $\{t_{ipl}\}$, $t_{ipl} \in [1, n^a]$ в дополнение к элементам $A_{ipl}$. После этого учёт ограничения наличия свободных бригад можно учесть дополнительным к (6) ограничением

$$T_{vd} \le s_v^d, \qquad (7)$$

где элементы множества $\{T_{vd}\}$ каждый раз увеличиваются на единицу одновременно с элементами $B_{dmJ_k}$ для номеров $T_{vd}$, $v = t_{I_k pl}$. Кроме ограничения (7) на отдельные типы заявок могут быть наложены дополнительные ограничения.

Алгоритм решения транспортной задачи допускает расчёт поэтапных объёмов заявок обслуживания потребителей. Для этого достаточно в качестве начальных маршрутов взять вычисленные маршруты предыдущего этапа и решить задачу с увеличенными нагрузками. На практике принято проводить решение поставленной задачи в начале недели для $n^d = 5$ и затем объёмы $\{A_{ipl}^*\}$ распределяются для перспективного планирования для $n^d = 10$. Период цикла планирования, как правило, составляет пять дней, хотя, при необходимости, проводят и суточный расчёт или корректировку текущего цикла. Объединенные множества маршрутов $\{I_k, J_k\}$ первых двух решений принимаются за начальные маршруты следующего этапа решения задачи.

Для проверки эффективности разработанного алгоритма проведём решение задачи для следующих параметров (данные задаются в табличном виде в формате xls).

Входные данные приведены в таблице 1.

**Таблица 1 – Входные данные**

| | M | N | | |
|---|---|---|---|---|
| | 2 | 4 | | |
| $A_i$ | 8 | 6 | | |
| $B_j$ | 2 | 5 | 3 | 2 |
| $C_{1J}$ | 10 | 2 | 5 | 6 |
| $C_{2J}$ | 2 | 5 | 10 | 1 |
| M, N | 1 | 2 | 3 | 4 |

В таблице: M- количество поставщиков, N-количество потребителей, $C_{ij}$ – стоимостная матрица, $A_i$ и $B_j$ - товары поставщика и потребителя, соответственно, расположенные в различных пунктах.

Результат (T+M-1 строк) расчёта приведён в таблице 2.

**Таблица 2 – Результат расчёта**

| Откуда | Куда | Сколько |
|---|---|---|
| 2 | 1 | 2 |
| 1 | 3 | 3 |
| 1 | 4 | 2 |
| 1 | 2 | 3 |
| 2 | 2 | 2 |

**Заключение.** На основе вышеизложенного можно сформулировать основные концепции решения поставленной задачи.

1. Для решения поставленной задачи минимизации числа поездок и оптимизации маршрутов для неоднородных продуктов и безусловного выполнения всех заявок применим алгоритм решения транспортной задачи.

2. Описанные выше алгоритмы являются основой составления графика обслуживания потребителей услуг. Известен не только состав работающих бригад на каждый день, но и виды заявок, выполняемых каждой отдельной бригадой с учётом их специализации, для которых гарантируется обеспеченность необходимыми материалами, поэтому решение окончательной задачи составления графика обслуживания потребителей существенно упрощается.

3. На базе разработанного модифицированного алгоритма Брудно-Лурье на основе решения транспортной задачи для минимизации числа рабочих циклов в системах массового обслуживания в среде Borland Delphi разработана и опробована программа «Модифицированная транспортная задача».

### *Библиографический список*

1. *Орехов Н.А., Лёвин А.Г., Горбунов Е.А.* Математические методы и модели в экономике: учеб. пособие для вузов / Под ред. проф. Н.А.Орехова - М.: ЮНИТИ-ДАНА, 2004. – 302 с. - (Серия «Профессиональный учебник: Экономика»).

2. *Cormen T. H., Leiserson C. E., and Rivest R. L.* Introduction to Algorithms. MIT Press / McGraw-Hill, 1990. – 146 с.