DOI 10.17587/issn.1684-6400

ISSN 1684-6400

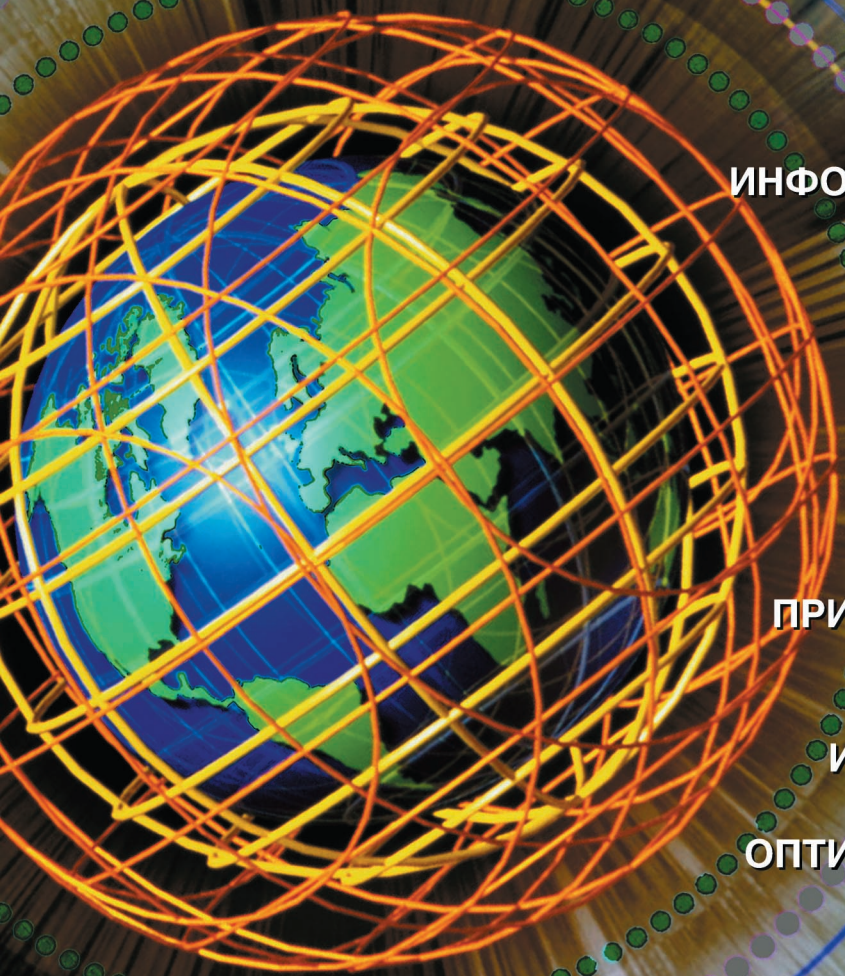
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Том 27

2021

№ 5

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ



САПР

КОМПЬЮТЕРНАЯ ГРАФИКА

МЕТОДЫ ПРОГРАММИРОВАНИЯ

ОПЕРАЦИОННЫЕ СИСТЕМЫ И СРЕДЫ

ТЕЛЕКОММУНИКАЦИИ И ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

НЕЙРОСЕТИ И НЕЙРОКОМПЬЮТЕРЫ

СТРУКТУРНЫЙ СИНТЕЗ

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

ОПТИМИЗАЦИЯ И МОДЕЛИРОВАНИЕ

ИТ В ОБРАЗОВАНИИ

ГИС

Рисунки к статье С. М. Салибекяна

«ОБЪЕКТНО-АТРИБУТНЫЙ ПОДХОД ДЛЯ СЕМАНТИЧЕСКОГО АНАЛИЗА ЕСТЕСТВЕННОГО ЯЗЫКА»

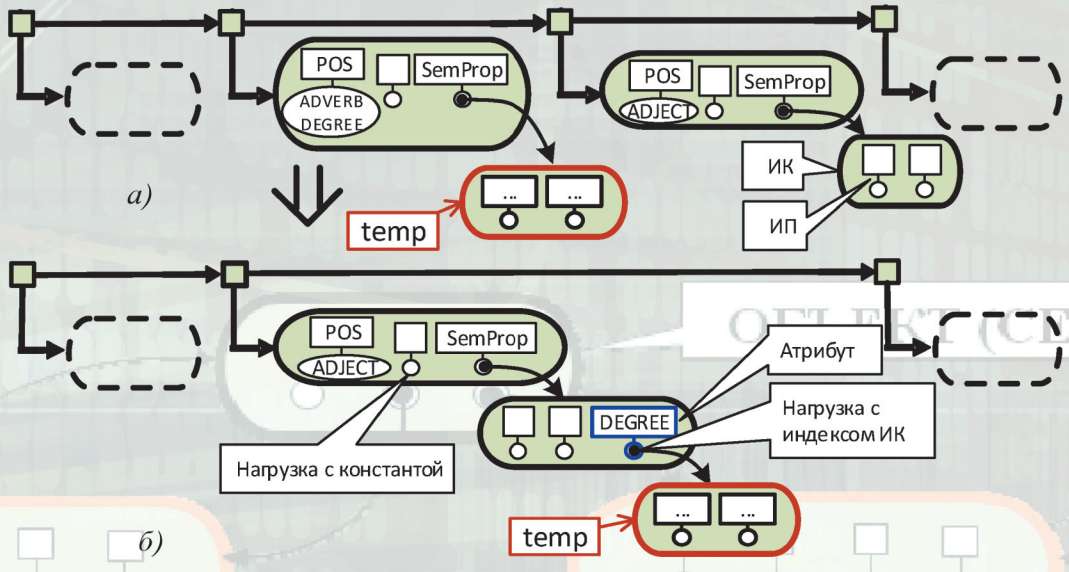


Рис. 1. Преобразование списка толкований слов

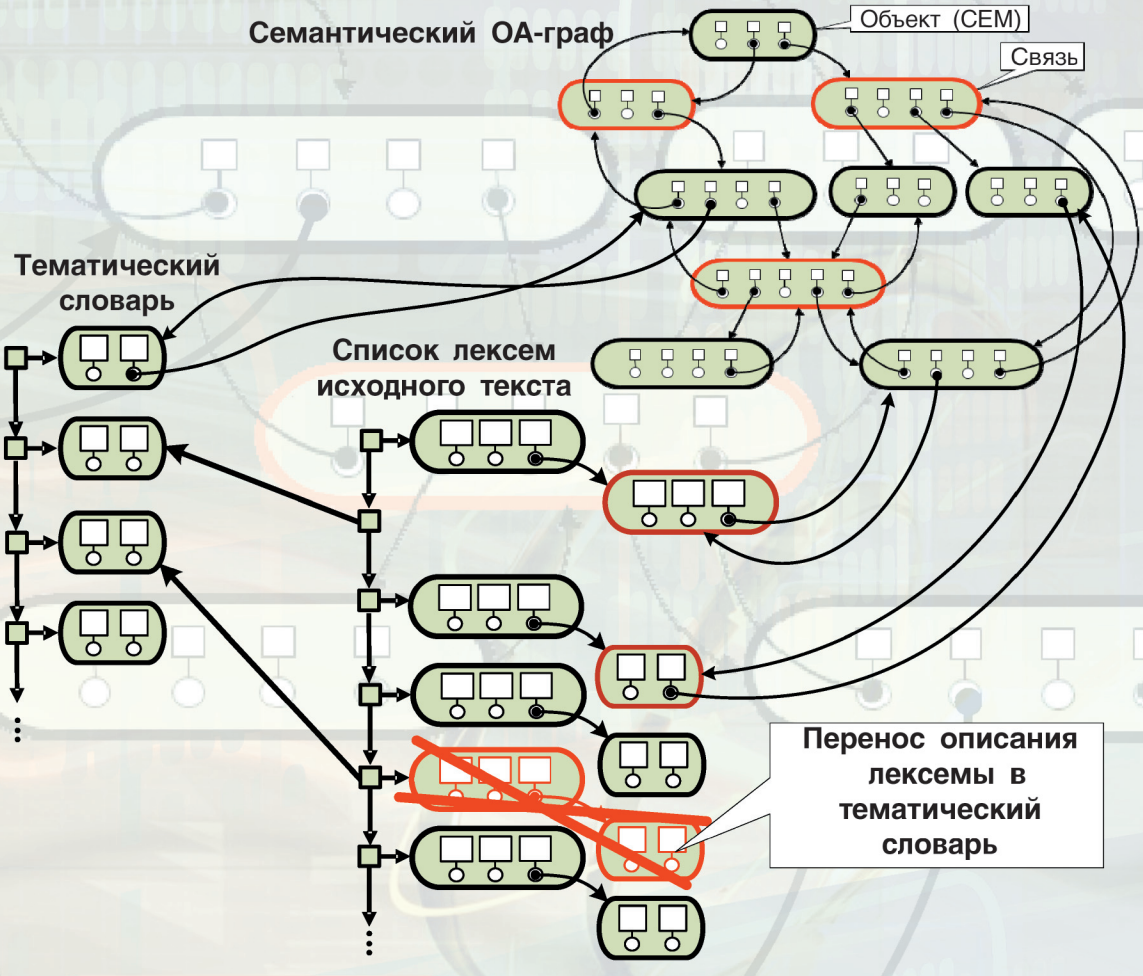


Рис. 2. Схема построения ОА-графа из списка лексем исходного текста

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Том 27
2021
№ 5

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г. DOI 10.17587/issn.1684-6400

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу **http://novtex.ru/IT.**
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

INFORMATION TECHNOLOGIES
INFORMACIONNYE TEHNOLOGII

Vol. 27
2021
No. 5

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

Published since November 1995

DOI 10.17587/issn.1684-6400

ISSN 1684-6400

CONTENTS

Complete Internet version of the journal at site: **http://novtex.ru/IT.**
According to the decision of the Higher Certifying Commission of the Ministry of Education of Russian Federation, the journal is inscribed in "The List of the Leading Scientific Journals and Editions wherein Main Scientific Results of Theses for Doctor's or Candidate's Degrees Should Be Published"

УДК 004.023, 004.925 DOI: 10.17587/it.27.227-234

А. Д. Обухов, канд. техн. наук, e-mail: Obuhov.art@gmail.com,
А. Е. Архипов, аспирант, e-mail: alexeiarh@gmail.com,
А. О. Сидорчук, бакалавр, e-mail: sidorchuk.a.o@yandex.ru,
Тамбовский государственный технический университет

Математическое моделирование и визуализация процессов горения в тренажерных комплексах*

Рассматривается задача моделирования и визуализации физических процессов горения в тренажерных комплексах. Предлагается математическая модель, позволяющая формализовать структуру виртуальных объектов горения, их свойства и процесс смены состояний от возгорания объекта до полного уничтожения или тушения. Разработаны диаграммы жизненных циклов и правила смены состояний объектов виртуальной реальности для организации процесса визуализации горения и тушения в виртуальной реальности.

Ключевые слова: *компьютерная графика, 3D-моделирование, визуализация физических процессов, адаптивные тренажерные комплексы, формализация процесса горения, модель состояний виртуальных объектов, теория множеств, подготовка персонала*

Введение

При организации подготовки персонала промышленных предприятий к деятельности в штатных и аварийных ситуациях одним из ключевых моментов обучения является отработка действий в случае пожара, возгорания, задымления. Реализовать в полной мере процесс пожаротушения, эвакуации, задымления на производстве затруднительно, так как часть операций невозможно осуществить на работающем предприятии, часть связана с риском для здоровья. Поэтому применение тренажерных комплексов на базе виртуальной реальности экономически оправдано в данной ситуации.

Однако решение задачи моделирования реалистичных пожаров, распространения огня и эффектов задымления связано с неизбежным ростом вычислительной нагрузки на оборудование. В случае виртуальной реальности это осложняется исходными высокими требованиями к вычислительной мощности для отрисовки трехмерной графики. Реалистичность и точность моделирования процесса горения очень важна, так как позволяет выработать правильные практические навыки у обучающихся, однако полностью моделировать физические процессы распространения пламени и сгорания материала, влияние воздушных потоков, герметичности и влажности бессмысленно: во-первых, это нецелесообразно относительно итогового результата, во-вторых, приведет к значительной потере производительности, в-третьих, значительно повысит стоимость тренажерного комплекса.

Поэтому актуальной задачей при разработке адаптивных тренажерных комплексов (АТК) является достоверная и качественная визуализация физических процессов, особенно процесс распространения пожара. Помимо непосредственно визуализации корректное компьютерное моделирование позволяет решать сопутствующие задачи, среди которых следует отметить определение времени эвакуации при пожарах и негативного воздействия на человека [1, 2], вопросы эффективной борьбы с лесными пожарами [3].

Однако моделировать процессы горения без соответствующего математического обеспечения с полной достоверностью невозможно. В статье [8], например, представлен метод моделирования и визуализации распространения дыма и пожаров, основанный на первоначальном решении уравнений течения газа и жидкости в двухмерной постановке. В статье [9] рассмотрены методы использования эмпирических данных о по-

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-013-00567 с использованием вычислительного оборудования ЦКП "Цифровое машиностроение".

жарах и результатах моделирования горения на основе уравнений газодинамики в тренажерах виртуальной реальности. В работе [10] подробно рассмотрены вопросы моделирования горения. Предложенный в данной статье метод основан на физическом моделировании очагов ламинарного и турбулентного горения. Полученные результаты позволяют моделировать распространение пожара от испаряемых горючих жидкостей и твердых горючих веществ.

Общим недостатком, присутствующим в большинстве работ, является необходимость решения дифференциальных уравнений математической модели горения или анализ большего объема эмпирических данных. Поэтому использование указанных методов при непосредственно программной реализации распространения пожаров в виртуальной реальности приводит к значительным затратам времени на расчеты и решение дифференциальных уравнений и, следовательно, к повышению стоимости тренажерного комплекса и снижению его производительности. Поэтому актуальной задачей является разработка математической модели, адаптированной для использования в средствах визуализации и обеспечивающей корректность расчетов процессов горения в АТК. Естественно, что подобная упрощенная модель будет содержать ряд допущений, однако в рамках поставленной задачи требуется лишь обеспечить визуальное соответствие моделируемых процессов горения реальным.

1. Постановка задачи исследования

Необходимо осуществить формализацию, моделирование и практическую реализацию процессов горения в виртуальной среде. К физическим процессам относятся: горение, тушение, задымление, изменение размеров и текстуры виртуальных объектов. Разработанный математический аппарат должен с достаточной точностью повторять реальные процессы распространения огня, а также обеспечивать приемлемый уровень быстродействия для его использования в системах виртуальной реальности. В данной предметной области быстродействие является определяющим фактором, так как технологии виртуальной реальности предъявляют особые требования к производительности системы. Помимо реализации математической модели необходимо разработать новые инструменты формализации жизненных циклов виртуальных объектов, позволяющие отобразить процессы изменения внешних и внутренних характеристик объектов.

2. Метод формализации физических процессов в виртуальной реальности на основе диаграмм жизненных циклов

На первом этапе решения поставленной задачи осуществлен выбор математического аппарата теории множеств для формализации физических процессов. Однако он позволяет сформулировать структуру и связи между виртуальными объектами, но не обеспечивает достаточной визуализации протекающих в виртуальной реальности процессов.

Поэтому для формализации жизненного цикла виртуальных объектов в процессе их возгорания, горения, тушения или полного уничтожения необходимо использовать графические представления. Обозначим их как диаграммы жизненного цикла. В предлагаемых диаграммах отображается жизненный цикл каждого виртуального объекта. На горизонтальной оси диаграммы отмечены пороговые значения объекта, а на вертикальной — последовательность событий (рис. 1).

Под пороговыми значениями понимается статус виртуального объекта по шкале от "возгорание невозможно" до "максимальное воспламенение". Тогда диаграммы жизненного цикла отражают смену пороговых значений объекта в рамках одного события в течение времени. Возникновение, окончание или смену какого-либо воздействия на объекты обозначим как событие. События отражены по горизонтальной оси и протекают в соответствии с указанным направлением. Переход по вертикали снизу вверх отражает смену событий.

Смена пороговых значений зависит от совокупности внешних воздействий, а также собственных свойств объекта. Рассмотрим перечень пороговых значений:

"–2" — возгорание абсолютно невозможно. Достигается при сильном воздействии на объект водой, пеной или иным веществом;

"–1" — граница между "мокрым" и "сухим" состоянием объекта;

"0" — нормальное состояние, к которому стремится объект при отсутствии различных воздействий. После полного разрушения (сгорания) объект также возвращается в данное состояние;

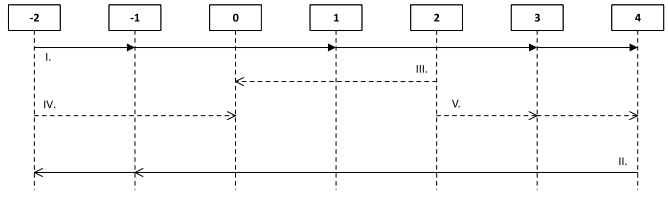


Рис. 1. Общий вид диаграммы жизненного цикла виртуального объекта

"1" — начальная стадия возгорания объекта. Осуществляется визуализация дымообразования, без открытого пламени и изменения цвета;

"2" — граница невозврата объекта к нормальному состоянию;

"3" — открытое пламя. Визуализация открытого пламени, дымобразования и изменения цвета объекта;

"4" — максимальная степень возгорания объекта. Параметры высоты языков пламени, дымообразования и температурного воздействия имеют максимальные значения.

Для некоторых объектов отдельные пороговые значения могут отсутствовать из-за типа материала или его свойств.

Вторая составляющая диаграммы жизненного цикла объекта — это правила смены состояний. На рис. 1 они обозначены римскими цифрами и отражают реальные физические процессы, протекающие в объектах при том или ином виде воздействия или же его отсутствии. Рассмотрим основные правила.

Первое правило описывает процесс смены состояний объекта при положительном для процесса горения внешнем воздействии (например, огонь). Необходимое условие: непрерывность воздействия. Прочность объекта начинает снижаться только после достижения состояния "2".

Второе правило отражает смену состояний объекта при отрицательном для процесса горения внешнем воздействии (например, вода). Условие: непрерывность. В процессе воздействия прочность объекта может постепенно снижаться, если ранее он горел (вплоть до момента достижения состояния "0").

Третье правило возвращает объект в нейтральное состояние "0". Условие: прекращение положительного воздействия в пределах пороговых значений от "0" до "2".

Четвертое правило возвращает объект в нейтральное состояние "0", но со стороны отрицательных пороговых значений, т. е. в процессе высыхания объекта после его тушения водой. Условие: прекращение отрицательного воздействия в пределах пороговых значений от "–2" до "–1".

Пятое правило отражает стремление объекта к воспламенению даже после прекращения положительного воздействия. Условие: прекращение положительного воздействия в пределах пороговых значений от "2" до "4".

Таким образом, можно условно выделить две группы воздействий: положительные для процесса горения (огонь, искры или высокая температура) и отрицательные (вода, пена, песок). Положительные воздействия влияют на развитие процесса горения, а отрицательные приводят к его прекращению.

Любой вид воздействия имеет различную силу в зависимости как от типа источника этого воздействия, так и материала объекта. К воздействиям также можно отнести некоторые параметры окружающей среды: объем виртуального помещения; циркуляция воздуха; электрическое оборудование и провода; доступность кислорода и герметичность; температура помещения; загазованность и задымленность; токсичность продуктов горения; влажность.

Таким образом, представленный подход к формализации процесса горения виртуальных объектов включает основные факторы, влияющие на состояние объектов, отражает их жизненный цикл в процессе горения и тушения, а также правила смены состояний.

3. Математическая модель изменения состояний виртуальных объектов

Далее на основе определенных выше понятий и метода формализации сформулируем математическую модель изменений состояний виртуальных объектов.

Обозначим O — множество объектов горения. Под каждым $o_i \in O$ понимается трехмерная модель реального объекта с присущими ему основными свойствами, влияющими непосредственно только на процесс горения. Рассмотрим основные параметры объектов горения и связанные с ними внешние сущности, представленные на схеме (рис. 2).

К основным параметрам объекта $o_i \in O$ можно отнести размер Z_i, массу W_i и тип материала M_i, из которого этот объект сделан, прочность объекта H_i. К дополнительным параметрам относятся: температура T_i, текущее пороговое значение состояния объекта S_i и скорость горения I_i. Тогда:

$$o_i = (Z_i, W_i, M_i, H_i, I_i, T_i, S_i). \quad (1)$$

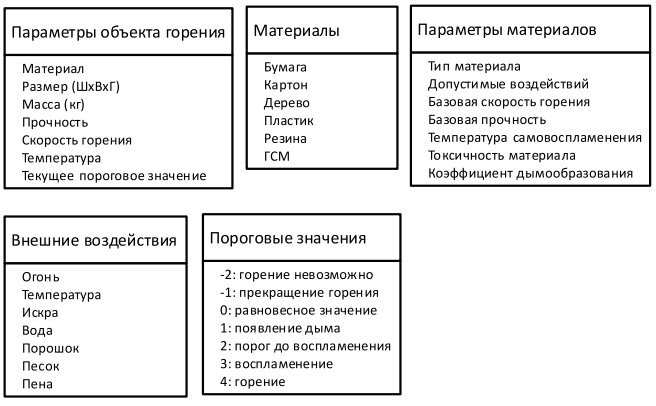


Рис 2. Основные сущности процесса горения

Прочность объекта H_i определяет время, в течение которого объект может выдержать негативное влияние огня, и зависит от размера, массы и типа материала объекта:

$$(Z_i, W_i, M_i) \to H_i. \quad (2)$$

Прочность объекта влияет на время от момента воспламенения объекта и до его полного сгорания, когда объект больше не может воспламеняться.

Скорость горения I_i объекта в следующий момент времени $t + \Delta t$ определяется эмпирической функцией F_s, зависящей от следующих параметров: текущей скорости горения $I_i(t)$ и состояния объекта $S_i(t)$. Различные внешние воздействия могут увеличивать (при усилении пламени) или уменьшать (при тушении) ее значение:

$$I_i(t + \Delta t) = F_S(S_i(t), I_i(t), M_i). \quad (3)$$

Таким образом, на основе соотношения (3) можно определить прочность объекта следующим образом:

$$H_i(t + \Delta t) = H_i(t) - I_i(t)\Delta t, \quad (4)$$

где Δt — время горения.

Температура T_i объекта в момент времени $t + \Delta t$ определяется функцией F_T и зависит как от общей температуры окружения T_E, так и температуры j-х объектов, расположенных вблизи i-го объекта, в момент времени t:

$$T_i(t + \Delta t) = T_i(t) + F_T(T_E(t), \{T_j(t) \mid j \neq i\}). \quad (5)$$

Основной материал объекта M_i также имеет ряд характеристик, участвующих в процессе моделирования процесса горения:

- тип материала (бумага, картон, дерево, пластик, резина, горюче-смазочные жидкости) задает основные свойства объекта;
- допустимые воздействия определяют перечень возможных воздействий для данного материала;
- базовая скорость горения и базовая прочность являются коэффициентами при расчете скорости горения и прочности конкретного объекта;
- температура самовоспламенения задает границу самовозгорания объекта;
- токсичность материала при горении оказывает влияние на состояние дыхательной системы человека в случае отсутствия средств защиты.

Пороговые состояния объекта S_i сменяют друг друга непрерывно с некоторым шагом, таким образом, определить текущее состояние объекта в диапазоне между двумя пороговыми значениями можно по следующей формуле:

$$S_i(t + \Delta t) = S_i(t) + k\Delta t \frac{|T_i(t) - T_j(t)|}{T_i(t)}, \quad (6)$$

где k — коэффициент, определяющий направление и интенсивность изменения порогового значения объекта, принимает значения в интервале $\{-1; 1\}$. Для некоторых материалов (например, бензин) k стремится к значениям в интервале $\{-1000; 1000\}$ для того, чтобы проходить некоторые пороговые значения за максимально короткий промежуток времени; Δt — время изменения порогового состояния; T_i, T_j — температура объекта и внешней среды (воздействия) соответственно в момент времени t.

Помимо параметров виртуального объекта на процесс горения оказывают существенное влияние характеристики окружения P_E и множество воздействий S_E.

Характеристики виртуального окружения определяют объем помещения Q_E, параметры вентиляции V_E, электричества E_E, состояние герметизации H_E, количество кислорода O_E, задымленность S_E, токсичность X_E, температуру T_E и влажность M_E в помещении:

$$P_E = (Q_E, V_E, E_E, H_E, O_E, S_E, X_E, T_E, M_E). \quad (7)$$

Воздействия (события) $S_E = \{s_{e,j}\}$ направлены на изменение характеристик объектов и окружения виртуальной реальности при движении процесса горения либо в положительную сторону (горение, нагревание), либо в отрицательную (тушение, остывание). Воздействие включает следующие компоненты:

$$s_{e,j} = (\lambda_j, T_j^{se}, t_j), \quad (8)$$

где λ_j — коэффициент, определяющий интенсивность воздействия; T_j^{se} — температура воздействия; t_j — общее время воздействия, включающее временные интервалы $t_{j,k}$, соответствующие различным пороговым значениям наблюдаемого объекта.

Изменение температуры объекта зависит от следующих факторов:

- горение объекта

$$T_i(t + \Delta t) = T_i(t) + \lambda_j \frac{H_i(t)}{H_0} t_j (T_j^{se} - T_i(t)); \quad (9)$$

- тушение объекта

$$T_i(t + \Delta t) = T_i(t) - \lambda_j \frac{H_i(t)}{H_0} t_j (T_i(t) - T_j^{se}); \quad (10)$$

- влияние окружающей среды

$$T_i(t + \Delta t) = T_i(t) + \lambda_E t_E (T_i(t) - T_E(t)), \quad (11)$$

где λ_E — коэффициент, определяющий направление и интенсивность влияния окружающий

среды, зависящий от параметров окружения, принимает положительные значения, если объект холоднее окружения, и отрицательные — в противном случае; t_E — время влияния окружающей среды; $H_i(t)$ и H_0 — текущая и начальная прочность объекта соответственно.

По формулам (9)—(11) проводится расчет температуры на каждом временном интервале от одного порогового значения до другого.

Используя представленную выше математическую модель изменения состояний виртуальных объектов, диаграммы пороговых значений и правила смены состояний, можно осуществить формализацию процесса горения объектов виртуальной реальности.

4. Практическая реализация

В качестве примера рассмотрим упрощенный сценарий возгорания объекта из дерева в качестве основного материала и тушение его с помощью воды.

Для формализации процесса смены событий в рамках виртуальной реальности кроме диаграмм жизненного цикла виртуальных объектов предлагается использовать матричное представление, включающее все атрибуты объектов, окружающего пространства, источников огня и т. д. Складывая и вычитая соответствующие строки матриц, оценивая полученные значения атрибутов, формализуем смену состояний объектов в процессе горения и тушения.

Для лучшего понимания и сокращения объемов аналитических расчетов в рамках данной статьи введем следующее допущение: расчет температур ведется дискретно по каждому пороговому значению, а не непрерывно в каждый момент времени. При программной реализации математической модели данное допущение снимается. Второе допущение: в данной статье не приводятся расчеты по формуле (6), так как в программном обеспечении коэффициенты k задаются эмпирически для каждого перехода из одного порогового состояния в другое, и представлять данные расчетные формулы в рамках рассматриваемых примеров не имеет смысла. Третье допущение: для сокращения объема расчетных формул примем следующую форму записи:

$$T = 100 \to T = 200 \equiv T = 100 \to 200. \quad (12)$$

Сценарий заключается в моделировании процесса возгорания объекта при периодическом воздействии открытым огнем с последующей ликвидацией возгорания.

Диаграмма на рис. 3 отображает жизненный цикл объекта в ходе первого сценария при возникновении события возгорания и последующего тушения. Длительность каждого порогового состояния в рамках события обозначена $t_{j,k}$.

Изначально наблюдаемый объект o находится в нейтральном пороговом состоянии "0". Состояние окружения P_E также условно нейтральное:

$$o + P_E = \begin{bmatrix} H = 5000 \\ S = 0 \\ T = 25 \\ I = 0 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,02 \\ T_E = 25 \end{bmatrix} = \begin{bmatrix} H = 5000 \\ S = 0 \\ T = 25 \\ I = 0 \end{bmatrix}. \quad (13)$$

В записи (13) и далее знаки равенства внутри квадратных скобок носят условный характер и используются для наглядного отображения текущих значений переменных.

В определенный момент времени происходит событие $s_{e,I}$ — прямое воздействие открытого огня на наблюдаемый объект в течение $t_I = 5$ с до наступления события $s_{e,II}$. В течение времени воздействия объект достигает состояния "1" за 3 с, после чего начинается отображаться дым, и объект достигает промежуточного состояния "1.7" за 2 с:

$$o + s_{e,I} + P_E = \begin{bmatrix} H = 5000 \\ S = 0 \\ T = 25 \\ I = 0 \end{bmatrix} + \begin{bmatrix} \lambda_I = 0,05 \\ T_I^{se} = 500 \\ t_I = 5 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,02 \\ T_E = 25 \\ t_E = 2 \end{bmatrix} = \begin{bmatrix} H = 5000 \\ S = 1,7 \\ T = 120 \\ I = 0 \end{bmatrix}, \quad (14)$$

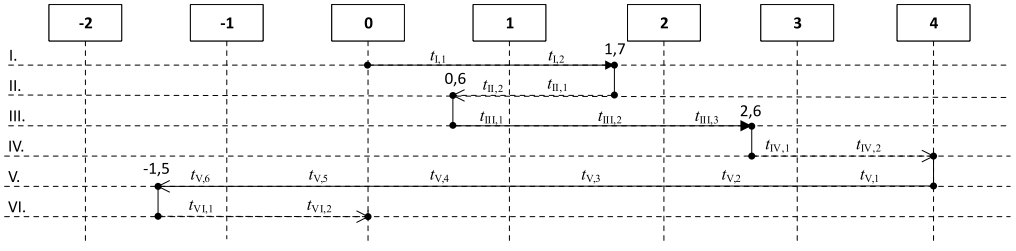


Рис. 3. Диаграмма жизненного цикла объекта по первому сценарию

$$t_I = \begin{cases} t_{I,1} = 3, S = 0 \to 1, \\ T = 25 \to 100, H = 5000; \\ t_{I,2} = 2, S = 1 \to 1,7, \\ T = 100 \to 137, H = 5000. \end{cases} \quad (15)$$

Отметим, что на этапе $t_{I,2}$ окружающая среда уже оказывает свое влияние из-за разницы температур между объектом и средой. После окончания воздействия согласно правилам переходных состояний объект из промежуточного состояния "1.7" будет стремиться к пороговому значению "0" под воздействием нейтральной окружающей среды. Таким образом, начинается событие $s_{e,\,II} = P_E$ — прекращение воздействия открытым пламенем на объект в течение $t_{II} = 30$ с (процесс остывания):

$$o + P_E = \begin{bmatrix} H = 5000 \\ S = 1,7 \\ T = 137 \\ I = 0 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,02 \\ T_E = 25 \\ t_E = 30 \end{bmatrix} = \begin{bmatrix} H = 5000 \\ S = 0,6 \\ T = 80 \\ I = 0 \end{bmatrix}, \quad (16)$$

$$t_{II} = \begin{cases} t_{II,1} = 16, S = 1,7 \to 1, \\ T = 137 \to 100, H = 5000 \to 5000; \\ t_{II,2} = 14, S = 1 \to 0,6, \\ T = 100 \to 80, H = 5000 \to 5000. \end{cases} \quad (17)$$

Следующее событие $s_{e,\,III}$ — возобновление воздействия открытого пламени, аналогично $s_{e,\,I}$, в течение $t_{III} = 14$ с:

$$o + s_{e,III} + P_E = \begin{bmatrix} H = 5000 \\ S = 0,6 \\ T = 80 \\ I = 0 \end{bmatrix} + \begin{bmatrix} \lambda_{III} = 0,05 \\ T_{III}^{se} = 500 \\ t_{III} = 14 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,02 \\ T_E = 25 \\ t_E = 14 \end{bmatrix} = \begin{bmatrix} H = 4737 \\ S = 2,6 \\ T = 310 \\ I = 25 \end{bmatrix}, \quad (18)$$

$$t_{III} = \begin{cases} t_{III,1} = 1, S = 0,6 \to 1, \\ T = 80 \to 100, H = 5000 \to 5000; \\ t_{III,2} = 2,5, S = 1 \to 2, \\ T = 100 \to 150, H = 5000 \to 5000; \\ t_{III,3} = 10,5, S = 2 \to 2.6, \\ T = 150 \to 310, H = 5000 \to 4737,5. \end{cases} \quad (19)$$

Так как объект преодолел пороговое значение "2", то процесс горения становится необратимым даже без внешнего источника пламени (в этом случае считается, что объект сам становится источником пламени для самого себя). Температура объекта также начинает влиять на температуру окружения и степень его влияния на объект, что приведет к изменению значений следующих параметров внешней среды: $T_E = 40$, $\lambda_E = 0,01$. Это отражается в событии $s_{e,\,IV}$, протекающем в течение $t_{IV} = 20$ с:

$$o + s_{e,IV} + P_E = \begin{bmatrix} H = 4737 \\ S = 2,6 \\ T = 310 \\ I = 25 \end{bmatrix} + \begin{bmatrix} \lambda_{IV} = 0,05 \\ T_{IV}^{se} = 500 \\ t_{IV} = 20 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,01 \\ T_E = 40 \\ t_E = 20 \end{bmatrix} = \begin{bmatrix} H = 4400 \\ S = 4 \\ T = 390 \\ I = 50 \end{bmatrix}, \quad (20)$$

$$t_{IV} = \begin{cases} t_{IV,1} = 11, S = 2,6 \to 3, \\ T = 310 \to 380, H = 4737,5 \to 4460; \\ t_{IV,2} = 9, S = 3 \to 4, T = 380 \to 390, \\ I = 25 \to 50, H = 4460 \to 4400. \end{cases} \quad (21)$$

Следующим происходит событие $s_{e,\,V}$ — воздействие человека в целях ликвидации возгорания с использованием воды в течение $t_V = 49$ с:

$$o + s_{e,V} + P_E = \begin{bmatrix} H = 4400 \\ S = 4 \\ T = 390 \\ I = 50 \end{bmatrix} + \begin{bmatrix} \lambda_V = -0,05 \\ T_V^{se} = 10 \\ t_V = 49 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,01 \\ T_E = 40 \\ t_E = 49 \end{bmatrix} = \begin{bmatrix} H = 3420 \\ S = -1,5 \\ T = 70 \\ I = 0 \end{bmatrix}, \quad (22)$$

$$t_V = \begin{cases} t_{V,1} = 6,5, S = 4 \to 3, T = 390 \to 300, \\ I = 50 \to 25, H = 4400 \to 4070; \\ t_{V,2} = 7, S = 3 \to 2, \\ T = 300 \to 230, H = 4070 \to 3890; \\ t_{V,3} = 10, S = 2 \to 1, \\ T = 230 \to 160, H = 3890 \to 3640; \\ t_{V,4} = 8,5, S = 1 \to 0, T = 160 \to 120, \\ I = 25 \to 0, H = 3640 \to 3420; \\ t_{V,5} = 9, S = 0 \to -1, \\ T = 120 \to 90, H = 3420; \\ t_{V,6} = 8, S = -1 \to 1, \\ T = 90 \to 70, H = 3420. \end{cases} \quad (23)$$

Изменение состояний объекта происходит вплоть до порогового значения "–1,5" за время t_V, когда горение объекта прекращается, и пов-

торное воспламенение без постороннего воздействия невозможно.

В точке "–1,5" прекращается подача воды и, соответственно, ее воздействие — наступает событие $s_{e,\,VI} = P_E$. По правилу стремления объекта в нейтральное состояние "0" после прекращения воздействия водой в течение времени $t_{VI} = 108$ с под воздействием только окружающей среды объект переходит в состояние "0", в котором и остается до наступления новых событий:

$$o + P_E = \begin{bmatrix} H = 3420 \\ S = -1,5 \\ T = 70 \\ I = 0 \end{bmatrix} + \begin{bmatrix} \lambda_E = 0,01 \\ T_E = 40 \\ t_E = 108 \end{bmatrix} = \begin{bmatrix} H = 3420 \\ S = 0 \\ T = 25 \\ I = 0 \end{bmatrix}, \quad (24)$$

$$t_{II} = \begin{cases} t_{II,1} = 66,\ S = -1,5 \to -1, \\ T = 70 \to 50,\ H = 3420; \\ t_{II,2} = 62,\ S = -1 \to 0, \\ T = 50 \to 25,\ H = 3420. \end{cases} \quad (25)$$

Это вновь приведет к изменению значений следующих параметров внешней среды: $T_E = 30$, $\lambda_E = 0,02$.

Далее на основе представленной формализации осуществляется программная реализация на языке программирования C# в среде разработки Unity3d. Так как при разработке математической модели мы учитывали объектно-ориентированную парадигму, используемую в Unity3d, перенос структуры объектов и процессов горения в программный код осуществляется достаточно быстро и просто. Пример процесса горения по рассмотренному сценарию представлен на рис. 4.

Универсальность представленных подходов позволяет моделировать процессы горения объектов самых различных форм, материалов и размеров, а также вне зависимости от параметров помещения успешно осуществлять реалистичную визуализацию горения и тушения объектов.

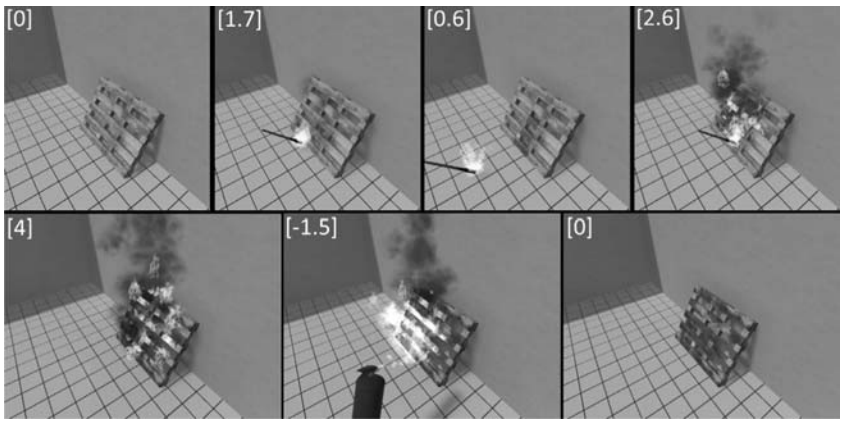


Рис 4. Программная реализация процесса горения

Заключение

В рамках данной статьи рассматривается важный аспект проектирования АТК — формализация физических процессов в виртуальном пространстве на примере горения объектов. Данный процесс достаточно исследован, однако существующие математические модели не адаптированы для использования в тренажерных комплексах, приводят к значительному повышению стоимости АТК и снижению производительности из-за сложности расчетов. Поэтому предлагается адаптированная математическая модель, позволяющая формализовать структуру объектов горения, основные их свойства и процессы смены состояний, начиная от возгорания объекта и заканчивая их полным уничтожением или тушением.

Кроме того, для решения задачи формализации процесса горения предлагаются оригинальные диаграммы жизненного цикла виртуальных объектов и система правил смены их состояний, позволяющие проследить весь жизненный цикл объектов и процедуру смены состояний. Закономерности смены состояний формализованы с помощью матричного представления свойств объектов. Данный подход отличается своей универсальностью и может использоваться для моделирования других физических процессов (задымления, затопления, замерзания, распространения токсичных газов и т.д.). Таким образом, проведенные научные исследования могут использоваться для формализации физических процессов в тренажерных комплексах для эргатических систем профессионального назначения.

Список литературы

1. **Kunkler K.** The role of medical simulation: an overview // Int. J. Med. Robot. Comput. Assist. Surg. 2006. Vol. 2, N. 3. P. 203—210.
2. **Manca D., Brambilla S., Colombo S.** Bridging between Virtual Reality and accident simulation for training of process-industry operators // Adv. Eng. Softw. Elsevier Ltd. 2013. Vol. 55. P. 1—9.
3. **Strayer D. L., Drews F. A.** Simulator training improves driver efficiency: Transfer from the simulator to the real world // Proceedings of the Second International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design. 2003. P. 190—193.
4. **Patle D. S., Ahmad Z., Rangaiah G. P.** Operator training simulators in the chemical industry: review, issues, and future directions // Rev. Chem. Eng. 2014. Vol. 30, N. 2. P. 199—216.
5. **Kuligowsky E. D., Peacock R. D.** A rewiew of evacuation models. National Institute of Standards and Technology, U. S. Department of Commerce, Technical note., 2005.

6. **Rushmeier H. E., Hamins A., Choi M. Y.** Case Study: Volume Rendering of Pool Fire Data // IEEE Comput. Graph. Applications. 1994. Vol. 4, N. 15. P. 382—385.

7. **Schadschneider A.** et al. Evacuation Dynamics: Empirical Results, Modeling and Applications // Encyclopedia of Complexity and Systems Science. New York, NY: Springer New York, 2009. P. 3142—3176.

8. **Krüger J., Westermann R.** GPU Simulation and Rendering of Volumetric Effects for Computer Games and Virtual Environments // Comput. Graph. Forum. 2005. Vol. 24, N. 3. P. 685—693.

9. **Cha M.** et al. A virtual reality based fire training simulator integrated with fire dynamics data // Fire Saf. J. Elsevier, 2012. Vol. 50. P. 12—24.

10. **Nguyen D. Q., Fedkiw R., Jensen H. W.** Physically based modeling and animation of fire // ACM Trans. Graph. 2002. Vol. 21, N. 3. P. 721—728.

11. **Krasnyanskiy M. N., Ostroukh A. V., Karpushkin S. V., Obukhov A. D.** Formulation of the Problem of Structural and Parametric Synthesis of Electronic Document Management System of Research and Education Institution. Global Journal of Pure and Applied Mathematics. 2016. Vol. 12, N. 3. P. 2395—2409.

12. **Rosenkopf L., Tushman M. L.** The Coevolution of Community Networks and Technology: Lessons from the Flight Simulation Industry // Ind. Corp. Chang. 1998. Vol. 7, N. 2. P. 311—346.

A. D. Obukhov, Assistant Professor, e-mail: Obuhov.art@gmail.com,
A. E. Arkhipov, Postgraduate Student, e-mail: alexeiarh@gmail.com,
A. O. Sidorchuk, Bachelor, e-mail: sidorchuk.a.o@yandex.ru,
Tambov State Technical University, Tambov, 392000, Russian Federation

Mathematical Modeling and Visualization of Combustion Processes in the Training Facilities

An important task in the development of training complexes is the implementation of physical processes in virtual space using the example of burning objects. To solve it, an adapted mathematical model is proposed that allows one to formalize the structure of combustion objects, their main properties and processes of changing states, starting from the ignition of the object and ending with their complete destruction or extinguishing. Original diagrams of the life cycle of virtual objects and a system of rules for changing their states have been developed, which allow tracing the entire life cycle of objects and the procedure for changing states. Each state and transition between them are clearly formalized and calculated on the basis of the properties of the object and external influences. The patterns of state change are formalized using a matrix representation of object properties. This approach is distinguished by its versatility and can be used to model other physical processes (smoke, flooding, freezing, spread of toxic gases, etc.). The practical implementation of the combustion process using the apparatus of life cycle diagrams of virtual objects has been carried out. The developed method has been successfully integrated into the Unity development environment. Its use made it possible to reliably and efficiently implement the combustion process without losing performance, which is especially important for virtual reality systems.

Keywords: *computer graphics, 3D modeling, visualization of physical processes, adaptive training complexes, formalization of the combustion process, model of states of virtual objects, set theory, personnel training*

Acknowledgements: The study was carried out with the financial support of the Russian Foundation for Basic Research within the framework of scientific project No. 19-013-00567 using the computing equipment of the Center for Collective Use Digital Engineering.

DOI: 10.17587/it.27.227-234

References

1. **Kunkler K.** The role of medical simulation: an overview, *Int. J. Med. Robot. Comput. Assist. Surg*, 2006, vol. 2, no. 3, pp. 203—210.

2. **Manca D., Brambilla S., Colombo S.** Bridging between Virtual Reality and accident simulation for training of process-industry operators, *Adv. Eng. Softw. Elsevier Ltd*, 2013, vol. 55, pp. 1—9.

3. **Strayer D. L., Drews F. A.** Simulator training improves driver efficiency: Transfer from the simulator to the real world, *Proceedings of the Second International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design*, 2003, pp. 190—193.

4. **Patle D. S., Ahmad Z., Rangaiah G. P.** Operator training simulators in the chemical industry: review, issues, and future directions, *Rev. Chem. Eng*, 2014, vol. 30, no. 2, pp. 199—216.

5. **Kuligowsky E. D., Peacock R. D.** A rewiew of evacuation models. National Institute of Standards and Technology, U. S. Department of Commerce, *Technical note,* 2005, p. 156.

6. **Rushmeier H. E., Hamins A., Choi M. Y.** Case Study: Volume Rendering of Pool Fire Data, *IEEE Comput. Graph. Applications*, 1994, vol. 4, no. 15, pp. 382—385.

7. **Schadschneider A., Klingsch W., Kl@ pfel H., Kretz T., Rogsch C., Seyfried A.** Evacuation Dynamics: Empirical Results, Modeling and Applications, Encyclopedia of Complexity and Systems Science, *NY: Springer New York*, 2009, pp. 3142—3176.

8. **Krüger J., Westermann R.** GPU Simulation and Rendering of Volumetric Effects for Computer Games and Virtual Environments, *Comput. Graph. Forum,* 2005, vol. 24, no. 3, pp. 685—693.

9. **Cha M. Han S., Lee J., Choi B.,** A virtual reality based fire training simulator integrated with fire dynamics data, *Fire Saf. J. Elsevier*, 2012, vol. 50, pp. 12—24.

10. **Nguyen D. Q., Fedkiw R., Jensen H. W.** Physically based modeling and animation of fire, *ACM Trans. Graph*. 2002, vol. 21, no. 3, pp. 721—728.

11. **Krasnyanskiy M. N., Ostroukh A. V., Karpushkin S. V., Obukhov A. D.** Formulation of the Problem of Structural and Parametric Synthesis of Electronic Document Management System of Research and Education Institution, *Global Journal of Pure and Applied Mathematics,* 2016, vol. 12, no. 3, pp. 2395—2409.

12. **Rosenkopf L., Tushman M. L.** The Coevolution of Community Networks and Technology: Lessons from the Flight Simulation Industry, *Ind. Corp. Chang*, 1998, vol. 7, no. 2, pp. 311—346.

УДК 519.7, 519.8 DOI: 10.17587/it.27.235-241

С. И. Колесникова, д-р техн. наук, проф., skolesnikova@yandex.ru,
С. А. Караванова, магистрант, sofy.karav@gmail.com,
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Оптимизация алгоритма многокритериального выбора с динамически пополняемым большим набором альтернатив*

Корректное ранжирование динамически пополняемого большого набора альтернатив в многокритериальных задачах выбора основано на алгоритме корректного учета предпочтений на множестве пар относительных весов сравниваемых между собой координат собственных векторов матриц, применяемых в классическом алгоритме анализа иерархий. В таком исполнении алгоритм обеспечивает сохранение ранее достигнутых предпочтений при добавлении новых альтернатив и, тем самым, дает возможность оптимизации при обработке больших объемов динамически изменяемых данных, что существенно расширяет применимость популярного алгоритма.

Ключевые слова: *большие данные, корректный алгоритм ранжирования альтернатив, оптимизация алгоритма ранжирования большого набора альтернатив, показатели эффективности алгоритма попарных сравнений, алгоритмическая сложность*

Введение

Хорошо известна проблема, характерная для методов многокритериального выбора, использующих попарные сравнения альтернатив по каждому из критериев, а затем аддитивную скаляризацию полученных локальных приоритетов и весовых коэффициентов критериев [1—6]: при добавлении новой альтернативы ранее достигнутые предпочтения могут измениться [2, 4, 5]. Указанная нежелательная особенность классического метода анализа иерархий (Analytic Hierarchy Process (AHP)) ограничивает его применение для динамически пополняемых наборов данных большого объема (рекрутинг, оценивание проектов, ...).

Пример 1. Действительно, рассмотрев сначала две альтернативы A_1, A_2 по двум критериям C_1, C_2 с весовыми коэффициентами $w(C_1) = 0{,}4$; $w(C_2) = 0{,}6$ с матрицами парных сравнений (МПС) вида $\mathbf{A}_1 = \begin{bmatrix} 1 & 1/3 \\ 3 & 1 \end{bmatrix}$; $\mathbf{A}_2 = \begin{bmatrix} 1 & 4 \\ 1/4 & 1 \end{bmatrix}$, получим итоговый вектор предпочтений, из которого следует $w_{n=2}(A_1) = 0,58 > w_{n=2}(A_2) = 0,42$.

Увеличив множество альтернатив до трех и оставляя оценки первых двух альтернатив без изменения, получим, согласно классическому AHP, МПС вида

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 1/3 & 3 \\ 3 & 1 & 3 \\ 1/3 & 1/3 & 1 \end{bmatrix}; \ \mathbf{B}_2 = \begin{bmatrix} 1 & 4 & 1/3 \\ 1/4 & 1 & 1/8 \\ 3 & 8 & 1 \end{bmatrix},$$

приводящие к изменившимся предпочтениям относительно исходных двух альтернатив $w_{n=3}(A_1) = 0,266 < w_{n=3}(A_2) = 0,278, w_{n=3}(A_3) = 0{,}456$. В работе [5] предложен и далее неоднократно применен в задачах тестового распознавания и распознавания состояний динамических систем (например, [6]) алгоритм, ранее названный модификацией [6] классического метода анализа иерархий AHP [1]. Более точным будет утверждение, что ниже обсуждаемый алгоритм есть синтез классического AHP и корректного (безошибочного на выборке альтернатив) алгоритма парных соотношений координат собственных векторов как характеристик МПС.

В данном тексте на этот алгоритм будем ссылаться как на AHP-C (Correct algorithm for ranking alternatives by comparing the coordinates of eigenvectors of matrices of pairwise comparisons).

Для лучшего понимания идеологии алгоритма AHP-C приведем краткое его изложение [5, 6] и пример применения, поскольку заложенная в нем идея оценивания альтернатив и вычисления весовых коэффициентов в каждой их паре привела к возможности его оптимизации для больших наборов исходных данных, суть которой показана ниже.

Алгоритм AHP-C

Шаг 1. Для заданных наборов критериев $C = \{C_1, C_2, \ldots, C_m\}$, альтернатив $A = \{a_1, a_2, \ldots, a_n\}$ сначала конструируются МПС (как и в классическом AHP алгоритме); итогом первого

*Исследование выполнено при частичной поддержке РФФИ, проект № 20-08-00747.

шага является набор нормализованных оценок собственных векторов для каждой МПС:

$$\mathbf{W}_1 = \begin{Vmatrix} w_{1C_1} & \dots & w_{nC_1} \\ \dots & \dots & \dots \\ w_{1C_m} & \dots & w_{nC_m} \end{Vmatrix}, \qquad (1)$$

где w_{iC_j} — оценка i-й альтернативы по C_j-му критерию, $i = \overline{1,n};\ j = \overline{1,m}$.

Шаг 2. Формируется набор из матриц $\mathbf{W}_{2j}, j = \overline{1,m}$, элементами которых являются двухкомпонентные векторы:

$$\mathbf{W}_{2j} = \begin{Vmatrix} (w_{1C_j}, w_{1C_j}) & (w_{1C_j}, w_{2C_j}) \dots & (w_{1C_j}, w_{nC_j}) \\ \dots & \dots & \dots \\ (w_{nC_j}, w_{1C_j}) & (w_{nC_j}, w_{2C_j}) \dots & (w_{nC_j}, w_{nC_j}) \end{Vmatrix}, \quad j = \overline{1,m}. \qquad (2)$$

Шаг 3. Нормализуются векторы $(\tilde{w}_{iC_j}, \tilde{w}_{kC_j})$, $i,k = \overline{1,n};\ j = \overline{1,m}$, матриц $\mathbf{W}_{2j}, j = \overline{1,m}$, по формулам

$$\tilde{w}_{iC_j} = \frac{w_{iC_j}}{w_{iC_j} + w_{kC_j}},\ \tilde{w}_{kC_j} = \frac{w_{kC_j}}{w_{iC_j} + w_{kC_j}}, \quad i,k = \overline{1,n};\ j = \overline{1,m}. \qquad (3)$$

Получаем совокупность матриц $\mathbf{W}_{3j}, j = \overline{1,m}$, элементами которых являются двухкомпонентные нормализованные векторы:

$$\mathbf{W}_{3j} = \begin{Vmatrix} (0,5;0,5) & (\tilde{w}_{1C_j}, \tilde{w}_{2C_j}) \dots & (\tilde{w}_{1C_j}, \tilde{w}_{nC_j}) \\ \dots & \dots & \dots \\ (\tilde{w}_{nC_j}, \tilde{w}_{1C_j}) & (\tilde{w}_{nC_j}, \tilde{w}_{2C_j}) \dots & (0,5;0,5) \end{Vmatrix}, \quad j = \overline{1,m}. \qquad (4)$$

Шаг 4. Формируется итоговая матрица, элементы которой суть векторы из взвешенных по критериям C_j компонент матриц $\mathbf{W}_{3j}$:

$$\mathbf{W}_4 = \begin{Vmatrix} (0,5;0,5) & \left(\sum_{j=1}^{m} C_j\tilde{w}_{1C_j}, \sum_{j=1}^{m} C_j\tilde{w}_{2C_j}\right) \dots & \left(\sum_{j=1}^{m} C_j\tilde{w}_{1C_j}, \sum_{j=1}^{m} C_j\tilde{w}_{nC_j}\right) \\ \dots & \dots & \dots \\ \left(\sum_{j=1}^{m} C_j\tilde{w}_{nC_j}, \sum_{j=1}^{m} C_j\tilde{w}_{1C_j}\right) & \left(\sum_{j=1}^{m} C_j\tilde{w}_{nC_j}, \sum_{j=1}^{m} C_j\tilde{w}_{2C_j}\right) \dots & (0,5;0,5) \end{Vmatrix}, \qquad (5)$$

элементы которой затем нормализуются по формулам, аналогичным (3). Нормализованные векторы полученной матрицы $\mathbf{U}$ обозначим как $(u_i^{(i,k)}, u_k^{(i,k)}), i,k = \overline{1,n}$. Здесь громоздкость обозначений вызвана необходимостью указать, что операция аддитивной скаляризации по критериям проводится для каждой пары $(i,k), i,k = \overline{1,n}$, отдельно:

$$u_i^{(i,k)} = \frac{\sum_{j=1}^{m} C_j\tilde{w}_{iC_j}}{\sum_{j=1}^{m} C_j\tilde{w}_{iC_j} + \sum_{j=1}^{m} C_j\tilde{w}_{kC_j}},$$
$$u_k^{(i,k)} = \frac{\sum_{j=1}^{m} C_j\tilde{w}_{kC_j}}{\sum_{j=1}^{m} C_j\tilde{w}_{iC_j} + \sum_{j=1}^{m} C_j\tilde{w}_{kC_j}}, \quad i,k = \overline{1,n}. \qquad (6)$$

Шаг 5. Формируется итоговый вектор весовых коэффициентов по формулам (суммы по первым компонентам каждой строки матрицы $\mathbf{W}_5 = \|(u_i^{(i,k)}, u_k^{(i,k)})\|, i,k = \overline{1,n}$)

$$\omega_{a_1} = \sum_{k=1}^{n} u_1^{(1,k)}, \omega_{a_2} = \sum_{k=1}^{n} u_2^{(2,k)}, \dots, \omega_{a_n} = \sum_{k=1}^{n} u_n^{(n,k)}, \qquad (7)$$

который при необходимости дальнейшего анализа полученного ранжирования нормализуется: $\mathbf{W}^* = (\omega^*_{a_1}, \omega^*_{a_2}, \dots, \omega^*_{a_n})$.

Изложенный подход позволил применять идеологию AHP-C для динамически пополняемых наборов альтернатив, на заботясь о проверке соблюдения ранее достигнутых предпочтений.

Пример 2. Приведем теперь кратко шаги AHP-C в условиях выше рассмотренного примера 1.

Согласно AHP-C (шаг 1) исходными данными для работы алгоритма являются нормализованные собственные векторы матриц $\mathbf{B}_1$, $\mathbf{B}_2$, здесь равные

$$\mathbf{W}_{11}: \ 0,281 \quad 0,584 \quad 0,135;$$
$$\mathbf{W}_{12}: \ 0,256 \quad 0,073 \quad 0,671.$$

Здесь и ниже первый индекс в $\mathbf{W}_{ij}$ указывает на номер шага алгоритма, второй — на номер критерия.

Далее формируются матрицы сравнений координат полученных собственных векторов, затем все полученные двумерные векторы нормализуются (шаги 2, 3).

Для полученного выше собственного вектора $\mathbf{W}_{11} = (0,281;\ 0,584;\ 0,135)$ исходная $\mathbf{W}_{21}$ и нормализованная $\mathbf{W}_{31}$ матрицы попарного сравнения его координат имеют вид, соответственно:

$$\mathbf{W}_{21} = \begin{pmatrix} (1;1) & (0,281;0,584) & (0,281;0,135) \\ (0,584;0,281) & (1;1) & (0,584;0,135) \\ (0,135;0,281) & (0,135;0,584) & (1;1) \end{pmatrix};$$

$$\mathbf{W}_{31} = \begin{pmatrix} (0,5;0,5) & \left(\boxed{0,325};0,675\right) & \left(\boxed{0,675};0,325\right) \\ (0,675;0,325) & (0,5;0,5) & (0,812;0,188) \\ (0,325;0,675) & (0,188;0,812) & (0,5;0,5) \end{pmatrix}.$$

Аналогичные матрицы получаем для собственного вектора $\mathbf{W}_{12} = (0,256;\ 0,073;\ 0,671)$:

$$\mathbf{W}_{22} = \begin{pmatrix} (1;1) & (0,256;0,073) & (0,256;0,671) \\ (0,073;0,256) & (1;1) & (0,073;0,671) \\ (0,671;0,256) & (0,671;0,073) & (1;1) \end{pmatrix};$$

$$\mathbf{W}_{32} = \begin{pmatrix} (0,5;0,5) & \left(\boxed{0,777};0,223\right) & \left(\boxed{0,276};0,724\right) \\ (0,223;0,777) & (0,5;0,5) & (0,098;0,902) \\ (0,724;0,276) & (0,902;0,098) & (0,5;0,5) \end{pmatrix}.$$

Замечание 1. В каждой паре фиксированная альтернатива имеет свой коэффициент значимости, определяющий вклад в результат сравнения именно в данной паре. В рамки обведены значения, участвующие в формуле их скаляризации как линейной свертки по весовым коэффициентам критериев в каждой паре нормализованных векторов матриц $\mathbf{W}_{31}$, $\mathbf{W}_{32}$.

На шаге 4 осуществляется операция аддитивной свертки по критериям и альтернативам в *каждой паре* (главную диагональ по понятной причине не меняем):

$$\mathbf{W}_4 = \begin{Bmatrix} (0,5;0,5) & w_{4(1,2)} & w_{4(1,3)} \\ w_{4(2,1)} & (0,5;0,5) & w_{4(2,3)} \\ w_{4(3,1)} & w_{4(3,2)} & (0,5;0,5) \end{Bmatrix},$$

где

$$w_{4(1,2)} = \left(\sum_{j=1}^{m} C_j \tilde{w}_{1C_j}, \sum_{j=1}^{m} C_j \tilde{w}_{2C_j} \right) =$$

$$= \left(\boxed{W_{C_1} 0,325 + W_{C_2} 0,777}; W_{C_1} 0,675 + W_{C_2} 0,223 \right);$$

$$w_{4(1,3)} = \left(\sum_{j=1}^{m} C_j \tilde{w}_{1C_j}, \sum_{j=1}^{m} C_j \tilde{w}_{3C_j} \right) =$$

$$= \left(\boxed{W_{C_1} 0,675 + W_{C_2} 0,276}; W_{C_1} 0,325 + W_{C_2} 0,724 \right);$$

$$w_{4(2,3)} = \left(\sum_{j=1}^{m} C_j \tilde{w}_{2C_j}, \sum_{j=1}^{m} C_j \tilde{w}_{3C_j} \right) =$$

$$= \left(W_{C_1} 0,812 + W_{C_2} 0,098; W_{C_1} 0,188 + W_{C_2} 0,902 \right).$$

Положив $w(C_1) = 0,4$; $w(C_2) = 0,6$ (см. пример 1), получаем матрицу $\mathbf{W}_4$ в явном виде и ее нормализованный аналог $\mathbf{U}$, соответственно:

$$\mathbf{U} = \begin{pmatrix} (0,5;0,5) & (0,596;0,404) & (0,436;0,564) \\ (0,404;0,596) & (0,5;0,5) & (0,384;0,616) \\ (0,564;0,436) & (0,616;0,384) & (0,5;0,5) \end{pmatrix}.$$

Согласно формуле (7) (шаг 5) получаем итоговый вектор весовых коэффициентов альтернатив:

$$\omega_{A_1} = \sum_{k=1}^{3} u_1^{(1,k)} = 0,5 + 0,596 + 0,436 = 1,532;$$

$$\omega_{A_2} = \sum_{k=1}^{3} u_2^{(2,k)} = 0,404 + 0,5 + 0,384 = 1,288;$$

$$\omega_{A_3} = \sum_{k=1}^{3} u_3^{(3,k)} = 0,564 + 0,616 + 0,5 = 1,680;$$

затем его нормализуем: $\mathbf{W}^* = (0,340;0,286;0,373)$.

Сравнение с классическим алгоритмом AHP $\mathbf{W}^{AHP} = (0,266;0,278;0,456)$, применение которого нарушило ранее достигнутое предпочтение $(A_1 \succ A_2)$, говорит в пользу алгоритма AHP-C.

Мотивация и постановка задачи оптимизации алгоритма AHP-C

Даны множество альтернатив $A = \{A_1, \ldots, A_n\}$ (n велико, $n \sim 1000$ и более) и множество критериев $C = \{C_1, \ldots, C_m\}$. Требуется получить ранжированный список альтернатив A^*, свободный от недостатка изменений ранее достигнутых предпочтений при добавлении новых альтернатив, при этом желательно, чтобы алгоритмическая сложность искомого алгоритма ранжирования была соизмерима с логарифмической.

Теоретически алгоритм AHP-C применим и для больших данных, однако его алгоритмическая сложность и требуемый достаточно большой объем оперативной памяти представляют собой определенное препятствие для практического применения, поскольку дополнительно к большому объему альтернатив (ранжирование клиентов, проектов, принятия решений в рекрутинге и т. д.) имеется необходимость получения достаточного объема сравнительной экспертной информации.

Пример 3. Задача выбора из 5000 альтернатив и двух критериев для числа с двойной точностью, размер которого 8 байт, характеризуется показателями (реализация выполнена на языке Java):

- $5000 \cdot 5000 \cdot 2 \cdot 8$ = 381,47 Мбайт — для МПС алгоритма AHP;
- $5000 \cdot 5000 \cdot 2 \cdot 2 \cdot 8$ = 762,94 Мбайт — для матриц относительных локальных приоритетов, требуемых для алгоритма AHP-C;

- 5000·5000·2·8 = 381,47 Мбайт — для обобщенной матрицы **U**, компонентами которой являются локальные весовые коэффициенты альтернатив относительно всей совокупности критериев.

Замечание 2. Полученные данные могут несколько отличаться в зависимости от выбранного типа данных для хранения вычислений и реализации. В приведенном выше примере 3 для хранения пар значений в массиве использовались ссылочные объекты. Размер объекта в этой конкретной среде выполнения рассчитывается как сумма размера ссылки и размера полей объекта. В нашем случае это 8 байт на хранение ссылки и 16 байт на хранение двух значений типа double. Если округлять размер данных до размера, кратному машинному слову [7, 8], тогда итоговый объем оперативной памяти вырастет не менее, чем в полтора раза.

Если данные МПС будут сразу удаляться (что для сред выполнения с автоматическим удалением неиспользуемых объектов имеет место), все равно потребуется около 3 Гб только на хранение промежуточного результата (здесь не принимается в расчет объем памяти для хранения информации об исходных данных).

Традиционно задача получения ранжированного списка в многокритериальной задаче решалась путем применения какого-либо алгоритма вычисления весовых коэффициентов альтернатив и последующей сортировкой, согласованной с их значениями. Далее будет показано, как изменение этой последовательности действий позволит снять ограничение на мощность множества альтернатив и ускорить вычисление.

Решение задачи

Для решения проблемы корректного применения AHP-C на большом наборе альтернатив предлагается декомпозиция исходной задачи выбора на всем множестве альтернатив на последовательность задач выбора на паре альтернатив, выполнение каждой из которых интерпретируется как функция сравнения двух элементов согласно обычному алгоритму сортировки числовой последовательности (например, вставками [9]).

Любой алгоритм сортировки базируется на сравнении элементов по какому-либо признаку(-ам). В связке с алгоритмом AHP-C сортировка будет происходить как последовательность следующих действий:

— получить два элемента из набора A согласно алгоритму сортировки;

— рассчитать для этой пары веса путем AHP-C;

—если $a_i > a_j$, выполнить перестановку, иначе перейти к следующему шагу.

Далее представлена реализация с использованием библиотечного метода sort, куда мы передаем функцию сравнения с AHP-C

```
alternatives.sort((o1, o2) -> {// o1, o2 -
текущие элементы сравнения в сортировке
List<Alternative> list = new ArrayList<>();
list.add(o1); // добавляем в список
list.add(o2);
calculateWeight(list, criteriaList); //
вычисляем веса
return Double.compare(list.get(1).getWeight(),
list.get(0).getWeight()); //return a1>a0
});
```

Очевидно, что такой подход предъявляет свои требования к используемым методам решения многокритериальных задач. В первую очередь это корректность получаемых результатов вычислений исходного алгоритма. Под корректностью здесь понимается отсутствие нарушения бинарных отношений транзитивности $a > b$ и $b > c \rightarrow a > c$, выражающих предпочтения альтернатив в отсортированной части ранжируемого набора.

На рис. 1 это явление проиллюстрировано на примере сортировки вставками: элемент $a[6]$ встал на неверную позицию, так как при контрольной проверке расчета весов в группе из первых трех элементов выяснилось, что он меньше 15.

Указанное условие означает, что при внесении во множество альтернатив нового элемента не должны нарушаться отношения доминирования между уже находящимися во множестве элементами.

Таким образом, рассматриваемый алгоритм оптимизации (назовем его AHP-CS — от Sorting an array of data) использует последовательные сравнения пар для расчета весовых коэффициентов всего множества альтернатив, что стало возможным только благодаря корректной модификации алгоритма AHP-C, которому отведена роль части сортирующей функции в новом алгоритме AHP+CS.

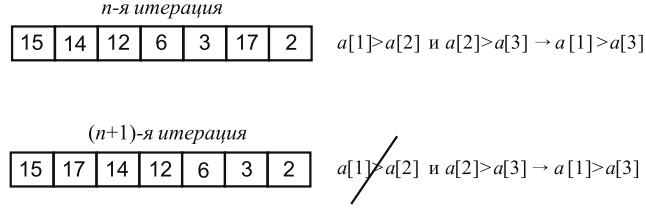


Рис. 1. Иллюстрация нарушения порядка следования

Численное моделирование алгоритмов AHP-C и AHP-CS. Анализ результатов

Алгоритмическая сложность алгоритма получения решения теперь зависит исключительно от числа сравнений в выбранном способе сортировки. На графике (рис. 2) приведены результаты тестирования времени выполнения AHP-C и AHP-CS на одинаковых наборах альтернатив. На рис. 3 также приведен график зависимости времени выполнения для AHP-CS на больших наборах альтернатив.

Требования к единовременно используемой оперативной памяти снижены. Теперь не нужно хранить результаты вычислений за исключением итоговых весов. Остальные объекты могут удаляться сборщиком мусора. К примеру, тест на 500 000 альтернатив и 2 критерия показал, что используется в тот момент только 140 Мб (рис. 4).

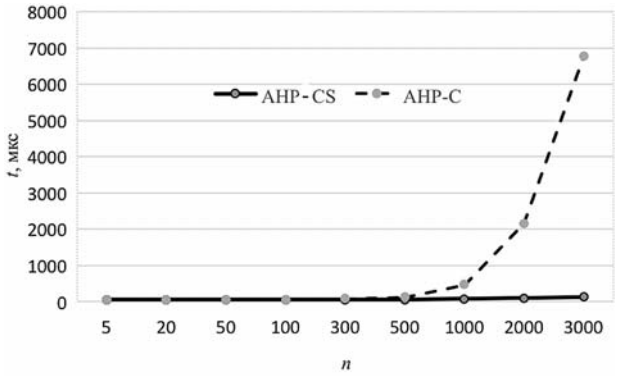


Рис. 2. График сравнения времени *t* выполнения для AHP-C и AHP-CS в зависимости от числа *n* альтернатив

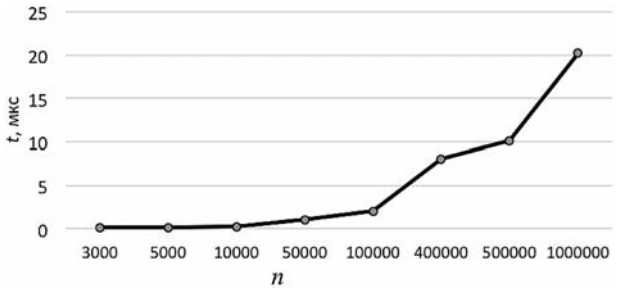


Рис. 3. График зависимости времени *t* выполнения от числа *n* альтернатив для AHP-CS

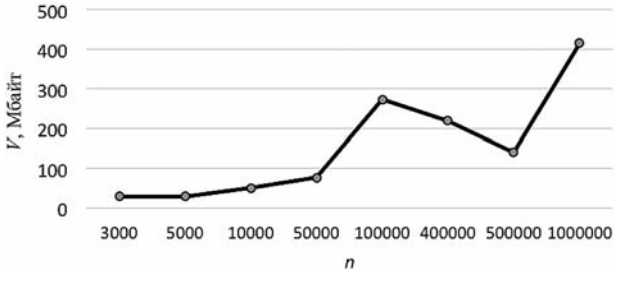


Рис. 4. График потребления объема *V* оперативной памяти в зависимости от числа *n* альтернатив для AHP-CS

Заключение

Основным результатом данной работы является распространение одного из самых популярных (по числу публикаций) методов решения многокритериальных задач выбора на наборы альтернатив и критериев большого объема в целях повышения скорости расчетов и снижения требований к вычислительным ресурсам. Поставленная задача оптимизации модифицированного алгоритма AHP-C выполнена на основе последовательного просмотра пар и их упорядочивания на принципах сортировки числового массива.

Исследуются алгоритмические сложности и затраты оперативной памяти существующих алгоритмов. Поясняются требования к алгоритмам с точки зрения корректности их работы на динамических множествах альтернатив.

Приводится иллюстративный пример и результаты тестирования предложенного алгоритма.

Важно заметить, что предложенная в статье модификация классического метода AHP стала возможной только при наличии алгоритма AHP-C, заведомо свободного от некорректного выбора в динамически изменяющихся наборах.

Согласно автору быстрой сортировки, "преждевременная оптимизация — корень всех бед" [10], поэтому осторожность в осуществлении любой оптимизации заключается в создании первоначально корректно работающего алгоритма (здесь — AHP-C).

Список литературы

1. **Саати Т. Л.** Принятие решений. Метод анализа иерархий. М.: Радио и связь, 1989. 311 с.
2. **Эрроу К. Дж.** Коллективный выбор и индивидуальные ценности. М.: ГУ ВШЭ, 2004. 204 с.
3. **Подиновский В. В., Ногин В. Д.** Парето-оптимальные решения многокритериальных задач. М.: Наука, 2007.
4. **Ногин В. Д.** Упрощенный вариант метода анализа иерархий на основе нелинейной свертки критериев // Журнал вычислительной математики и математической физики. 2004. Т. 44, № 7. С. 1259—1268.
5. **Самохвалов Ю. Я.** Групповой учет относительного превосходства альтернатив в задачах принятия решений // Кибернетика и системный анализ. 2003. № 6. С. 141—145.
6. **Колесникова С. И.** Особенности применения линейной свертки критериев в методе парных сравнений // Информационные технологии. 2011. № 1. С. 24—29.
7. **Oracle** documentation: Официальный сайт. URL: https://docs.oracle.com/en (дата обращения: 08.09.2020).
8. **Лафоре Р.** Структуры данных и алгоритмы Java. СПБ.: Питер, 2018. 704 с.
9. **Кнут Д. Э.** Искусство программирования. Т. 3. Сортировка и поиск / Пер. под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). М.: Вильямс, 2007. Т. 3. 832 с.
10. **Hoare C.** Quicksort // The Computer Journal. 1962. Vol. 5, N. 1. P. 10—16. DOI:10.1093/comjnl/5.1.10.

ПРИЛОЖЕНИЕ

Задача и ее решение на основе AHP-C и AHP-CS. Выполнить отбор наиболее перспективного кандидата на должность секретаря-референта по следующим критериям (табл. П1):

- уровень владения иностранным языком (ИЯ) — C_1;
- опыт работы — C_2;
- уровень владения офисными инструментами (ПК) — C_3.

Таблица П1

Исходные данные

Номер альтернативы	Уровень владения ИЯ, в баллах	Опыт работы, в годах	Уровень владения ПК, в баллах
1	7	9	5
2	8	10	8
3	9	5	2
4	1	5	9
5	2	3	5
6	8	8	4
7	3	7	4
8	7	10	3

Пусть $c_1 = 0{,}2$, $c_2 = c_3 = 0{,}4$, где $c_i = W(C_i)$ — весовой коэффициент i-го критерия, $i = 1{,}2{,}3$.

Этап 1. Реализация первых шагов классического AHP — составление матриц парных сравнений и получение соответствующих весовых оценок альтернатив по каждому критерию (расчеты сделаны с точностью до 10^{-3}).

МПС 1. Уровень владения ИЯ.

1,000 0,875 0,778 7,000 3,500 0,875 2,333 1,000
1,143 1,000 0,889 8,000 4,000 1,000 2,667 1,143
1,286 1,125 1,000 9,000 4,500 1,125 3,000 1,286
0,143 0,125 0,111 1,000 0,500 0,125 0,333 0,143
0,286 0,250 0,222 2,000 1,000 0,250 0,667 0,286
1,143 1,000 0,889 8,000 4,000 1,000 2,667 1,143
0,429 0,375 0,333 3,000 1,500 0,375 1,000 0,429
1,000 0,875 0,778 7,000 3,500 0,875 2,333 1,000

ВКА-1

$$W_{C_1} = (w_{1C_1}, \ldots, w_{8C_1}) =$$

$$= (0{,}156;\ 0{,}178;\ 0{,}200;\ 0{,}022;0{,}044;\ 0{,}178;\ 0{,}067;\ 0{,}156).$$

МПС 2. Опыт работы в годах.

1,000 0,900 1,800 1,800 3,000 1,125 1,286 0,900
1,111 1,000 2,000 2,000 3,333 1,250 1,429 1,000
0,556 0,500 1,000 1,000 1,667 0,625 0,714 0,500
0,556 0,500 1,000 1,000 1,667 0,625 0,714 0,500
0,333 0,300 0,600 0,600 1,000 0,375 0,429 0,300
0,889 0,800 1,600 1,600 2,667 1,000 1,143 0,800
0,778 0,700 1,400 1,400 2,333 0,875 1,000 0,700
1,111 1,000 2,000 2,000 3,333 1,250 1,429 1,000

ВКА-2

$$W_{C_2} = (w_{1C_2}, \ldots, w_{8C_2}) =$$

$$= (0{,}158;\ 0{,}175;\ 0{,}088;\ 0{,}088;\ 0{,}053;\ 0{,}140;\ 0{,}123;\ 0{,}175).$$

МПС 3. Уровень владения ПК

1,000 0,625 2,500 0,556 1,000 1,250 1,250 1,667
1,600 1,000 4,000 0,889 1,600 2,000 2,000 2,667
0,400 0,250 1,000 0,222 0,400 0,500 0,500 0,667
1,800 1,125 4,500 1,000 1,800 2,250 2,250 3,000
1,000 0,625 2,500 0,556 1,000 1,250 1,250 1,667
0,800 0,500 2,000 0,444 0,800 1,000 1,000 1,333
0,800 0,500 2,000 0,444 0,800 1,000 1,000 1,333
0,600 0,375 1,500 0,333 0,600 0,750 0,750 1,000

ВКА-3

$$W_{C_3} = (w_{1C_3}, \ldots, w_{8C_3}) =$$

$$= (0{,}125;\ 0{,}200;\ 0{,}050;\ 0{,}225;\ 0{,}125;\ 0{,}100;\ 0{,}100;\ 0{,}075).$$

Этап 2. Реализация корректной модификации AHP-C на основе операции попарного сравнения элементов каждого вектора ВКА-i, $i = 1, 2, 3$, с *последующей нормализацией в каждой паре.*

Таблица П2

Ранжированный список альтернатив по трем алгоритмам

Ранг	AHP-C	AHP	AHP-CS
1	2	2	2
2	1	1	1
3	6	6	6
4	8	8	8
5	7	4	7
6	4	7	4
7	3	3	3
8	5	5	5

Относительные весовые коэффициенты (ОВК) на основе ВКА-1:

(1,000;1,000) (0,158;0,175) (0,158;0,088) (0,158;0,088) (0,158;0,053) (0,158;0,140) (0,158;0,123) (0,158;0,175)
(0,175;0,158) (1,000;1,000) (0,175;0,088) (0,175;0,088) (0,175;0,053) (0,175;0,140) (0,175;0,123) (0,175;0,175)
(0,088;0,158) (0,088;0,175) (1,000;1,000) (0,088;0,088) (0,088;0,053) (0,088;0,140) (0,088;0,123) (0,088;0,175)
(0,088;0,158) (0,088;0,175) (0,088;0,088) (1,000;1,000) (0,088;0,053) (0,088;0,140) (0,088;0,123) (0,088;0,175)
(0,053;0,158) (0,053;0,175) (0,053;0,088) (0,053;0,088) (1,000;1,000) (0,053;0,140) (0,053;0,123) (0,053;0,175)
(0,140;0,158) (0,140;0,175) (0,140;0,088) (0,140;0,088) (0,140;0,053) (1,000;1,000) (0,140;0,123) (0,140;0,175)
(0,123;0,158) (0,123;0,175) (0,123;0,088) (0,123;0,088) (0,123;0,053) (0,123;0,140) (1,000;1,000) (0,123;0,175)
(0,175;0,158) (0,175;0,175) (0,175;0,088) (0,175;0,088) (0,175;0,053) (0,175;0,140) (0,175;0,123) (1,000;1,000)

Нормализованные ОВК-1 на основе ВКА-1:

(0,500;0,500) (0,474;0,526) (0,643;0,357) (0,643;0,357) (0,750;0,250) (0,529;0,471) (0,563;0,438) (0,474;0,526)
(0,526;0,474) (0,500;0,500) (0,667;0,333) (0,667;0,333) (0,769;0,231) (0,556;0,444) (0,588;0,412) (0,500;0,500)
(0,357;0,643) (0,333;0,667) (0,500;0,500) (0,500;0,500) (0,625;0,375) (0,385;0,615) (0,417;0,583) (0,333;0,667)
(0,357;0,643) (0,333;0,667) (0,500;0,500) (0,500;0,500) (0,625;0,375) (0,385;0,615) (0,417;0,583) (0,333;0,667)
(0,250;0,750) (0,231;0,769) (0,375;0,625) (0,375;0,625) (0,500;0,500) (0,273;0,727) (0,300;0,700) (0,231;0,769)
(0,471;0,529) (0,444;0,556) (0,615;0,385) (0,615;0,385) (0,727;0,273) (0,500;0,500) (0,533;0,467) (0,444;0,556)
(0,438;0,563) (0,412;0,588) (0,583;0,417) (0,583;0,417) (0,700;0,300) (0,467;0,533) (0,500;0,500) (0,412;0,588)
(0,526;0,474) (0,500;0,500) (0,667;0,333) (0,667;0,333) (0,769;0,231) (0,556;0,444) (0,588;0,412) (0,500;0,500)

Относительные весовые коэффициенты на основе ВКА-2:

(1,000;1,000) (0,125;0,200) (0,125;0,050) (0,125;0,225) (0,125;0,125) (0,125;0,100) (0,125;0,100) (0,125;0,075)
(0,200;0,125) (1,000;1,000) (0,200;0,050) (0,200;0,225) (0,200;0,125) (0,200;0,100) (0,200;0,100) (0,200;0,075)
(0,050;0,125) (0,050;0,200) (1,000;1,000) (0,050;0,225) (0,050;0,125) (0,050;0,100) (0,050;0,100) (0,050;0,075)
(0,225;0,125) (0,225;0,200) (0,225;0,050) (1,000;1,000) (0,225;0,125) (0,225;0,100) (0,225;0,100) (0,225;0,075)
(0,125;0,125) (0,125;0,200) (0,125;0,050) (0,125;0,225) (1,000;1,000) (0,125;0,100) (0,125;0,100) (0,125;0,075)
(0,100;0,125) (0,100;0,200) (0,100;0,050) (0,100;0,225) (0,100;0,125) (1,000;1,000) (0,100;0,100) (0,100;0,075)
(0,100;0,125) (0,100;0,200) (0,100;0,050) (0,100;0,225) (0,100;0,125) (0,100;0,100) (1,000;1,000) (0,100;0,075)
(0,075;0,125) (0,075;0,200) (0,075;0,050) (0,075;0,225) (0,075;0,125) (0,075;0,100) (0,075;0,100) (1,000;1,000)

Нормализованные ОВК-2 на основе ВКА-2:

(0,500;0,500) (0,385;0,615) (0,714;0,286) (0,357;0,643) (0,500;0,500) (0,556;0,444) (0,556;0,444) (0,625;0,375) (0,615;0,385) (0,500;0,500) (0,800;0,200) (0,471;0,529) (0,615;0,385) (0,667;0,333) (0,667;0,333) (0,727;0,273) (0,286;0,714) (0,200;0,800) (0,500;0,500) (0,182;0,818) (0,286;0,714) (0,333;0,667) (0,333;0,667) (0,400;0,600) (0,643;0,357) (0,529;0,471) (0,818;0,182) (0,500;0,500) (0,643;0,357) (0,692;0,308) (0,692;0,308) (0,750;0,250) (0,500;0,500) (0,385;0,615) (0,714;0,286) (0,357;0,643) (0,500;0,500) (0,556;0,444) (0,556;0,444) (0,625;0,375) (0,444;0,556) (0,333;0,667) (0,667;0,333) (0,308;0,692) (0,444;0,556) (0,500;0,500) (0,500;0,500) (0,571;0,429) (0,444;0,556) (0,333;0,667) (0,667;0,333) (0,308;0,692) (0,444;0,556) (0,500;0,500) (0,500;0,500) (0,571;0,429) (0,375;0,625) (0,273;0,727) (0,600;0,400) (0,250;0,750) (0,375;0,625) (0,429;0,571) (0,429;0,571) (0,500;0,500)

Относительные весовые коэффициенты на основе ВКА-3:

(1,000;1,000) (0,156;0,178) (0,156;0,200) (0,156;0,022) (0,156;0,044) (0,156;0,178) (0,156;0,067) (0,156;0,156) (0,178;0,156) (1,000;1,000) (0,178;0,200) (0,178;0,022) (0,178;0,044) (0,178;0,178) (0,178;0,067) (0,178;0,156) (0,200;0,156) (0,200;0,178) (1,000;1,000) (0,200;0,022) (0,200;0,044) (0,200;0,178) (0,200;0,067) (0,200;0,156) (0,022;0,156) (0,022;0,178) (0,022;0,200) (1,000;1,000) (0,022;0,044) (0,022;0,178) (0,022;0,067) (0,022;0,156) (0,044;0,156) (0,044;0,178) (0,044;0,200) (0,044;0,022) (1,000;1,000) (0,044;0,178) (0,044;0,067) (0,044;0,156) (0,178;0,156) (0,178;0,178) (0,178;0,200) (0,178;0,022) (0,178;0,044) (1,000;1,000) (0,178;0,067) (0,178;0,156) (0,067;0,156) (0,067;0,178) (0,067;0,200) (0,067;0,022) (0,067;0,044) (0,067;0,178) (1,000;1,000) (0,067;0,156) (0,156;0,156) (0,156;0,178) (0,156;0,200) (0,156;0,022) (0,156;0,044) (0,156;0,178) (0,156;0,067) (1,000;1,000)

Нормализованные ОВК-3 на основе ВКА-3:

(0,500;0,500) (0,467;0,533) (0,438;0,563) (0,875;0,125) (0,778;0,222) (0,467;0,533) (0,700;0,300) (0,500;0,500) (0,533;0,467) (0,500;0,500) (0,471;0,529) (0,889;0,111) (0,800;0,200) (0,500;0,500) (0,727;0,273) (0,533;0,467) (0,563;0,438) (0,529;0,471) (0,500;0,500) (0,900;0,100) (0,818;0,182) (0,529;0,471) (0,750;0,250) (0,563;0,438) (0,125;0,875) (0,111;0,889) (0,100;0,900) (0,500;0,500) (0,333;0,667) (0,111;0,889) (0,250;0,750) (0,125;0,875) (0,222;0,778) (0,200;0,800) (0,182;0,818) (0,667;0,333) (0,500;0,500) (0,200;0,800) (0,400;0,600) (0,222;0,778) (0,533;0,467) (0,500;0,500) (0,471;0,529) (0,889;0,111) (0,800;0,200) (0,500;0,500) (0,727;0,273) (0,533;0,467) (0,300;0,700) (0,273;0,727) (0,250;0,750) (0,750;0,250) (0,600;0,400) (0,273;0,727) (0,500;0,500) (0,300;0,700) (0,500;0,500) (0,467;0,533) (0,438;0,563) (0,875;0,125) (0,778;0,222) (0,467;0,533) (0,700;0,300) (0,500;0,500)

Этап 3. Итоговая реализация корректной модификации AHP-CS. Получение линейной свертки:

(0,500;0,500) (0,437;0,563) (0,630;0,370) (0,575;0,425) (0,656;0,344) (0,527;0,473) (0,587;0,413) (0,539;0,461) (0,563;0,437) (0,500;0,500) (0,681;0,319) (0,633;0,367) (0,714;0,286) (0,589;0,411) (0,647;0,353) (0,598;0,402) (0,370;0,630) (0,319;0,681) (0,500;0,500) (0,453;0,547) (0,528;0,472) (0,393;0,607) (0,450;0,550) (0,406;0,594) (0,425;0,575) (0,367;0,633) (0,547;0,453) (0,500;0,500) (0,574;0,426) (0,453;0,547) (0,494;0,506) (0,458;0,542) (0,344;0,656) (0,286;0,714) (0,472;0,528) (0,426;0,574) (0,500;0,500) (0,371;0,629) (0,422;0,578) (0,387;0,613) (0,473;0,527) (0,411;0,589) (0,607;0,393) (0,547;0,453) (0,629;0,371) (0,500;0,500) (0,559;0,441) (0,513;0,487) (0,413;0,587) (0,353;0,647) (0,550;0,450) (0,506;0,494) (0,578;0,422) (0,441;0,559) (0,500;0,500) (0,453;0,547) (0,461;0,539) (0,402;0,598) (0,594;0,406) (0,542;0,458) (0,613;0,387) (0,487;0,513) (0,547;0,453) (0,500;0,500)

S. I. Kolesnikova, D. Sc., skolesnikova@yandex.ru, **S. A. Karavanova,** Master's Student,
Saint-Petersburg State University of Aerospace Instrumentation, St. Petersburg, 190000, Russian Federation

Optimization of Multi-Criterial Selection Algorithm with a Dynamically Filled Large Set of Alternatives

We consider the problem of the correct ranking of a dynamically replenished large set of alternatives in multicriteria choice problems that use in the solution the previously obtained modified method for analyzing hierarchies, based on the operation of additive convolution of local priorities not on the obtained set of characteristics of paired comparison matrices (as in the classical method), but on a set of pairs the relative weights of the coordinates of the eigenvectors being compared with each other for each criterion and the subsequent operation of additive convolution according to the criteria and alternatives in each pair. In this version, the algorithm ensures that previously achieved preferences are preserved when adding new alternatives and, thereby, makes it possible to optimize when processing large volumes of dynamically changing data, which significantly expands the applicability of the popular algorithm.

Keywords: *big data, correct algorithm for ranking alternatives, optimization of the algorithm for ranking a large set of alternatives, performance indicators of the algorithm for pairwise comparisons; algorithmic complexity*

Acknowledgements: The reported study was funded by RFBR according to the research project № 20-08-00747

DOI: 10.17587/it.27.235-241

References

1. **Saaty L. T.** Decision making. The analytic hierarchy process, Moskow, Radio i svyaz, 1989, 311 p. (in Russian).

2. **Arrow K. J.** Collective choice and individual values, Moskow, Nacionalnyj issledovatelski universitet Vysshej shkoly ekonomiki, 2004, 204 p. (in Russian).

3. **Podinovskii V. V., Noghin V. D.** Pareto-optimal Decisions in Multicriteria Optimization Problems, Moscow, Nauka, 2007, 256 p. (in Russian).

4. **Noghin V. D.** Simplified version of the hierarchy analysis method based on nonlinear convolution of criteria, *Zhurnal Vychislitelnoj Matematiki i Matematicheskoj Fiziki*, 2004, vol. 44, no 7, pp. 1259—1268 (in Russian).

5. **Samohvalov U. Ya.** Group accounting for the relative superiority of alternatives in decision-making problems, *Kibernetika i Sistemny Analiz*, 2003, no. 6, pp. 141—145 (in Russian).

6. **Kolesnikova S. I.** Features of the application of linear convolution of criteria in the method of paired comparisons, *Informacionnye Tekhnologii*, 2003, no. 1, pp. 24—29 (in Russian).

7. Oracle documentation: Official site. URL: https://docs.oracle.com/en (date accessed: 09/08/2020).

8. **Laphore R.** Java data structures and algorithms, St. Petersburg, Piter, 2018, 704 p. (in Russian).

9. **Knut D. E.** The art of programming, trans. ed. V. T. Tertyshny (ch. 5) and I. V. Krasikov (ch. 6), Moskow, Williams, 2007, vol. 3, 832 p. (in Russian).

10. **Hoare C.** Quicksort, *The Computer Journal*, 1962, vol. 5, no. 1, pp. 10—16.

УДК 519.687.1 DOI: 10.17587/it.27.242-248

П. Е. Голосов, канд. техн. наук, декан факультета информационных технологий и анализа данных, e-mail: pgolosov@gmail.com,
Институт ЭМИТ, Российская академия народного хозяйства и государственной службы при Президенте РФ (РАНХиГС), Москва,
И. М. Гостев, д-р техн. наук, вед. науч. сотр., e-mail: igostev@gmail.com,
Институт проблем передачи информации им. А. А. Харкевича Российской академии наук (ИППИ РАН), Москва

Оптимизация распределения потока задач поиска хеш-решений при априорно заданной сложности решений

Рост числа вычислительно трудоемких задач в условиях развития цифровой экономики (в рамках внедрения блокчейн-решений, распределенных реестров и пр.) требует все больше вычислительных ресурсов. При этом пользователи для минимизации расходов стремятся перенести вычислительный процесс в облако, а владельцы облачных сервисов вынуждены искать решения для повышения эффективности их использования. В работе рассматриваются подходы, позволяющие рассмотреть возможности оптимизации использования параллельных вычислительных ресурсов для поступающих наборов ресурсоемких задач, анализируются различные подходы к стратегии назначения задач на вычислительные ресурсы. Представлены результаты модельных экспериментов, учитывающих распределение заданий, параметризованных предельным временем выполнения в рамках моделирования исполнения соглашения с пользователем об уровне сервиса.

Ключевые слова: *оптимизация, параллельные вычисления, планирование вычислений, оценка эффективности*

Введение

В последнее время активно развивается тенденция переноса сложных и трудоемких вычислений с аппаратуры пользователя на облачные сервисы [1—3]. Это обусловлено тем, что пользователи стремятся сократить расходы на содержание сложной вычислительной системы и переложить их на специализированные сервисы облачных операторов. При этом некоторые виды вычислительных процессов требуют выделения существенных аппаратно-программных ресурсов. Одной из наиболее известных задач такого типа являются задача майнинга [4] при генерации различных криптовалют. При ее решении основные ресурсы отводятся на вычисление хеш-функций методами, основанными на простом переборе. Трудоемкость и время вычислений возрастает с увеличением длины хеша. Кроме задач майнинга в последнее время технологии блокчейна стали активно применять в распределенных базах данных, реестрах, кадастрах и т. п. Все эти задачи фактически решаются методами перебора, а характерным признаком таких задач является непредсказуемое время получения решения. Для обслуживания таких задач владельцы облачных сервисов стремятся минимизировать свои расходы посредством повышения эффективности использования своих вычислительных ресурсов. Однако в связи с необходимостью применения процедуры подтверждения работы при внедрении блокчейн-решений, в которых основное время занимает поиск нужного хеш-решения [5], эффективность функционирования сложных, гетерогенных вычислительных облачных систем постоянно снижается [6, 7]. Такое явление обусловлено тем фактом, что время, за которое необходимо получить решение, является случайной величиной, распределенной по равномерному закону (время, затрачиваемое на простой перебор решений). При этом специализированные планировщики различного уровня, позволяющие учитывать задачи такого типа, отсутствуют.

Все это приводит к необходимости рассмотрения ситуаций, при которых некоторый поток задач, который в дальнейшем будем называть *пакетом*, поступает в распределенную систему. Далее задачи, как правило, разделяются на несколько одинаковых частей, каждая из которых будет выполняться на своем вычислительном устройстве (процессоре). При окончании вычислений на одном процессоре (получении значения хеша требуемой сложности) все части задачи прекращают работу, а занятые ресурсы освобождаются.

Итак, основная цель настоящей работы заключается в моделировании функционирования

распределенной вычислительной системы для отыскания значений хеш-функций (решений) с заданными свойствами сложности (далее — операция), адресуемого пользователем специализированному облачному сервису. Результатом моделирования будут верхние оценки директивных сроков выполнения. Основным требуемым условием здесь является возможность представления задачи в виде большого числа малых вычислительных подзадач, называемых элементарными заданиями, каждое из которых оперирует с подмножеством исходного множества.

1. Постановка задачи

Рассмотрим задачу поиска хеш-функций как задачу перебора чисел с поиском заданных свойств. Примером такой операции служит подбор одноразовых чисел, например, для сети Биткоин [8], где число вариантов для поля ***nonce*** составляет 2^{32}. В силу свойства ее аддитивности разбиение пространства подстановок значений в функцию таково, что каждое элементарное задание независимо от остальных и все элементарные задания могут выполняться одновременно (параллельно) на различных вычислительных устройствах параллельной вычислительной системы (сети). Будем считать, что доступ к таким вычислителям предоставляется облачным сервисом[1]. В качестве искомого решения будем принимать первый отвечающий входным ограничениям результат (например, отыскание хеша заданной сложности, такого как SHA-256 [9, 10]), полученный на любой из параллельных ветвей исполнения операции для элементарных заданий. Будем понимать, что каждая из таких задач при выделении на ее решение всего ресурса системы решается за случайное время с равномерном законом распределения [11]. В работах [12, 13] показано, что процесс решения представляет собой процесс восстановления. Далее будем рассматривать различные варианты планирования процесса вычислений с учетом распараллеливания пакетов на вычислительной системе с распределенными ресурсами. При этом будем считать, что процесс решения будет происходить в распределенной, параллельной, гомогенной среде.

[1]В данной работе не учитываются времена: пересылки заданий от пользователя к вычислителю, загрузки и освобождения процессоров и ресурсов, необходимых для выполнения заданий.

2. Определения и процесс моделирования

Пусть в массово-параллельную вычислительную систему [14-16], обладающую единственным типом вычислительных устройств (например, GPU [17], далее процессор), на обработку поступает пакет из $m \geqslant 2$ задач.

Предположим, что система может одновременно обрабатывать как одно, так и несколько заданий на всех работоспособных в данный момент процессорах. В свою очередь отдельное задание может быть разделено на подзадания, каждое из которых может выполняться независимо от других на отдельном процессоре. При этом на каждом процессоре в единицу времени может обрабатываться только одно задание или его часть. Также будем считать, что задания поступают от пользователей одновременно в виде некоторого пакета, без каких-либо признаков или отношений предшествования между ними. Каждое задание из пакета может быть поставлено на обработку в любое определенное некоторым диспетчером время.

При полном владении ресурсом системы i-я задача решается за случайное время X_i, $i = \overline{1,m}$. Пусть функции распределения случайных величин X_i принадлежат множеству $\Im = \{F_1, \ldots, F_m\}$. Не ограничивая общности, полагаем, что X_i принимает значения на отрезке [0,1], т. е. $F(0) = 0$ и $F_i(x) = 1$ при $x \geqslant 1$, а для всех X_i, $i = \overline{1,m}$, существует конечный первый момент. ***Планом*** выполнения пакета задач Z будем называть набор из m векторов, $r_i = (r_1^{(i)}, \ldots, r_m^{(i)})$, $i = \overline{1,m}$, в котором $r_\upsilon^{(1)}$, $\upsilon = \overline{1,m}$, — это доля ресурса, выделяемая υ-й задаче при запуске пакета Z на обработку, а $r_\upsilon^{(i+1)}$, $\upsilon = \overline{1,m}$, — доля ресурса, выделяемая υ-й задаче после того как завершено решение i задач, $i = \overline{1,m-1}$. Для любого плана $z_i \in Z$, $i = \overline{1,m}$, и $x > 0$ обозначим $t_z(x)$ — условное математическое ожидание времени пребывания в системе k-й задачи при условии, что ее длина есть x: $t_Z(x) = M(t_{x_i}^Z | X_i = x)$, а момент времени пребывания i-й задачи в системе обозначим T_i.

Регулярные планы выполнения пакета задач определим следующим образом. Зафиксируем натуральное число $k : 1 \leqslant k \leqslant m$. Из имеющегося пакета выберем случайным образом k задач, разделим ресурс на k частей и начнем обрабатывать отобранные задачи одновременно. После окончания решения одной или нескольких задач освободившийся ресурс разделим поровну на все незавершенные задачи и продолжим их обработку. Будем продолжать процесс до тех пор, пока все k задач не будут

решены. После этого описанная процедура повторяется до тех пор, пока общий список имеющихся задач не будет исчерпан [18].

Как отмечено ранее, ресурс вычислительной системы и любая его часть кратны m. Аналогично предполагается делимость рассматриваемых задач на произвольное число независимых вычислительных ветвей.

Введем в рассмотрение два граничных плана при $k = 1$ и $k = m$, обозначив их как **1-*регулярный*** и ***m*-регулярный**, соответственно. При этом 1-*регулярный* план будем называть последовательным, а *m*-*регулярный* — параллельным. Очевидно, что общее время решения всех задач пакета не зависит от плана и равно $X_1 + X_2 + ... + X_m$.

Обозначим X вектор $(X_1, ..., X_m)$, где X_i — время решения i-го задания — случайная величина, имеющая функцию распределения $F(x)$. Упорядочим его компоненты по возрастанию и перенумеруем. Полученный вариационный ряд обозначим $X^* = (X_1^*, ..., X_m^*)$, для него: $X_1^* + X_2^* + ... + X_m^* = X_1 + X_2 + ... + X_m$.

Введем следующие обозначения:

$\overline{S}_m = \frac{1}{m}(X_1 + ... + X_m)$ — среднее время решения задач из множества Z; T_{seq} — среднее время пребывания задач в системе при последовательном плане; T_{par} — среднее время пребывания задач в системе при параллельном плане.

Утверждение 1. Для параллельного плана среднее время пребывания задач в системе определяется следующим равенством:

$$T_{par} = (2m-1)\overline{S}_m - \frac{2}{m}\sum_{k=1}^{m}(k-1)X_k^*.$$

Доказательство. Поскольку при делении ресурса на k частей время решения задач увеличивается в k раз, и освободившийся ресурс делится поровну, справедливы равенства:

$$T_1 = mX_1^*;\ T_2 = mX_1^* + (m-1)(X_2^* - X_1^*);$$
$$T_k - T_{k-1} = (m-k+1)(X_k^* - X_{k-1}^*);$$
$$T_k = X_1^* + ... + X_k^* + (m-k)X_k^*,\ k = \overline{2,m}.$$

После выполнения преобразований получаем:

$$T_{par} = \frac{1}{m}\sum_{k=1}^{m}[(X_1^* + ... + X_k^*) + (m-k)X_k^*] =$$
$$= \frac{1}{m}\left(\sum_{k=1}^{m}(m-k+1)X_k^* + (m-k)X_k^*\right) =$$
$$= \frac{2m-1}{m}\sum_{k=1}^{m}X_k^* - \frac{2}{m}\sum_{k=1}^{m}(k-1)X_k^* =$$
$$= \frac{(2m-1)}{m}\sum_{k=1}^{m}X_k - \frac{2}{m}\sum_{k=1}^{m}(k-1)X_k^*.$$

Утверждение доказано.

Замечание. Пусть время выполнения каждого задания — случайная величина, равномерно распределенная на отрезке [0,1]. Поскольку X_k^* — случайная величина, значение которой ограничено конечным интервалом, поэтому X_k^* имеет бета-распределение [19] с параметрами $(k,\ m - k + 1)$. Тогда для математического ожидания X_k^* справедливо равенство

$$MX_k^* = \frac{k}{m+1}.$$

Отсюда следует, что при использовании параллельного плана математическое ожидание времени пребывания k-го задания в системе равно

$$MT_{par}^w = \frac{k(2m-k+1)}{2(m+1)}.$$

При этом математическое ожидание времени задержки T_{del} между выполнением k-го и $(k-1)$-го заданий линейно убывает с ростом k и равно

$$MT_{del} = \frac{m-k+1}{m+1}.$$

Лемма 1. Пусть случайные величины X_i, $i = \overline{1,m}$, независимы, имеют функцию распределения $F(x)$ и имеют конечный первый момент. Тогда справедливо равенство

$$S(F) = M\sum_{k=1}^{m}(k-1)X_k^* = \binom{m}{2}M\max(X_1, X_2).$$

Доказательство. Поскольку функция распределения наибольшей порядковой статистики X_n имеет вид $F_n(x) = P\{X_n \leqslant x\} = P^n(x)$ [20], откуда

$$F_1(x) = P\{X_1 \leqslant x\} = 1 - P(X_1 > x) = 1 - [1 - P(x)]^n$$

и в более общем виде:

$$P\{X_k^* \leqslant x\} = \sum_{r=k}^{m}\binom{m}{r}F^r(x)(1-F(x))^{m-r},$$

то отсюда следует, что

$$\sum_{k=1}^{m}(k-1)MX_k^* =$$
$$= \int_0^1 x d\left(\sum_{k=1}^{m}(k-1)\sum_{r=k}^{m}\binom{m}{r}F^r(x)\left(1-F^{m-r}(x)\right)\right) =$$
$$= \int_0^1 x d\left(\sum_{r=1}^{m}\binom{m}{r}F^r(x)(1-F(x))^{m-r}\sum_{k=1}^{r}(k-1)\right) =$$
$$= \binom{m}{2}\int_0^1 x d\left(F^2(x)\sum_{r=0}^{m-2}\binom{m-2}{r}F^r(x)(1-F(x))^{m-r}\right) =$$
$$= \binom{m}{2}\int_0^1 x d(F^2(x)) = \binom{m}{2}M\max(X_1, X_2).$$

Последнее равенство верно, поскольку $F^2(x)$ является функцией распределения случайной величины $\max(X_1, X_2)$.

Утверждение 2. При применении последовательного плана условное математическое ожидание среднего времени пребывания задачи в системе относительно σ-алгебры, порожденной случайными величинами X_1, ..., X_m, равно

$$M(T_{seq}\,|X_1,\ldots,X_m) = \frac{m+1}{2}\overline{S}_m.$$

Доказательство. Если задачи из пакета Z выполняются последовательно в порядке i_1, ..., i_m, то время пребывания k-й задачи в системе равно $T_k = X_{i_1} + \ldots + X_{i_k}$, а среднее время прохождения задачи в системе равно

$$T_{seq} = \frac{1}{m}\sum_{k=1}^{m} T_k = \frac{1}{m}\sum_{k=1}^{m}(m-k+1)X_{i_k}.$$

Вероятность выбора каждой из последовательностей i_1, ..., i_m равна $(m!)^{-1}$. Следовательно,

$$M(T_{seq}\,|X_1,\ldots,X_m) =$$

$$= \frac{1}{m}\sum_{k=1}^{m}(m-k+1)\frac{(m-1)!}{m!}(X_1+\ldots+X_m) =$$

$$= \frac{1}{m^2}(X_1+\ldots+X_m)\sum_{k=1}^{m}(m-k+1) = \frac{m+1}{2}\overline{S}_m.$$

Доказательство завершено.

Теорема 1. Пусть случайные величины X_1, ..., X_m независимы, и множество F состоит из двух функций: $F_{m-l+1}(x) = \ldots = F_m(x) = G(x) = 0$; для $x < 1$; $l = \overline{0,m}$, и

$$F_1 = \ldots = F_{m-l}(x) = F(x).$$

Тогда для математических ожиданий среднего времени пребывания задачи в системе справедливы равенства

$$MT_{seq} = \frac{m+1}{2m}(l + (m-l)\mathrm{M}X_1);$$

$$MT_{par} = \frac{l^2}{m} + \frac{(m-l)}{m}\times$$

$$\times\{(2m-1)MX_1 - (m-l-1)M\max(X_1,X_2)\};$$

$$\Delta = M(T_{par} - T_{seq}) =$$

$$= \frac{l}{m}\left(l - \frac{m+1}{2}\right) + \frac{(m-l)}{m}\times$$

$$\times\left\{\frac{3}{2}(m-1)MX_1 - (m-l-1)M\max(X_1,X_2)\right\}.$$

Доказательство следует из утверждений 1 и 2 и теоремы. Кроме того, поскольку $X^*_{m-l+1} = \ldots = X^*_m = 1$ и $\overline{S}_m = \frac{1}{m}(l + (m-l)MX_1)$, то

$$MT_{par} = \frac{2m-1}{m}(l + (m-l)MX_1) -$$

$$-\frac{2}{m}\left\{\sum_{k=m-l+1}^{m}(k-1) + \sum_{k=1}^{m-l}(k-1)MX^*_k\right\}.$$

Замечание 1. Случай $l = 0$ в теореме 1 характерен для естественного протекания процесса обработки. Случай $l > 0$ связан с различными нарушениями в процессе постановки задач и надежностью вычислительных средств.

Замечание 2. Пусть случайные величины X_1, ..., X_m независимы в совокупности. Какие-либо l, $l = \overline{0,m}$, из них принимают значение "1" с вероятностью $p = 1$, а остальные равномерно распределены на отрезке [0, 1].

Тогда:

$$MX_1 = \frac{1}{2},\ M\max(X_1,X_2) = \frac{2}{3},$$

$$\Delta = \frac{(m-1)(m+l)}{12m} + \frac{l(l-1)}{3m} \geqslant 0.$$

Это означает, что последовательный план совпадает с параллельным только при $m = 1$. В частности, при $l = 0$, $\Delta = (m-1)/12$, а при $l = m$: $\Delta = (m-1)/2$.

Замечание 3. Для произвольной функции распределения $F(x)$ и $l = 0$ можно записать:

$$T_{seq} = \frac{m+1}{2}MX_1;$$

$$T_{par} = (2m-1)MX_1 - (m-1)M\max(X_1,X_2);$$

$$\Delta = (m-1)\left(\frac{3}{2}MX_1 - M\max(X_1,X_2)\right).$$

Последнее означает, что качество плана существенным образом зависит от такого параметра распределения $F(x)$, как $\left(\frac{3}{2}MX_1 - M\max(X_1,X_2)\right)$.

Следует обратить внимание, что полученные аналитические результаты существенным образом используют гипотезу о симметричной неопределенности, которая в реальности выполняется далеко не всегда, если вообще справедлива.

3. Обсуждение и анализ результатов моделирования

Для исследования предложенной модели были проведены следующие вычислительные эксперименты. Набор заданий, удовлетворяющий определенным свойствам, генерировался случайным образом. Далее в первом случае все задания пакета выполнялись одно за другим, при этом каждой из задач последовательно от-

давался весь имеющийся в наличии вычислительный ресурс. Во втором случае на выполнение был поставлен весь пакет одновременно, т. е. имеющиеся задания выполнялись параллельно. При этом мощность всех процессоров первоначально была разделена на все задачи, а освободившиеся ресурсы (процессоры) делились между оставшимися в зависимости от выбранной стратегии. Пакеты формировались как из задач со случайным временем окончания решения, равномерно распределенным на определенном временном интервале, так и из задач с фиксированным временем счета. Целью эксперимента являлось сравнение результатов обработки системой пакетов задач, обладающих различными свойствами, при параллельной и последовательной дисциплинах обслуживания.

На основе анализа результатов рассмотренных моделей относительно выбора планов решения необходимо обратить внимание на некоторые практически значимые аспекты.

Среднее время пребывания заданий в системе является лишь интегральной характеристикой случайного процесса

$$\mu(t) = \sum_{i=1}^{m} I(T_i \leqslant t),$$

где $I(A)$ — индикаторная функция [21] события A, т. е. $I(a) = 1$, если $a \in A$ и $I(a) = 0$ в противном случае.

В случае последовательного плана решения задач поведение процесса $\mu(t)$ совпадает с поведением процесса $\nu(t)$, определенного следующим образом. Если при $n \to \infty$ функции

$$F_n(x) = 0, \quad 0 \leqslant x < \frac{1}{n},$$

$$F_n(x) = \sum_{\nu \leqslant nx} p_\nu^{(n)}, \quad \frac{1}{n} \leqslant x \leqslant 1,$$

сходятся на отрезке [0, 1] равномерно к непрерывной функции $F(x)$, тогда конечномерные распределения процесса $\nu_n(t)$ слабо сходятся к конечномерным распределениям процесса восстановления

$$\nu(t) = \max\{k : X_0 + \ldots + X_k \leqslant t\}, \; t \geqslant 0.$$

Для случая параллельного плана при $m > 1$ требуется исследование поведения уже не процесса восстановления, а считающего процесса [22], что связано со значительными теоретическими и вычислительными сложностями. Тем не менее, в целях исследования поведения системы при решении пакетов задач было проведено статистическое моделирование, условия и основные результаты которого приведены ниже [23].

Считалось, что в систему одновременно поступает пакет из m задач, каждая из которых определяется случайной выборкой без возвращений. Предполагалось, что каждая такая задача имеет решение. Максимально возможное время решения такой задачи $\overline{\tau_i}$, $i = \overline{1,m}$, для каждой группы экспериментов определялось отдельно, при этом длительность решения каждой задачи τ_i полагалась либо случайной величиной, равномерно распределенной на отрезке $[0; \overline{\tau_i}]$, либо была зафиксирована и равна $\overline{\tau_i}$. Кроме того, для каждой задачи назначался директивный срок окончания ее выполнения d_i, $i = \overline{1,m}$.

Были рассмотрены четыре стратегии решения пакета задач.

- ***Последовательное выполнение***. В каждый момент времени системой решается ровно одна задача, при этом весь доступный ресурс отдается этой задаче. По окончанию работы предыдущего процесса начинается выполнение последующего. Задачи решаются в случайном порядке.
- ***Последовательное выполнение с упорядочиванием*** по максимально возможному времени решения задачи $\overline{\tau_i}$. Имеющийся пакет задач упорядочивается по возрастанию $\overline{\tau_i}$ и перенумеровывается, на первое место помещается априорно самая "короткая" задача, на последнее — самая "объемная".
- ***Параллельное выполнение*** с выделением приоритетных задач. Это означает, что все задачи начинают выполняться одновременно, при этом ресурс между ними распределяется пропорционально отношению $\overline{\tau_i}/d_i$. После завершения какой-либо задачи освободившийся ресурс распределяется между оставшимися также пропорционально отношению $\overline{\tau_i}/d_i$.
- ***Параллельное выполнение с предпочтением коротких задач*** и упорядочиванием по максимально возможному времени решения задачи $\overline{\tau_i}$. Это означает, что все задачи начинают решаться одновременно, при этом первоначально ресурс между ними распределяется пропорционально величине $\overline{\tau_i}/d_i$. После завершения очередной задачи весь освободившийся ресурс отдается той задаче, у которой меньше максимально возможное время счета (наименьшая сложность).

Для сравнения параллельного и последовательного планов решения задач был проведен ряд модельных экспериментов в целях выявления особенностей процедуры их прохождения. Для каждой модели генерировалось 10^6 случайных реализаций наборов данных, вы-

Число решенных задач при различных планах обслуживания

числялись выборочные средние и среднеквадратичные отклонения исследуемых величин. Результаты счета сравнивались по следующим показателям: среднее время решения i-й задачи, средняя разность моментов завершения i-й и $(i-1)$-й задач, их среднеквадратичное отклонение и другие параметры.

На рисунке приведены результаты моделирования процесса обработки пакетов объемом 12 задач, максимальное время решения каждой из которых $\overline{\tau_i}$ являлось равномерно распределенной случайной величиной, нормированной следующим образом:

$$\overline{\tau_i} = k\alpha_i,$$

где α_i — случайная величина, равномерно распределенная на отрезке [0,1]; $k = \dfrac{12}{\sum\limits_{i=1}^{12} \alpha_i}$ — нормирующий коэффициент.

Директивный срок окончания решения для всех задач одинаков и составляет 12 единиц времени. Время решения каждой задачи — случайная величина, равномерно распределенная на интервале $[0, \overline{\tau_i}]$.

Результаты экспериментов полностью подтвердили теоретические положения, сформулированные выше. В зависимости от выбранной общей стратегии решения пакетов ресурсоемких задач путем группирования и упорядочивания их по некоторым априорным или статистическим характеристикам имеется реальная возможность согласовывать график решения с директивными сроками выполнения, обеспечив условия соглашения "потребитель—поставщик".

Заключение

Полученные результаты позволяют построить некоторую стратегию управления ресурсами распределенной параллельной вычислительной системы, на основе которой в дальнейшем будут разрабатываться планировщики различного уровня, обеспечивающие максимальную эффективность таких систем. Вопросы экономических оценок эффективности вычислений также будут рассмотрены в отдельной статье.

Список литературы

1. **Armbrust M., Fox A., Griffith R.** et al. Above the Clouds: A Berkeley View of Cloud Computing. UCBerkeley Reliable Adaptive Distributed Systems Laboratory, 2009.
2. **Toby Velte, Anthony Velte, Robert C. Elsenpeter.** Cloud Computing, A Practical Approach. McGraw Hill Professional, 2009. 400 p.
3. **Мирин С., Башилов Г., Патрикеев Д.** Cloud providing 2018—2022: economy, strategies, business-models. iKS-Consulting, 2018. 180 с.
4. **Rosenfeld M.** Analysis of Bitcoin Pooled Mining Reward Systems // CoRR: Computing Research Repository abs/1112.4980, arXiv:1112.4980 (2011).
5. **Carter J. L., Wegman M. N.** Universal classes of hash functions // J. Comp. Sys. Sci. 1979. Vol. 18. P. 143—154.
6. **Narayanan A., Bonneau J.** et al. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction Hardcover Princeton University Press, 2016. 336 p.
7. **Cohen R.** Global Bitcoin Computing Power Now 256 Times Faster Than Top 500 Supercomputers Combined // Forbes. 2013.
8. **Nakamoto S.** Bitcoin: //A Peer-to-Peer Electronic Cash System. 2008. URL: https://bitcoin.org/bitcoin.pdf (дата обращения: 08.07.2020).
9. **Kraft D.** Difficulty control for blockchain-based consensus systems // Peer-to-Peer Networking and Applications. 2015. P. 1—17.
10. **Frankenfield J.** Cryptocurrency Difficulty. 2020 [Электронный ресурс]. URL: https://www.investopedia.com/terms/d/difficulty-cryptocurrencies.asp. Электронный ресурс (дата обращения: 08.07.2020).
11. **Корн Г., Корн Т.** Справочник по математике для научных работников и инженеров. М.: Наука, 1970.
12. **Кокс Д. Р., Смит В. Л.** Теория восстановления. Пер. с англ. М.: Сов. радио, 1967.
13. **Математическая** энциклопедия. Т. 1 (А — Г). М.: Советская Энциклопедия, 1977. 1152 с.
14. **Таненбаум Э., Стен М.** Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.
15. **Almasi G. S., Gottlieb A.** Highly Parallel Computing. Benjamin-Cummings publishers, Redwood City, CA, 1989.
16. **Корнеев В. В.** Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985.
17. **Орлов С.** Гетерогенные вычисления и новые серверные платформы // ИнформКурьер-Связь. 2019. № 3. С. 44—50.
18. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002.
19. **Королюк В. С., Портенко Н. И.,Скороход А. В., Турбин А. Ф.** Справочник по теории вероятностей и математической статистике. М.: Наука, 1985. 640 с.
20. **Дэйвид Г.** Порядковые статистики. М.: Наука, 1979.
21. **Кормен Т., Лейзерсон Ч., Ривест Р.** Алгоритмы: Построение и анализ М.: МЦНМО, 2001.
22. **Кабанов Ю. М., Липцер Р. Ш., Ширяев А. Н.** Слабая и сильная сходимость распределений считающих процессов // Теория вероятности и ее применения. 1983. Т. XXVIII, № 2. С. 288—319.
23. **Голосов П. Е., Козлов М. В., Малашенко Ю. Е.** и др. Модель системы управления специализированным вычислительным комплексом. М.: ВЦ РАН, 2010. 48 с.

P. E. Golosov, Cand. of Tech. Sc. (Ph.D.), Dean of the Faculty of Information Technologies and Data Analysis, e-mail: golosov-pe@ranepa.ru,
Institute of EMIT of the Russian Academy of National Economy and Public Administration under the Russian President (RANEPA),
Moscow, Russian Federation, 119571, Moscow,
I. M. Gostev, D. Sc., e-mail: igostev@gmail.com,
Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Moscow, 127051, Russian Federation

Optimization of the Distribution of Hash Calculation Tasks Flow at a Priori Given Complexity

The increasing number of computationally intensive tasks induced by digital economy development (within the framework of implementing block-chain solutions, distributed ledgers, etc.) requires more and more computational resources. At the same time users tend to move the computational process to the cloud in order to minimize costs, and the owners of cloud services are forced to look for solutions to improve their efficiency. In this paper we consider approaches that allow optimize the use of parallel computing resources for incoming sets of resource-intensive tasks, analyze different approaches to the strategy of assigning tasks to computing resources. The results of modelling experiments are provided taking into account task distribution, parameterized by execution time limit within modeling the service level agreement with the user.

Keywords: *optimization, parallel computations, computation planning, efficiency estimation*

DOI: 10.17587/it.27.242-248

References

1. **Armbrust M., Fox A., Griffith R. et al.** Above the Clouds: A Berkeley View of Cloud Computing, UCBerkeley Reliable Adaptive Distributed Systems Laboratory, 2009.
2. **Velte T., Velte A., Elsenpeter R. C.** Cloud Computing, A Practical Approach by McGraw Hill Professional, 2009, 400 p.
3. **Mirin S., Bashilov G., Patrykeev D.** Cloud providing 2018—2022: economy, strategies, business-models, *iKS-Consulting*, 2018, 180 p. (in Russian).
4. **Rosenfeld M.** Analysis of Bitcoin Pooled Mining Reward Systems. CoRR: Computing Research Repository abs/1112.4980, arXiv:1112.4980 (2011).
5. **Carter J. L., Wegman M. N.** Universal classes of hash functions, *J. Comp. Sys. Sci.,* 1979, vol. 18, pp. 143—154.
6. **Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S.** Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction Hardcover, Princeton University Press, 2016, 336 p.
7. **Cohen R.** Global Bitcoin Computing Power Now 256 Times Faster Than Top 500 Supercomputers Combined, *In Forbes, 2013.*
8. **Nakamoto S.** Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, available at: https://bitcoin.org/bitcoin.pdf (circulation date: 08.07.2020).
9. **Kraft D.** Difficulty control for blockchain-based systems, *Peer-to-Peer Networking and Applications*, 2015, pp. 1—17.
10. **Frankenfield J.** Cryptocurrency Difficulty. 2020, available at: https://www.investopedia.com/terms/d/difficulty-cryptocurrencies.asp. Electronic resource (date of access: 08.07.2020).
11. **Korn G., Korn T.** Handbook of Mathematics for Researchers and Engineers, Moscow, Science, 1970 (in Russian).
12. **Cox D. R., Smith V. L.** Theory of Restoration, Per. English, Moscow, 1967 (in Russian).
13. **Vinogradov I. M.** et al. Mathematical Encyclopedia. P. 1 (A—G), Moscow, The Soviet Encyclopedia, 1977, 1152 p. (in Russian).
14. **Tanenbaum E., Stan M.** Distributed systems. Principles and paradigms, SPb, 2003, 877 p. (in Russian).
15. **Almasi G. S., Gottlieb A.** Highly Parallel Computing. Benjamin-Cummings publishers, Redwood City, CA, 1989.
16. **Korneev V. V.** Architecture of Computing Systems with Programmable Structure, Novosibirsk. Nauka, 1985 (in Russian).
17. **Orlov S.** Heterogeneous computations and new server platforms, *InformCourier Communication,* 2019, no. 3, pp. 44—50 (in Russian).
18. **Voevodin V. V., Voevodin V. V.** Parallel computing, St. Petersburg, BHV Petersburg, 2002 (in Russian).
19. **Korolyuk V. S., Portenko N. I., Skorokhod A. V., Turbin A. F.** Handbook on Probability Theory and Mathematical Statistics, Moscow, Nauka, 1985, 640 p. (in Russian).
20. **David G.** Ordinary statistics, Moscow, Nauka, 1979 (in Russian).
21. **Korman T., Layserson C., Rivest R.** Algorithms: Construction and Analysis, Moscow, ICSEMO, 2001.
22. **Kabanov Yu. M., Liptser R. S., Shiryaev A. N.** Weak and strong convergence of distributions of counting processes, Theory of Probability and its Application, Moscow, 1983, vol. XXVIII, ch. 2, pp. 288—319 (in Russian).
23. **Golosov P. E., Kozlov M. V., Malashenko Yu.** Model of control system for specialized computing complex, Moscow, VZ RAN, 2010, 48 p. (in Russian).

УДК 519.8 DOI: 10.17587/it.27.249-258

Ю. А. Зак, д-р техн. наук, e-mail: yuriy_zack@hotmail.com,
Аахен, Германия

Многостадийные open-shop-problem: расписания выполнения *N* заданий на *K* различных по техническим характеристикам машинах при произвольном порядке выполнения заданий

Рассматриваются постановки и математические модели open-shop-problem: выполнение N заданий на K машинах при произвольном порядке обработки каждого из заданий на всех машинах. В отличие от большинства публикаций, посвященных решению данной проблемы, поставленная задача решается в многостадийной постановке в условиях работы последовательной цепочки участков и цехов предприятия, а также в условиях наличия ограничений на времена начала и завершения выполнения заданий и допустимые времена работы машин. Кроме того, для систем календарного планирования работы предприятий чрезвычайно актуальным является рассмотрение этой задачи в многостадийной постановке, т. е. в условиях работы последовательной цепочки участков и цехов. Исследуются свойства допустимых и оптимальных расписаний, а также алгоритмы точных и приближенных методов решения этих задач последовательными алгоритмами оптимизации. На ранних этапах решения задачи устанавливается факт несовместности системы ограничений задачи и определяется подмножество заданий или ресурсов, временной диапазон которых должен быть расширен. Описанные алгоритмы решения задачи иллюстрируются числовыми примерами.

Ключевые слова: *многостадийные расписания, open-shop-problem, ограничения на времена выполнения заданий, нижняя граница длины расписания, последовательные алгоритмы оптимизации, эвристический алгоритм*

Введение

Рассмотрим задачу построения расписаний выполнения N заданий на K машинах в условиях, когда каждое из заданий $i = 1, ..., N$ должно пройти обработку на каждой из $k = 1, ..., K$ машин. При этом допустим произвольный порядок обработки каждого из заданий на всех машинах. Эта задача известна в литературе как open-shop-problem. В отличие от данной задачи в задачах job-shop-problem для каждого задания определен строгий порядок выполнения операций и последовательность машин, на которых они выполняются [1—5, 7, 8, 10]. Эти два класса задач теории расписаний имеют много практических приложений в календарном планировании производства и организации технического и сервисного обслуживания объектов.

В литературе [1, 4, 7, 8, 10, 12—14] задачи open-shop-problem рассматривались, в основном, в постановке без учета ограничений на времена завершения выполнения заданий. В такой постановке заданы времена выполнения всех заданий на каждой машине t_{ik}, и эти времена не могут претерпевать разрывы во времени выполнения. Одновременно на каждой машине может выполняться только одно задание. Необходимо определить последовательности выполнения всех заданий на каждой машине, а также времена начала и завершения выполнения каждого из заданий на всех машинах. В качестве критериев оптимальности задачи рассматривались завершение выполнения заданий в кратчайшие сроки, а также минимизация суммарного средневзвешенного времени работы машин.

Рассматриваемые задачи относятся к классу NP-сложных проблем. По сравнению с большим числом публикаций по решению wan-machine-problem, flow-shop, job-shop-problem, а также задач построения расписаний выполнения работ на параллельных машинах, задачам open-shop-problem уделялось очень мало внимания в монографиях и периодической литературе. Публикации касались, в основном, решения частных задач для случаев, когда число машин $K = 2$ (см., например, работы [1, 7, 9, 11]), по критерию минимизации времени цикла (выполнение расписаний в кратчайшие сроки). В литературе не рассматривались постановки задач в условиях заданной системы ограничений на времена начала и завершения выполнения заданий и допустимые времена работы машин. Для практических приложений, когда перед предприятиями в первую очередь стоят задачи выполнения в установленные сроки принятых портфелей заказов, решение сформулированных проблем в такой постановке является чрезвычайно актуальным. В ряде

случаев такая постановка задачи важнее, чем оптимизация какого-либо из критериев эффективности.

В данной статье рассматриваются постановки и математические модели open-shop-problem в самой общей постановке (выполнение N заданий на K машинах) в условиях, когда заданы ограничения на времена начала и завершения выполнения заданий и допустимые времена работы машин. Кроме того, для систем календарного планирования работы предприятий чрезвычайно актуальным является рассмотрение этой задачи в многостадийной постановке, т.е. в условиях работы последовательной цепочки участков и цехов. В данной работе исследуются свойства допустимых и оптимальных расписаний, а также точные и приближенные методы решения таких задач последовательными алгоритмами оптимизации. На ранних этапах решения задачи устанавливается факт несовместности системы ограничений задачи и определение подмножества заданий или ресурсов, временной диапазон которых должен быть расширен. Предложенные в работе алгоритмы решения этих задач с достаточно высокой эффективностью могут быть использованы и для решения open-shop-problem при отсутствии каких-либо ограничений. Описанные алгоритмы решения задачи иллюстрируются числовыми примерами.

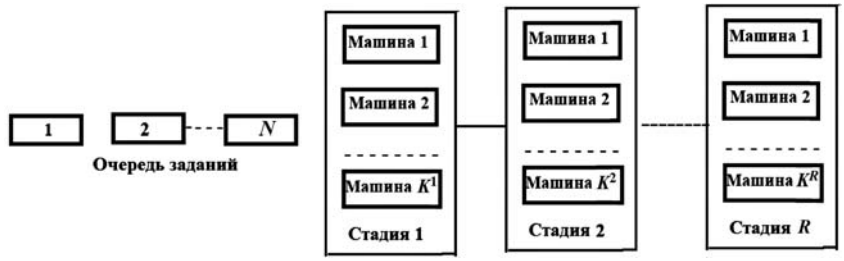


Схема производства многостадийных open-shop-problem

1. Постановка и математическая формулировка задачи

Пусть на K рабочих станциях (машинах), $k = 1, ..., K$, должно быть выполнено N заданий. i-е задание состоит из множества $\tilde{J}_i$ операций, каждая из которых выполняется на некоторой определенной машине. Порядок выполнения всех операций каждого из заданий может быть произвольным. В дальнейшем каждая операция (обозначим ее O_{ijk}, $i = 1, ..., N$, $j = 1, \ldots, \overline{J}_i$, $k = 1, ..., K$) однозначно определяется мультииндексом $a = (i, j, k)$. Времена выполнения операций суть целочисленные величины. Не допускается прерываний времен выполнения операций на машине. Одновременно на каждой машине не может выполняться более одной операции какого-либо задания. Каждая операция одного и того же задания выполняется на конкретной, определенной для нее технологией обработки машине. Схема производства многостадийных open-shop-problem представлена на рис. 1.

В случае многостадийных open-shop-problem приняты следующие обозначения переменных:

$r = 1, ..., R$ — индексы стадий производственного процесса;

$t^r(i, j, k)$ — время выполнения j-й операции i-го задания на k-й машине на r-й стадии обработки, $r = 1, ..., R$;

$x^r(i, j; k, l)$, $\theta^r(i, j; k, l) = [x^r(i, j; k, l), + t^r(i, j; k, l) - 1]$ — соответственно фактическое (установленное построенным расписанием выполнения работ) время начала и время завершения выполнения операции O_{ijk}, стоящей в очередности l на k-й машине на r-й стадии обработки, $r = 1, ..., R$;

$T^r(i)$ — время завершения выполнения i-го задания на r-й стадии обработки, $r = 1, ..., R$;

K^r — число машин на r-й стадии обработки;

$h^r(k)$, $H^r(k)$ — соответственно допустимые наиболее ранние сроки начала и наиболее поздние сроки окончания работы k-й машины, $k = 1, ..., K$, на r-й стадии обработки, $r = 1, ..., R$;

$d(i)$, $i = 1, ..., N$, — наиболее ранний срок начала выполнения i-го задания.

Значения возможных наиболее ранних и допустимых самых поздних времен начала выполнения всех этих операций на каждой k-й машине могут быть определены из следующих соотношений:

$$x^1(i,1;k,1) = \max[d(i), h^1(k)];$$
$$x^1(i,1;k,l) = \max[d(i), \{\theta^1(p,j;k,l-1)+1\}], \quad (1)$$
$$i,p = 1,\ldots,N;\ j = 1,\ldots,m_i;\ k,g = 1,\ldots,K;$$
$$l = 1,\ldots,L_k;$$

$$x^1(i,j;k,1) = \max[h^1(k), \{\theta^1(i,j-1;g,1)+1\}],$$
$$i = 1,\ldots,N;\ j = 1,\ldots,m_i; \quad (2)$$
$$k,g = 1,\ldots,K;\ l = 1,\ldots,L_k;$$

$$x^1(i,j;k,l) =$$
$$= \max[h^1(k), \theta^1(i,j-1;g,q), \theta^1(p,q;k,l-1)]+1, \quad (3)$$
$$i,p = 1,\ldots,N;\ j = 1,\ldots,m_i;\ k,g = 1,\ldots,N;\ l,q = 1,\ldots,L_k;$$

$$x^r(i,1;k,1) = \max[h^r(k),\{\theta^{r-1}(i,m_i;g,l)+1\}], \quad i=1,\dots,N;\ k,g=1,\dots,K;\ r=2,\dots,R;\ l=1,\dots,L_k; \tag{4}$$

$$x^r(i,1;k,l) = \max[\theta^{r-1}(i,m_i;g,q),\theta^r(p,j;k,l-1)]+1, \quad i=1,\dots,N;\ j=1,\dots,m_i;\ k,g=1,\dots,K;\ l,q=1,\dots,L_k;\ r=2,\dots,R; \tag{5}$$

$$x^r(i,j;k,1) = \max[h^r(k),\{\theta^r(i,j-1;g,l)+1\}], \quad i=1,\dots,N;\ j=1,\dots,m_i;\ k,g=1,\dots,K;\ l=1,\dots,L_k;\ r=2,\dots,R; \tag{6}$$

$$x^r(i,j;k,l) = \max[h^r(k),\theta^r(i,j-1;g,q),\theta^r(p,v;k,l-1)]+1, \quad i,p=1,\dots,N;\ j,v=1,\dots,m_i;\ k,g=1,\dots,K;\ l,q=1,\dots,L_k;\ r=2,\dots,R; \tag{7}$$

$$\theta^r(i,j;k,l) = x^r(i,j;k,l)+t(i,j;k)-1, \quad i=1,\dots,N;\ j=1,\dots,m_i;\ k=1,\dots,K;\ l=1,\dots,L_k;\ r=1,\dots,R. \tag{8}$$

Время завершения выполнения i-го задания на r-й стадии обработки определяется выражением

$$T^r(i) = \theta^r(i,m_i;k,l), \quad i=1,\dots,N, r=1,\dots,R \tag{9}$$

Время завершения выполнения всех операций на k-й машине на r-й стадии обработки равно

$$T^r(k) = \theta^r(i,j;k,L_k), k=1,\dots,K, r=1,\dots,R. \tag{10}$$

Допустимые времена выполнения всех операций каждого задания рассчитываются по формулам

$$T^R(i) = \theta^R(i,m_i;k,l) \leqslant D(i), i=1,\dots,N; \tag{11}$$

$$T^r(k) = \theta^r(i,j;k,L_k) \leqslant H(k), \quad k=1,\dots,K, r=1,\dots,R. \tag{12}$$

Время завершения выполнения всех заданий определяется выражением

$$F = \max_{1\leqslant i\leqslant N} T(i). \tag{13}$$

2. Нижняя граница оптимального решения задачи

Рассмотрим нижнюю границу значения критерия оптимальности для одностадийных open-shop-problem.

Если $t(i,j;k,l)$ — время выполнения j-й операции i-го задания на k-й машине, то время завершения выполнения расписания не может быть меньше следующего значения:

$$\vartheta(T) = \max\left[\max_{1\leqslant i\leqslant N}\sum_{j=1}^{m_i} t(i,j;k,l), \max_{1\leqslant k\leqslant K}\sum_{l=1}^{L_k} t(i,j;k,l)\right]. \tag{14}$$

Если справедливо $\vartheta(T) = \max\limits_{1\leqslant i\leqslant N}\sum\limits_{j=1}^{m_i} t(i,j;k,l)$ для некоторого индекса $k = k_1$, то в job-shop-problem к данной величине еще необходимо добавить минимальное суммарное время выполнения всех операций на других машинах какого-либо из заданий, т. е. величину

$$\beta_{k1} = \min_{1\leqslant i\leqslant n}\sum_{k=1}^{K}\{t(i,j;k,l) \mid k \neq k_1\}. \tag{15}$$

Если справедливо $\vartheta(T) = \max\limits_{1\leqslant k\leqslant K}\sum\limits_{l=1}^{L_k} t(i,j;k,l)$ для некоторого индекса $i = i_1$, то к данной величине еще необходимо добавить минимальное суммарное время выполнения всех операций на какой-либо машине одного из заданий, т.е. величину

$$\beta_{k2} = \min_{1\leqslant k\leqslant K}\sum_{j=1}^{m_i}\{t(i,j;k,l) \mid i \neq i_1\}. \tag{16}$$

Следовательно,

$$\vartheta(T) = \max\left\{[\max_{1\leqslant i\leqslant N}\sum_{j=1}^{m_i} t(i,j;k,l)+\beta_{k1}], \max_{1\leqslant k\leqslant K}[\sum_{l=1}^{L_k} t(i,j;k,l)+\beta_{k2}]\right\} \tag{17}$$

Пусть известны значения нижних границ длины выполнения всего комплекса работ на каждой стадии обработки $\vartheta(T^r)$, $r = 1, ..., R$, которые вычислены по формулам (14) или (15)—(17). Вычислим значения

$$B^r = \min_{1\leqslant i\leqslant N} B_i^r = \min_{1\leqslant i\leqslant N}\sum_{j=1}^{m_i} t^r(i,j;k), r=1,\dots,R. \tag{18}$$

Так как перед началом и после завершения выполнения всех заданий на каждой стадии обработки необходимо, как минимум, завершение выполнений хотя бы данного задания на всех других стадиях обработки, значение нижней границы выполнения расписания многостадийной обработки вычисляется по формуле

$$\vartheta(F) = \max_{1\leqslant r\leqslant R}\left[\vartheta(T^r) + \sum_{p=1}^{R}\{B^p \mid p \neq r\}\right]. \quad (19)$$

Если справедливо выражение $\vartheta(T) = \max_{1\leqslant k\leqslant K}\sum_{l=1}^{L_k} t(i,j;k,l)$, и оно достигается для некоторого индекса $i = i_1$, то к данной величине необходимо добавить минимальное суммарное время выполнения всех операций на какой-либо машине одного из заданий, т. е. величину

$$\beta_{k2} = \min_{1\leqslant k\leqslant K}\sum_{j=1}^{m_i}\{t(i,j;k,l) \mid i \neq i_1\}. \quad (20)$$

Следовательно,

$$\vartheta(T) = \max\left\{[\max_{1\leqslant i\leqslant N}\sum_{j=1}^{m_i} t(i,j;k,l) + \beta_{k1}],\ \max_{1\leqslant k\leqslant K}[\sum_{l=1}^{L_k} t(i,j;k,l) + \beta_{k2}]\right\}. \quad (21)$$

Пусть определены $\vartheta(T^r)$, $r = 1, ..., R$. Вычислим также величины

$$B^r = \min_{1\leqslant i\leqslant N} B_i^r = \min_{1\leqslant i\leqslant N}\sum_{j=1}^{m_i} t^r(i,j;k), r = 1,\ldots,R, \quad (22)$$

и другое, требующее меньшего объема вычислений выражение для определеня нижней границы длины расписания:

$$\vartheta(F) = \max_{1\leqslant r\leqslant R}\left[\vartheta(T^r) + \sum_{p=1}^{R}\{B^p \mid p \neq r\}\right]. \quad (23)$$

3. Свойства допустимых и оптимальных расписаний

Утверждение 1. Если справедливо хотя бы одно из системы неравенств

$$d(i) + \sum_{r=1}^{R}\sum_{j=1}^{m_i} t^r(i,j,k) > D(i), i = 1,\ldots,N;$$
$$h(k) + \sum_{r=1}^{R}\sum_{i=1}^{N} t^r(i,j,k) > H(k), k = 1,\ldots,K, \quad (24)$$

то сформулированная задача не содержит допустимых решений.

Пусть в процессе решения задачи определены некоторые значения $x^r(i,j;k,l)$, и $\theta^r(i,j;k,l)$. Обозначим $\tilde{J}_i^{1r}, \tilde{J}_i^{1r}$ — соответственно подмножество выполненных и подлежащих выполнению операций i-го задания на r-й стадии обработки; $\tilde{Y}_k^{1r}, \tilde{Y}_k^{2r}$ — соответственно подмножество выполненных и подлежащих выполнению операций на k-й машине на r-й стадии обработки.

Утверждение 2. Если справедливо хотя бы одно из следующей ниже системы неравенств:

$$\theta^r(i,j;k,l) + \sum_{j\in\tilde{J}_i^{2k}} t^r(i,j;k,l) + \sum_{g=r+1}^{R}\sum_{j=1}^{m_i} t^g(i,j;k,l) > D(i), i = 1,\ldots,N; \quad (25)$$

$$\theta^r(i,j;k,l) + \sum_{l\in\tilde{Y}_k^{2r}} t^r(i,j;k,l) + \sum_{g=r+1}^{R}\sum_{l=1}^{L_k} t^g(i,j;k,l) > H(k), k = 1,\ldots,K, \quad (26)$$

то строящееся расписание не содержит допустимых решений и может быть исключено из рассмотрения.

Определим значение нижней границы длины выполняемого расписания.

Вычислим следующие величины:

$$\delta_1(F) = \max_{1\leqslant i\leqslant N}\{\theta^r(i,j;k,l) + \sum_{j\in\tilde{J}_i^{2k}} t^r(i,j;k,l) + \sum_{g=r+1}^{R}\vartheta(T^r)\}; \quad (27)$$

$$\delta_2(F) = \max_{1\leqslant k\leqslant K}\{\theta^r(i,j;k,l) + \sum_{l\in\tilde{Y}_k^{2r}} t^r(i,j;k,l) + \sum_{g=r+1}^{R}\vartheta(T^r)\}. \quad (28)$$

Тогда значение нижней границы определяется по формуле

$$\vartheta(F) = \max[\delta_1(F), \delta_2(F)]. \quad (29)$$

4. Алгоритм построения допустимых решений задачи

В целом ряде практических приложений нахождение допустимого решения задачи является достаточно важным и связано с определенными трудностями. Поэтому в качестве главной цели построения расписания будем рассматривать построение последовательностей выполнения

операций всех заданий на всех стадиях обработки, обеспечивающих получение допустимого решения, удовлетворяющего всем ограничениям задачи. Эта задача также относится к классу *NP*-сложных проблем. Рассмотрим алгоритм приближенного решения этой задачи.

Алгоритм решения задачи состоит из некоторого числа итераций $s = 1,\ldots,S,$ на каждой из которых строится последовательность выполнения операций некоторых заданий на различных машинах определенных стадий обработки. Проводится расчет времен начала и завершения выполнения этих операций по формулам (1)—(8). Проверяется возможность выполнения системы ограничений, связанная с таким назначением, в соответствии с формулами (19)—(22).

Упорядочим все выполняемые задания в последовательность

$$\tilde{W} = \{i_1, i_2, \ldots, i_\gamma, i_{\gamma+1}, \ldots, i_N \mid D(i_\gamma) \leqslant D(i_{\gamma+1})\}. \quad (30)$$

Такая последовательность на *s*-м шаге итеративного процесса имеет вид

$$\tilde{W}^s = \{i_1^s, \ldots, i_\gamma^s, i_{\gamma+1}^s, \ldots, i_n^s \mid D(i_\gamma^s) \leqslant D(i_{\gamma+1}^s)\}, n \leqslant N. \quad (31)$$

Пусть $\tilde{I}_1^s, \tilde{I}_2^s$ — соответственно подмножества индексов заданий, для которых на *s*-м шаге алгоритма определены и еще не определены времена завершения их выполнения

$$\tilde{I}_1^s \cup \tilde{I}_2^s = \tilde{I} = \{i = 1, 2, \ldots, n \leqslant N\},$$
$$\tilde{I}_1^s \cap \tilde{I}_2^s = \varnothing, s = 1, \ldots, S.$$

В начале процесса положим $\tilde{I}_1^s = \varnothing, \tilde{I}_2^s = \tilde{I}$. Вычислим значения следующих величин:

$$Z(i) = d(i) + \sum_{r=1}^{R} \sum_{j=1}^{m_i} t^r(i, j; k, l);$$
$$\Delta(i) = D(i) - Z(i), i = 1, \ldots, N; \quad (32)$$

$$\theta^s(i, p; k, l) = \max_{j \in \tilde{J}_i^{1s}} \theta^s(i, j; k, l); \quad (33)$$

$$Z^s(i) = \theta^s(i, p; k, l) + \sum_{r=\rho}^{R} \sum_{j \in \tilde{J}_i^{2r}} t^r(i, j; k, l),$$
$$\Delta^s(i) = D(i) - Z^s(i), i \in \tilde{I}_2^s, \rho \geqslant 1. \quad (34)$$

Значения $\Delta(i)$ и $\Delta^s(i)$ могут рассматриваться как граничные значения возможности выполнения ограничений на время завершения выполнения *i*-го задания в установленные сроки. Рассмотрим последовательности

$$\tilde{U} = \{i_1, i_2, \ldots, i_\gamma, i_{\gamma+1}, \ldots, i_N \mid \Delta(i_\gamma) \leqslant \Delta(i_{\gamma+1})\}, \quad (35)$$

а на *s*-м шаге итеративного процесса —

$$\tilde{U}^s = \{i_1^s, \ldots, i_\gamma^s, i_{\gamma+1}^s, \ldots, i_n^s \mid \Delta(i_\gamma^s) \leqslant \Delta(i_{\gamma+1}^s)\}, n \leqslant N. \quad (36)$$

Алгоритм предусматривает выполнение следующих шагов.

Шаг 1. Если $\tilde{I}_1^s = \tilde{I}, \tilde{I}_2^s = \varnothing,$ и выполняются все ограничения задачи (11) и (12), то получено допустимое решение задачи. Если какое-либо из этих ограничений не выполняется, то алгоритм завершает свою работу с сообщением, что не найдено допустимых решений задачи. В противном случае вычислим значения $Z(i)$ и $\Delta(i)$ по формулам (32). Если хотя бы для одного из индексов $i = 1, \ldots, N$ выполняется неравенство $\Delta(i) < 0$, то не существует допустимых решений, и алгоритм завершает работу. В противном случае выполняем следующие вычисления.

Пусть $\tilde{K}^r$ — множество машин на *r*-й стадии обработки. Тогда

1) выбираем первые в множестве (35) $\gamma = \min(K^1, N)$ заданий — $(i_1, \ldots, i_\gamma)$;

2) определяем номер операции и номер машины, на которой она выполняется:

$$(i_1, k_1) = \arg \min_{1 \leqslant k \leqslant K^1} [\max\{h(k), d(i_1)\} + t(i_1, j; k, l)], \quad (37)$$

$$\tilde{K}_2^1 = (\tilde{K}^1 / k_1), \ \tilde{K}_1^1 = \{k_1\}; \quad (38)$$

3) вычисляем также последовательно для всех остальных заданий

$$\bar{j}(i_v, k_v) = \arg \min_{k \in \tilde{K}_{v-1}^1} [\max\{h(k), d(i_v)\} + t(i_v, j; k, l)], \quad (39)$$
$$\tilde{K}_v^1 \in (\tilde{K}_{v-1}^1 / k_v), v = 2, \ldots, \gamma.$$

Здесь $\bar{j}(i_v, k_v)$ — индексы выбранных операций и номера машины i_v-го задания на данном этапе обработки;

4) рассчитываем значения $x^1(i_v, 1; k, 1)$, $\theta^1(i_v, 1; k, 1)$, $v = 1, 2, \ldots, \gamma,$ по формулам (1);

5) определяем

$$\tilde{J}_{i_v}^{11} = \bar{j}(i_v, k_v), \ \tilde{J}_{i_v}^{21} = \{\tilde{J}_{i_v} / \tilde{J}(i_v, k_v)\}; \ \tilde{I}_1^1 = \varnothing,$$
$$\tilde{I}_2^1 = \tilde{I}; \ \tilde{K}_1^1 = \left\{\bigcup_{v=1}^{\mu} k_v\right\}, \ \tilde{K}_2^1 = \{\tilde{K}_1 / \tilde{K}_1^1\}. \quad (40)$$

Полагаем $r = 2$ и переходим к выполнению шагов $s = 2,\ldots,S$.

Шаг s.

1) определяем подмножество заданий $\tilde{U}^s$ по формулам (36). Выбираем стоящие первыми $\gamma^s = \min(K^s, N^s)$ заданий — $(i_1^s,\ldots i_v^s,\ldots,i_\gamma^s)$. Здесь $N^s \leqslant N$ — число заданий, время завершения выполнения которых на всех стадиях обработки еще не определено на данном шаге работы алгоритма;

2) рассчитываем время завершения выполнения этих заданий $\theta^{s,r}(i_v^s, j; k, l)$, $r = 1,\ldots, R$, по формулам (1)—(8);

3) выбираем индекс задания с минимальным значением $\Delta(D_i)$ по формуле

$$\bar{i}^{\,s} = \min_{i\in\tilde{I}_2^s} \Delta(D_i).$$

4) полагаем $\tilde{I}_1^s = \{\tilde{I}_1^s \cup \bar{i}^{\,s}\}$, $\tilde{I}_2^s = \{\tilde{I}_2^s / \bar{i}^{\,s}\}$, $s := (s+1)$ и вновь переходим к выполнению шага 1.

Через некоторое число итераций либо будет получено допустимое решение задачи, либо установлено, что задача не имеет допустимых решений.

5. Алгоритм решения построения многостадийных расписаний в условиях отсутствия ограничений

Рассмотрим две стратегии выбора последовательности машин, на которых осуществляется выполнение задания u_i на каждой стадии обработки. На всех стадиях обработки $r = 1,2,\ldots,(R-1)$, кроме последней R-й стадии, выбор осуществляется согласно стратегии 1, а на последней — согласно стратегии 2. Пусть построена некоторая подпоследовательность выполнения заданий $\tilde{U} = \{u_1, u_2,\ldots,u_{i-1}\}$. Рассмотрим последовательность выбора машин, на которых будет выполняться установленное на последнее место в $\tilde{U}$ задание u_i.

Стратегия 1. В этой стратегии на каждом шаге осуществляется выбор той машины, на которой будет достигнуто минимальное время завершения выполнения этого задания. Пусть определены следующие подмножества: $\tilde{K}_1^r(u_i) = \{k_1^r, k_2^r,\ldots,k_{p-1}^r\}$, $\tilde{K}_2^r(u_i) = \{k_p^r, k_{p+1}^r,\ldots,k_{K^r}^r\}$ — подмножества машин r-й стадии обработки, на которых, соответственно, выполнены операции u_i-го задания и других машин, на которых еще должны быть выполнены операции этого u_i-го задания.

Пусть K^r — число машин на r-й стадии обработки; $T^r(u_i)$ — время завершения выполнения задания на всех машинах r-й стадии обработки.

Если $\tilde{K}_2^r(u_i) = \varnothing$, то выбор машины, стоящей на первом месте в последовательности $k_1 \in \tilde{K}_2^r(u_i)$ осуществляется из условий

$$g = \arg \min_{k_1\in\tilde{K}_2^r(u_i)} \{\max[d_{u_i}, h_{k_1}^1] + t_{u_i,k_1}^1\}, \qquad (41)$$

$$g = \arg \min_{k_p\in\tilde{K}_2^r(u_i)} \{\max[\theta_{u_1,,k_{p-1}}^r, h_{k_p}^r] + t_{u_i,k_p}^r\}, \quad r = 2,\ldots,R-1. \qquad (42)$$

Если $\tilde{K}_2^r(u_i) \neq \varnothing$, то для всех остальных машин $k_p \in \tilde{K}_2^r(u_i)$ этот выбор происходит из условий

$$g = \arg \min_{k_1\in\tilde{K}_2^r(u_i)} \{\max[\theta_{u_i,,k_{p-1}}^r, \theta_{u_{i-1},,k_p}^r] + t_{u_i,k_1}^r\}. \qquad (43)$$

Если $\tilde{K}_2^{r-1}(u_i) \neq \varnothing$ и $\tilde{K}_2^r(u_i) \neq \varnothing$, то выбор первой машины в последовательности $k_p \in \tilde{K}_2^r(u_i)$ осуществляем по правилам

$$g = \arg \min_{k_p\in\tilde{K}_2^r(u_i)} \{\max[\theta_{u_{i-1},,k_p}^r, T^{r-1}(u_i)] + t_{u_i,k_p}^r\}, \quad r = 2,\ldots,R-1. \qquad (44)$$

Каждый раз после выбора g-й машины выполняем преобразование

$$\tilde{K}_1^r(u_i) = \{\tilde{K}_1^r(u_i) \cup g\},\ \tilde{K}_2^r(u_i) = \{\tilde{K}_2^r(u_i)/g\}. \qquad (45)$$

Если $\tilde{K}_2^{R-1}(u_i) = \varnothing$, то алгоритм реализации стратегии 1 определения последовательности машин, на которых выполняется задание u_i на первых $(R-1)$ стадиях обработки, завершает свою работу.

Стратегия 2. На последней стадии обработки последовательность машин выбирается из условий минимизации значения нижней границы критерия оптимальности. Для выбранного решения значение нижней границы вычисляется по упрощенной формуле (23).

Пусть построена $\tilde{K}^R(u_i) = \{k_1^R, k_2^R,\ldots,k_{K^R}^R\}$ — последовательность машин, на которых осуществлено выполнение u_i-го задания на последней R-й стадии обработки. Известны $\tilde{I}_1(u_i)$ — подмножество заданий, стоящих в строящейся последовательности их выполнения перед заданием u_i; $\tilde{I}_2(u_i) = \{\tilde{I}/\tilde{I}_1(u_i)\}$ — подмножество заданий, подлежащих выполнению.

Времена начала и завершения выполнения заданий на каждой машине для этой последовательности вычисляются по формулам (41)—

(44) при $r := R$. Выбор последовательности машин $\tilde{K}^R(u_i)$ проводится из условий

$$\vartheta[F(u_i)] = \min \max_{k_p \in \tilde{K}^R} \left\{ (\theta^R_{u_{i,},k_p} + t^R_{u_{i,k_p}}) + \sum_{j \in \tilde{I}_2(u_i)} t^R_{j,k_p} \right\}. \quad (46)$$

Решение этой задачи часто тривиально, выполняется эвристическим алгоритмом, а при небольшом числе машин — полным или частичным перебором.

Алгоритм 3 — приближенное решение задачи.

Алгоритм построения последовательности выполнения заданий состоит из N шагов $s = 1, 2, \ldots, N-1$. На каждом шаге на последнее место в строящейся последовательности $\tilde{U}$ устанавливается одно задание из подмножества $\tilde{I}_2^s$. В начале процесса $s = 0$ положим $\tilde{I}_1^s = \tilde{U}^s = \varnothing$, $\tilde{I}_2^s = \tilde{I}$.

Шаг 1. Проверяем после установки каждого из заданий на 1-е место в строящейся последовательности $\tilde{U}^1$ эффективность каждой из альтернатив. Для этого рассчитываем для каждого из заданий по формулам (1)—(9) значения величин

$$x^1(i,1;k,1), x^r(i,1;k,l), x^r(i,j;k,l),$$
$$r = 2,\ldots,R;$$
$$\theta^r(i,j;k,l) = x^r(i,j;k,l) + t(i,j;k) - 1,$$
$$r = 1,\ldots,R.$$

Вычисляем индекс задания, для которого достигается минимальное значение нижней границы оптимального решения задачи по формуле

$$u_1 = \arg\min_{i \in \tilde{I}} \left[\theta^R(i,j;k) + \sum_{p \in \{\tilde{I}/i\}} t^R(p,j;k) \right]. \quad (47)$$

Задание с индексом $\bar{i}$ устанавливаем на 1-е место в строящейся последовательности $\tilde{U}^1 = \{u_1\}$, $\tilde{I}_1^1 = \{u_1\}$, $\tilde{I}_2^1 = \{\tilde{I}/u_1\}$. Переходим к выполнению шага s_1. Для каждого из заданий $s = 2,\ldots,N-1$ подмножества $\tilde{I}_2^s$ выполняем последовательно вычисления, описанные в шаге s_1 и шаге s_2. После выбора соответствующего задания на последнее место в строящейся последовательности проводим корректировку последовательности $\tilde{U}^s$ и корректировку подмножеств $\tilde{I}_1^s$ и $\tilde{I}_2^s$.

Шаг s_1. Если $\tilde{I}_2^s = \varnothing$, то построено расписание выполнения заданий, и алгоритм построения последовательности завершает работу. Переходим к выполнению последнего шага *S*. В противном случае для каждого из заданий $i \in \tilde{I}_2^s$ для стадий обработки $r = 1,2,\ldots,R-1$ проводим вычисления в соответствии со стратегией 1. Переходим к выполнению шага s_2.

Шаг s_2. На последней стадии обработки проводим вычисления в соответствии со стратегией 2. В результате этих вычислений будет осуществлен выбор задания $u_i \in \tilde{I}_2^s$, которое устанавливается на последнее место в строящейся последовательности $\tilde{U}^s$. Выполняем корректировку подмножеств $\tilde{I}_1^s$ и $\tilde{I}_2^s$, полагаем $s := (s+1)$, и переходим к выполнению шага s_1.

Шаг S. Построена оптимальная последовательность выполнения заданий $\tilde{U}^N$. Времена начала и завершения выполнения всех заданий, следующих друг за другом в последовательности $\tilde{U}^N$, вычислим по формулам (1)—(9). Время завершения расписания выполнения всех заданий (значение критерия оптимальности), которое для построенного расписания равно нижней границе оптимального решения, определяется выражением $F = \max_{i \in \tilde{I}} T_i^R$. На этом алгоритм завершает свою работу.

6. Иллюстративный пример

Решим с использованием алгоритма 1 задачу построения многостадийного расписания open-shop-problem в условиях отсутствия ограничений на времена начала и завершения выполнения заданий. Времена выполнения заданий машинами трех стадий обработки приведены в табл. 1.

В табл. 2 представлены суммарные времена выполнения заданий всеми машинами каждой стадии обработки.

Таблица 1

Времена выполнения заданий трех стадий обработки

№ заданий	Времена выполнения заданий машинами трех стадий обработки								
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии		
	1	2	3	1	2	3	1	2	3
1	7	6	9	3	8	10	5	12	10
2	8	7	3	4	12	8	3	15	6
3	10	4	7	9	7	6	8	9	4
4	5	10	5	3	11	5	6	10	8
5	3	6	12	9	7	8	7	7	10
Сумма	33	33	36	28	45	37	31	53	38

Таблица 2

Суммарные времена выполнения заданий

№ заданий	Суммарные времена выполнения заданий всех стадий обработки			
	1	2	3	Суммы всех трех стадий
1	22	21	27	70
2	18	24	24	66
3	21	21	21	63
4	20	19	24	63
5	21	24	24	69

Вычислим по формулам (22), (23) нижнюю границу длины многостадийного расписания.

$$\vartheta(T_1) = \max\{(22 + 3 + 7 + 5 + 12), (18 + 9 + 7 + 5 + 12), (21 + 9 + 3 + 5 + 12), (20 + 9 + 3 + 7 + 12), (21 + 9 + 3 + 7 + 5), (33 + 6 + 5), (33 + 10), (36 + 9)\} = 51;$$

$$\vartheta(T_2) = \max\{(21 + 8 + 6 + 5 + 8), (24 + 10 + 6 + 5 + 8), (21 + 10 + 8 + 5 + 8), (19 + 10 + 8 + 5 + 8), (24 + 10 + 8 + 6 + 5), (28 + 7 + 6), (45 + 3 + 5), (37 + 3 + 8)\} = 53;$$

$$\vartheta(T_3) = \max\{(27 + 6 + 4 + 8 + 10), (24 + 10 + 4 + 8 + 10), (21 + 10 + 6 + 8 + 10), (24 + 10 + 6 + 4 + 10), (24 + 10 + 6 + 4 + 8), (31 + 9 + 4), (53 + 3 + 6), (38 + 7 + 7)\} = 62;$$

$$\vartheta(F) = \max\{(51 + 21 + 21), (53 + 21 + 21), (62 + 21 + 21)\} = 104.$$

Решение задачи алгоритмом 1, включающим пять шагов, приводится ниже.

Шаг 1. Рассчитываем наиболее ранние времена выполнения каждого из заданий и соответствующие значения нижней границы. Результаты вычислений приведены в табл. 3. На первое место в строящейся последовательности выбираем четвертое задание.

Шаг 2. Выбор задания на второе место в последовательности. Времена выполнения заданий на втором шаге алгоритма приведены в табл. 4. Выбираем задание 1 и устанавливаем его на второе место в строящейся последовательности.

Шаг 3. Выбор задания на третье место в последовательности. Времена выполнения заданий на третьем шаге алгоритма приведены в табл. 5. Выбираем пятое задание на третье место в строящейся последовательности.

Таблица 3

Времена выполнения заданий на первом шаге алгоритма

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки									Значение нижней границы
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии			
	1	2	3	1	2	3	1	2	3	
1	13	6	22	25	33	43	60	55	70	98
2	18	10	3	22	42	30	60	57	66	98
3	21	4	11	43	34	27	64	52	56	96
4	5	20	10	23	39	28	45	55	65	96
5	3	9	21	45	28	36	79	69	62	115

Таблица 4

Времена завершения выполнения заданий на втором шаге алгоритма

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки									Значение нижней границы
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии			
	1	2	3	1	2	3	1	2	3	
1	12	27	21	30	46	38	51	67	77	98
2	13	27	16	31	51	39	54	70	83	107
3	15	26	22	43	50	34	58	67	71	107
5	8	28	22	37	52	45	57	64	75	100

Таблица 5

Времена завершения выполнения заданий на третьем шаге алгоритма

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки									Значение нижней границы
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии			
	1	2	3	1	2	3	1	2	3	
2	20	34	24	38	58	46	61	82	88	102
3	22	32	28	41	54	47	62	76	81	104
5	15	39	33	48	55	63	70	77	87	101

Шаг 4. Выбор задания на четвертое и пятое места в последовательности. Времена выполнения заданий на четвертом шаге алгоритма приведены в табл. 6.

Следовательно, на четвертое место в последовательности ставится третье задание, а на пятое — задание 2.

Времена начала и завершения выполнения заданий в построенной последовательности

Таблица 6

Времена завершения выполнения заданий на четвертом шаге алгоритма

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки									Значение нижней границы
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии			
	1	2	3	1	2	3	1	2	3	
2	23	43	36	52	67	75	78	93	99	103
3	25	44	40	53	62	68	76	86	91	101

Таблица 7

Результаты решения задачи open-shop-problem

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки									Значение нижней границы
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии			
	1	2	3	1	2	3	1	2	3	
4	5	20	10	23	39	28	45	55	65	96
1	12	27	21	30	46	38	51	67	77	98
5	16	39	33	48	55	63	70	77	87	101
3	25	44	40	53	62	68	76	86	91	101
2	33	51	43	55	67	75	79	100	106	106

$\tilde{U} = \{4,1,5,3,2\}$ на машинах всех стадий обработки приведены в табл. 7.

Для сравнения в табл. 8 приведено решение задачи flow-shop-problem в условиях строго фиксированных и одинаковых для всех заданий на всех стадиях обработки последовательностей выполнения всех операций.

Таблица 8

Результаты решения задачи многостадийной job-shop-problem

№ заданий	Времена завершения выполнения заданий машинами трех стадий обработки								
	Машины 1-й стадии			Машины 2-й стадии			Машины 3-й стадии		
	1	2	3	1	2	3	1	2	3
4	5	15	20	23	34	39	45	55	63
3	15	19	27	36	43	49	57	66	70
2	23	30	33	40	55	63	66	81	87
5	26	36	48	57	64	72	79	86	96
1	33	42	57	60	68	82	87	98	108

Заключение

Рассматривается постановка задачи теории расписаний для дискретных производств в условиях, когда множество изделий должно пройти последовательную обработку в R структурных подразделениях предприятия (например, последовательная цепочка, несколько участков и (или) цехов с различным числом и отличающимся своими техническими характеристиками оборудованием). Каждое такое структурное подразделение рассматривается как отдельная стадия производства. Такими стадиями обработки могут быть группы станков, многопроцессорные, многопоточные или автоматические линии. На каждой стадии производства (обработки изделий) очередность выполнения технологических операций может осуществляться в произвольной последовательности (open-shop-problem). Необходимо найти последовательность запуска в производство этого множества изделий, обеспечивающую выполнение всех ресурсных ограничений и оптимальное значение некоторого критерия оптимальности.

Предложены математические модели, установлены свойства допустимых и оптимальных расписаний, для выбранной последовательности запуска изделий приведены формулы расчета времен начала и завершения выполнения операций для всех структурных подразделений производства, а также значение нижней границы критерия оптимального решения для анализируемых подпоследовательностей на всех этапах решения.

Сформулированные задачи относятся к классу *NP*-трудных проблем экспоненциальной сложности. Разработаны алгоритмы приближенного решения этих задач полиномиальной сложности, использующие разные эвристики на различных этапах решения, которые проиллюстрированы на числовом примере.

Эффективность предложенных алгоритмов приемлема для практических приложений. Полученные в работе результаты могут использоваться при проектировании систем календарного планирования и организации работы единичного, серийного и массового производства в различных отраслях промышленности.

Список литературы

1. **Конвей Р. В., Максвелл В. Л., Миллер Л. В.** Теория расписаний. М.: Физматгиз, Наука. 1975. 359 с.

2. **Танаев В. С., Сотсков Ю. Н., Струсевич В. А.** Теория расписаний. Многостадийные системы. М.: URSS, 1989. 328 с.

3. **Зак Ю. А.** Прикладные задачи теории расписаний и маршрутизации перевозок. М.: URSS, 2012. 394 с.

4. **Лазарев А. А.** Теория расписаний. Методы и алгоритмы. М.: ИПУ РАН, 2019. 408 с.

5. **Лазарев А. А., Графов Е. Р.** Теория расписаний. Задачи и алгоритмы. М.: Изд-во МГУ, 2011. 224 с.

6. **Пяткин А. В., Черных И. Д.** Задача open-shop с маршрутизацией двухвершинной сети и разрешением прерываний // Дискретный анализ и исследование операций. 2012. Т. 19, № 3. С. 65—78.

7. **Brucker P.** Scheduling Algorithms. Berlin, Heidelberg und New York. Springer-Verlag, 1998.

8. **Domschke W., Scholl A., Voß S.** Produktionsplanung. Ablauforganisatorische Aspekte. Berlin, Heidelberg, Springer Verlag, 2005. 456 p.

9. **Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G.** Minimizing maximum lateness in two-machine open shop // Mathematics of Operations Research. 1981. N. 6. P. 153—158.

10. **Achubgue J. O., Chin F. Y.** Scheduling the open-shop to minimize mean flow time // SIAM Journal on Computing. 1982. N. 11. P. 709—720.

11. **Du J., Long J.** Minimizing mean flow time in two machine open shops and flow shops // Journal of Algorithms. 1993. 14. P. 24—44.

12. **Averbakh I., Berman O., Chernykh I.** The routing open-shop problem ona network: complexity and approximation // Eur. J. Oper. Res. 2006. Vol. 173, N. 2. P. 531—539.

13. **Gonzalez T., Sahni S.** Open shop scheduling to minimize finish time // J. ACM. 1976. Vol. 23, N. 4. P. 665—679.

14. **Williamson D. P., Hall L. A., Hoogeveen J. A., Hurkens C. A. J.,Lenstra J. K., Sevastianov S. V., Shmoys D. B.** Short shop schedules // Oper. Res. 1997. Vol. 45, N. 2, P. 288—294.

Yu. A. Zack, D. Sc., e-mail: yuriy_zack@hotmail.com,
Aachen, Germany

Multistage Open-Shop-Problem: Schedules for N Tasks on K Machines with Different Technical Characteristics for an Arbitrary order of Tasks

The statements and mathematical models of the open — shop-problem are considered: performing N tasks on K machines with an arbitrary order of processing each task on all machines. Unlike most publications on solving this problem, the task posed is solved in a multi-stage formulation under the conditions of a sequential chain of sections and workshops of the enterprise, as well as in the presence of restrictions on the start and end times of tasks and the permissible operating times of machines. In addition, for scheduling systems of enterprises, it is extremely important to consider this task in a multi-stage setting, i.e. in a sequential chain of plots and workshops. The properties of admissible and optimal schedules, as well as algorithms of exact and approximate methods for solving these problems by successive optimization algorithms are investigated. At the early stages of solving the problem, the fact of incompatibility of the system of constraints of the task and the definition of a subset of tasks or resources, the time range of which should be expanded, is established. The described algorithms for solving the problem are illustrated by numerical examples.

Keywords: *multi-stage schedules, open — shop-problem, restrictions on the execution time of tasks, lower bound on the length of the schedule, sequential optimization algorithms, heuristic algorithm*

DOI: 10.17587/it.27.249-258

References

1. **Konwej R. W., Maxwell W. L., Müller W. L.** Scheduling theory, Moscow, Fismatgis, Nauka, 1975, 359 p. (in Russian).

2. **Tanajev V. S., Sotskov Yu. N., Strusevitch W. A.** Schedule theory. Multistage systems, Moscow, URSS, 1989, 328 p. (in Russian).

3. **Zack Yu. A.** Applied problems of the theory of schedules and traffic routing, Moscow, URSS, 2012, 394 p. (in Russian).

4. **Lazarev A. A.** Schedule theory. Methods and Algorithms, Moscow, IPU RAN, 2019, 408 p. (in Russian).

5. **Lazarev A. A., Grafov E. R.** Schedule theory. Tasks and algorithms, Moscow, Publishing house of MGU, 2011, 224 p. (in Russian).

6. **Pyatkin A. V., Chernich I. D.** Open-shop task with two-node routing and interrupt resolution, *Discrete Analysis and Operations Research,* 2012, vol. 19, no. 3, pp. 65—78 (in Russian).

7. **Brucker P.** Scheduling Algorithms, Berlin, Heidelberg und New York, Springer-Verlag, 1998.

8. **Domschke W., Scholl A., Vo@ S.** Produktionsplanung. Ablauforganisatorische Aspekte, Berlin, Heidelberg, Springer Verlag, 2005, 456 p.

9. **Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G.** Minimizing maximum lateness in two-machine open shop, *Mathematics of Operations Research*, 1981, no. 6, pp. 153—158.

10. **Achubgue J. O., Chin F. Y.** Scheduling the open-shop to minimize mean flow time, *SIAM Journal on Computing*, 1982, no. 11, pp. 709—720.

11. **Du J., Long J.** Minimizing mean flow time in two machine open shops and flow shops, *Journal of Algorithms*, 1993, vol. 14, pp. 24—44.

12. **Averbakh I., Berman O., Chernykh I.** The routing open-shop problem ona network: complexity and approximation, *Eur. J. Oper. Res.*, 2006, vol. 173, no. 2, pp. 531—539.

13. **Gonzalez T., Sahni S.** Open shop scheduling to minimize finish time, *J. ACM,* 1976, vol. 23, no. 4, pp. 665—679.

14. **Williamson D. P., Hall L. A., Hoogeveen J. A., Hurkens C. A. J., Lenstra J. K., Sevastianov S. V., Shmoys D. B.** Short shop schedules, *Oper. Res.*, 1997, vol. 45, no. 2, pp. 288—294.

УДК 004.056.53 DOI: 10.17587/it.27.259-266

Л. В. Аршинский, д-р техн. наук, проф., e-mail: larsh@mail.ru,
Г. Н. Шурховецкий, аспирант, e-mail: gshn5@yandex.ru,
Иркутский государственный университет путей сообщения

Особенности применения метода рассечения—разнесения для безопасного хранения данных во внешних хранилищах

Рассматриваются вопросы применения метода рассечения—разнесения для защищенного хранения информации во внешних, в первую очередь облачных, хранилищах данных. Анализируются различные подходы к реализации метода, включая патентный материал. Показывается, что метод наиболее эффективен при побитовом рассечении файлов и случайном размещении битов в потоках данных, направляемых в отдельные хранилища.

Ключевые слова: *защита информации, хранилища данных, облачные технологии, метод рассечения—разнесения*

Введение

Безопасность хранения данных во внешних, в первую очередь облачных, хранилищах данных — одно из активно обсуждаемых сегодня направлений информационной безопасности (см., например, работы [1—9]). При этом одной из основных проблем применения облачных технологий для хранения данных является доверие к поставщикам этих услуг [2—4]. Согласно работе [5] несанкционированный доступ персонала и кража конфиденциальной информации являются весьма значимой угрозой при облачном хранении. К сожалению, сегодня ни один поставщик не гарантирует в полной мере защиту клиентских данных от постороннего вмешательства. Угроза вполне реальная, поскольку на некоторых ресурсах не обеспечивается даже шифрование [1, 4]. Если злоумышленнику удастся обойти систему защиты, он получит возможность работать со сведениями, которые клиент предпочел бы сохранить в тайне. Причем шифрование на стороне поставщика также не освобождает от рисков доступа посторонних к клиентским данным, например, в силу недобросовестности персонала, договоренностей с третьими лицами и т.п. Это существенно снижает интерес к таким технологиям со стороны потенциальных клиентов [6, 7].

Несмотря на комплексность проблемы [7—9], одним из наиболее распространенных элементов защиты сегодня является шифрование [3—9]. Надежные алгоритмы плюс не скомпрометировавший себя ключ существенно повышают защищенность данных, хотя по-прежнему остаются сомнения в их полной безопасности в силу, например, неудачного выбора ключа, его случайной компрометации, недокументированных проблем в алгоритмах и т. п. Основная уязвимость здесь — хранение всего объема информации в одном месте, хотя и в защищенном виде. В связи с этим возникает вопрос поиска иных, в том числе некриптографических, способов защиты, которые снизят риск внешнего доступа к важной для клиента информации.

Среди некриптографических методов наибольший интерес вызывает процедура рассечения исходной информации на части с раздельной обработкой частей. В телекоммуникации и стеганографии известен метод пространственно-временного распыления, заключающийся в дроблении исходного сообщения на возможно мелкие составляющие (предложения, слова, символы, блоки символов, группы байтов, байты, группы битов, биты) с последующей их передачей частями, распределенными по нескольким каналам связи (см., например, работы [10—12]). Также известен метод рас-

сечения—разнесения, также представляющий собой метод дробления (рассечения) исходной информации на части с последующей передачей частей по разным каналам связи или размещением в разных хранилищах (см., например, работы [13—15]). В отличие от передачи информации как достаточно кратковременного процесса при ее помещении в хранилище у "заинтересованных лиц" появляется, условно говоря, неограниченное время доступа к ней, что расширяет возможности атакующего. В связи с этим возникает проблема анализа особенностей подобного хранения с точки зрения информационной безопасности.

1. Описание метода

Метод пространственно-временного распыления основан на защите передаваемых данных путем их дробления на блоки, которые затем направляются от источника A к получателю B в определенные моменты времени [12]. Блоки могут помещаться в контейнеры, среди которых могут быть и пустые (ложные). Этот прием, в частности, используется в стеганографии [10]. Пространственно-временное распыление — хороший способ защиты при передаче данных, однако характер информации, помещаемой в хранилища, накладывает свои требования, в частности, большой объем и недопустимость частой замены (отсутствует "временная" часть), продолжительное хранение информации, размещение ее "открытым" образом и т. д. Как результат, приходится ограничиваться пространственным"распылением" в виде рассечения файлов на отдельные фрагменты и разнесения их по различным, в том числе географически-удаленным, местам (хранилищам). На этом основан известный метод рассечения—разнесения. Идея метода сама по себе проста. Информация делится на части, которые помещаются в разные, в том числе географически удаленные, места хранения так, что владение одним куском не дает представление об информации в целом. Это снижает ущерб от постороннего вмешательства. Наличие сегодня большого числа сервисов облачного хранения дает такую возможность, и ею следует пользоваться. Поскольку принцип рассечения и места хранения частей в идеале знает только владелец информации, он может быть относительно спокоен за нее, особенно, если она не имеет явного текстового или иного подобного характера, позволяющего воспользоваться сведениями, хранящимися в одной отдельной части. Примером могут быть фото- и видеоданные, звуковые файлы и т. п. Но даже текстовая информация может быть достаточно надежно сохранена при надлежащем использовании метода. Рассмотрим его возможности при двух основных подходах к рассечению—разнесению:

1. Рассечение "монолитное", когда информация делится на равные или не равные части с разнесением в различные, в том числе географически удаленные, хранилища так, что каждая часть представляет собой слитный — единый и непрерывный — фрагмент исходной информации.

2. Рассечение "диффузное", когда отдельная часть (назовем ее потоком) содержит фрагменты различных участков исходного материала, так что при завладении одним или даже частью потоков смысл исходного сообщения оказывается трудноустановим.

Проанализируем возможности такого метода при разных способах реализации.

2. Анализ особенностей

Рассмотрим *первый подход.*

Самый простой случай — это размещение исходной информации в единственном потоке с направлением ее в хранилище. По сути это означает ее незащищенное хранение и упомянуто для полноты изложения.

О защищенности можно говорить, если потоков более одного. Тогда атакующий владеет лишь частью информации и оказывается перед задачей восстановления всего содержимого по имеющемуся фрагменту. Здесь возможны два варианта.

I. Для работы с файлом необходимы все фрагменты. В этом случае обладание единичным фрагментом ничего не дает. Такая реализация метода обеспечит надежную защиту, во всяком случае в той степени, в какой информация защищена отсутствием других фрагментов. Однако далеко не все пользовательские файлы обладают такой возможностью.

II. Имеющийся фрагмент позволяет выявить хранящуюся в нем информацию. Например, владея куском текста, мы воспринимаем его, хотя недостающие части придется восстанавливать. Последнее можно попытаться сделать или подбором, или додумывая содержимое по полученному фрагменту. И то и другое достаточно проблематично, но тем не менее, частью информации при таком подходе атакующий владеет. Если же он владеет всеми фрагментами, то восстановление информации потребует $S!$ комбинаций потоков, где S — их

число. При небольшом числе потоков (реальный сценарий) это небольшая величина. Иными словами, владение всеми потоками делает такую реализацию метода уязвимой.

При *втором подходе* исходная информация делится на части существенно меньше длины потока. Части складываются в потоки внешне случайным, неизвестным атакующему образом. О ключе — порядке формирования потоков — осведомлен только владелец информации, и только он, в идеале, способен восстановить исходный файл из потоков. Алгоритм в общем случае может выглядеть следующим образом:

1. Задать закон рассечения—разнесения, для чего:

1.1. Определить характер фрагментации исходного сообщения, разбив его на фрагменты, существенно меньшие, чем длина будущих потоков, например, на слова, символы, группы символов определенной длины и т. п.

1.2. Задать ключ рассечения—разнесения в виде набора $P = \{(p_1,a_1), ..., (p_l,a_l), ..., (p_L,a_L)\}$, где p_l — идентификаторы (к примеру, номера) потоков, куда будут направлены соответствующие фрагменты (идентификаторы могут повторяться), a_l — относительный (в пределах действия ключа) адрес l-го фрагмента в соответствующем потоке (фрагмент помещается в позицию с номером a_l), L — длина ключа. К примеру, для $P = \{..., (p^*,2), ..., (p^*,1), ...\}$ к потоку p^* сначала будет присоединен фрагмент, помеченный $(p^*,1)$, а затем фрагмент $(p^*,2)$. Таким образом, произойдет изменение порядка следования (перемешивание) фрагментов в потоке. Порядок, в котором нет перемешивания, рассматриваем как "нормальный". В данном примере это $P = \{..., (p^*,1), ..., (p^*,2),...\}$. Если порядок нормальный, адрес a_l можно не указывать и записывать ключ как $P = \{p_1, p_2,..., p_L\}$. В этом случае очередной фрагмент присоединяется в конец соответствующего потока.

2. Поставить в соответствие каждому фрагменту двойку (p_l, a_l) или идентификатор p_l, если порядок нормальный. Очевидный способ — последовательное "наложение" ключа на исходное сообщение.

3. Сформировать потоки согласно ключу. Например, для трех потоков, ключей {1, 2, 3} и фрагментов-слов текст "Однажды, в студеную зимнюю пору, я из лесу вышел; был сильный мороз." разобьется следующим образом:

3.1. Однажды, зимнюю из был

3.2. в пору, лесу сильный

3.3. студеную я вышел; мороз.

4. Поместить потоки в соответствующие места хранения.

5. Стоп!

Сборка исходного сообщения выполняется обратной процедурой:

1. Собрать потоки из мест хранения.

2. Пользуясь законом рассечения—разнесения:

1.1. Выполнить последовательное считывание фрагментов из потоков.

1.2. Собрать фрагменты в исходное сообщение.

3. Стоп!

Видно, что в принципе процедура легко выполнима и обеспечивает более высокую степень защиты. Здесь каждый поток содержит "произвольные" фрагменты сообщения, в результате чего сложнее установить его смысл по отдельному потоку. Хотя, если фрагментами являются слова или достаточно большие группы символов, а информация текстовая, можно предположить, чему она посвящена. Однако для некоторых файлов, например, содержащих фото-, видео-, аудиоданные и файлов программ (это типичные случаи [4]) этот прием может быть вполне подходящим.

Если у атакующего в распоряжении имеется только часть потоков, ему необходимо:

1. Подобрать закон рассечения—разнесения, для чего:

1.1. Предположить, сколько потоков было всего (гипотеза 1).

1.2. Предположить, как выполнялось рассечение на фрагменты (гипотеза 2).

1.3. Предположить, какое место занимали фрагменты имеющихся потоков в исходном информационном массиве (гипотеза 3).

2. Последовательно разместить имеющиеся фрагменты согласно гипотезам 1—3 в едином сообщении, оставляя пропуски для отсутствующих фрагментов.

3. Заполнить пропуски так, чтобы получилось осмысленное сообщение (гипотеза 4).

Число вариантов здесь может быть крайне велико. В частности, если $P(L,S)$ — ключ рассечения для S потоков, а $\boldsymbol{P(L,S)}$ — множество таких ключей, то мощность множества равна числу возможных цепочек $\{(p_1,l_1),(p_2,l_2),...,(p_L,l_L)\}$:

$$|\boldsymbol{P}(\boldsymbol{L},\boldsymbol{S})| = \frac{(L-1)!}{(S-1)!(L-S)!}L!. \qquad (1)$$

Здесь $\frac{(L-1)!}{(S-1)!(L-S)!} = C_{L-1}^{S-1}$ — число композиций длиной S числа L, т.е. наборов $\{n_1, ..., n_s, ..., n_S\}_j$ таких, что $n_1 + ... + n_s + ... + n_S = L$, где s — номер потока, j — номер композиции; $L!$ — число перестановок пар (p_l,a_l) в ключе.

Несложно заметить, что мощность $|\boldsymbol{P(L,S)}|$ растет с увеличением L. Наибольших значений она достигает при $S \approx \frac{L}{2}$, однако при большой длине ключа это может создать проблемы с размещением. Того же результата можно добиться простым увеличением L при малом числе потоков S.

Для нормального порядка вместо соотношения (1) работает формула

$$|P(L,S)| = \sum_{j=1}^{C_{L-1}^{S-1}} \frac{L!}{(n_1!\cdot\ldots\cdot n_s!\cdot\ldots\cdot n_S!)_j}. \qquad (2)$$

Так как $n_1 + ... + n_s + ... + n_S = L$, выражение под суммой можно оценить как $q(S,L)^L$, где $q(L,S) > 1$. Это позволяет представить формулу (2) как

$$|\boldsymbol{P(L,S)}| = \frac{(L-1)!}{(S-1)!(L-S)!} q(L,S)^L. \qquad (3)$$

Функция (3) растет существенно медленнее, чем (1), однако является функцией показательного роста. В частности, при $S = 2$ имеем:

$$|\boldsymbol{P(L,S)}| = 2^L - 2.$$

Наконец, частным случаем (1) и (2) при $S = L$ будет

$$|\boldsymbol{P(L,S)}| = S!. \qquad (4)$$

Последнее означает, что самый плохой вариант использования метода рассечения—разнесения — совпадение длины ключа с числом потоков.

В целом из соотношений (1)—(4) следует, что:

I. Защищенность в большей мере зависит от длины ключа, чем от числа потоков.

II. Длина ключа должна превышать число потоков: $L > S$, причем высокая степень защищенности (число подбираемых вариантов закона рассечения) может быть достигнута уже одним увеличением L. Увеличение числа потоков S также повышает защищенность, но это усложняет процедуру размещения информации в хранилищах и зависит от внешних условий, тогда как L определяется владельцем информации. Хотя рассечение—разнесение — это не криптологическая парадигма защиты, тем не менее *принцип максимизации длины ключа* сохраняется и здесь.

III. Метод рассечения—разнесения обеспечивает наилучшую защиту, когда закон с точки зрения атакующего выглядит *случайным*. В противном случае она обеспечивается только сложностью получения всех потоков.

IV. Для повышения защищенности рекомендуется соблюдать следующее условие. Если F — длина файла, а f — характерный размер фрагмента, отношение $C = \frac{F}{f}$, которое является своеобразным показателем или коэффициентом дробления, должно быть *максимально*:

$$C = \frac{F}{f} \to \max,$$

или в более слабом варианте, $f \ll F$. Действительно, с учетом того, что выполняется условие $L \leqslant C$, величина L может быть тем больше, чем больше C. Это особенно важно для случаев, когда защищаемые файлы невелики. Для файлов большого размера ключ P может применяться периодическим образом.

V. При должном подборе ключа метод обеспечивает защиту против угрозы "атакующий владеет всеми потоками".

В результате можно заключить:

1. *При достаточно большой длине ключа и его случайном характере обладание даже всеми потоками не позволит атакующему восстановить сообщение.* Отметим, что эффект достигается даже при небольшом числе потоков, например, равном 2 или 3.

2. Рост L обеспечивается ростом коэффициента дробления C.

Далее. Если атакующий владеет частью потоков и знает закон рассечения—разнесения, ему необходимо заполнить пропуски (отсутствующие фрагменты — гипотеза 4). Если файл текстовый, а фрагменты — символы, можно попробовать подбирать исходя из контекста частотности пар, троек и т. д. символов. Для нетекстовых файлов это проблематично, потому в общем случае считаем, что сложность подбора пропорциональна числу вариантов выбора. Его можно оценить величиной 2^E, где E число отсутствующих битов. Например, если у атакующего имеется в распоряжении два из трех потоков, потоки имеют равную длину, а размер исходного файла 1Мбайт (довольно типичная величина), число вариантов $\sim 2^{280000}$. Даже для маленького по сегодняшним меркам файла 1Кбайт это составит $\sim 2^{2730}$ вариантов. Более чем астрономическая величина! Это означает, что незнание всех потоков при отсутствии закона рассечения еще более (можно сказать драматически) ухудшает ситуацию, так как наравне с подбором ключа атакующий должен заполнить пропуски.

Если атакующий владеет всеми потоками, а ключ имеет длину, равную S, число проверочных комбинаций составляет вполне приемлемую величину порядка $S!$. Незнание длины ключа усложнит процедуру подбора в L раз:

$$|\boldsymbol{P}(\boldsymbol{L},\boldsymbol{S})| = L \cdot S!,$$

но это не принципиально. Как только получится осмысленная комбинация начальных фрагментов потоков, ее можно применить к оставшейся части данных и восстановить искомый файл. Таким образом, регулярность рассечения—разнесения в случае владения всеми потоками при $L = S$ делает метод уязвимым к атаке "владение всеми потоками". Защитой может служить только низкая вероятность или невозможность получения всех потоков. Повысить защищенность можно, меняя ключ вдоль всего исходного сообщения, а также изменяя длину фрагментов, но это приводит к усложнению процедуры и чрезмерному росту длины ключа.

В целом для "диффузного" варианта метода рассечения—разнесения защищенность D информации можно оценить величиной

$$D \sim \frac{2^E}{\varepsilon^S} \cdot |\, \boldsymbol{P}(\boldsymbol{L},\boldsymbol{S}) \,| - 1, \qquad (5)$$

где E — число неизвестных битов сообщения; ε — вероятность получения доступа к потоку.

Из вышесказанного следует, что наилучшим способом реализации метода служит рассечение на минимально возможные фрагменты при достаточно большой длине случайно сгенерированного ключа. Даже при ε, близком к 1, и небольшом числе потоков это надежно защищает данные. Наименьшим фрагментом информации является бит.

3. Побитовое рассечение

Рассечение—разнесение как способ защиты информации известен достаточно давно. Информацию в целях ее сокрытия можно не только шифровать, но и делить на части так, чтобы она нигде не была представлена полностью [16]. На этом строятся многие современные подходы к работе с закрытыми сведениями. Для всей полноты сведений фрагменты нужно получить и сложить в определенном порядке. Защищенность здесь определяется главным образом трудностью получения всех фрагментов, хотя при монолитном рассечении владение даже одним из них может дать какое-то представление об общем содержимом.

Монолитное рассечение—разнесение — подходящий прием, когда дробление на слишком мелкие фрагменты с их последующими хранением, получением и сборкой трудноосуществимо, а вероятность доступа к потокам мала. С развитием вычислительных и информационно-коммуникационных технологий такое рассечение целесообразно заменять диффузным, при котором фрагменты могут предельно минимизироваться, а затем передаваться, храниться и собираться в любой последовательности. Сами потоки при этом становятся внешне бессмысленными, что придает защите новое качество.

Пример диффузного подхода представлен в работах [13—15]. Здесь предлагается посимвольная фрагментация сообщений с разнесением символов по потокам согласно некоторой таблице с вычислением идентификатора (номера) потока по формуле

$$p_s = n(r_i - 1) + c_j,$$

где n — число столбцов; r_i — числовой идентификатор строки; c_j — идентификатор столбца в таблице. Порядок формирования потоков нормальный. Например, для двух строк с идентификаторами $r = \{2,1\}$ и четырех столбцов с идентификаторами $c = \{4,1,3,2\}$ получаем табл. 1.

Таблица 1

	4	1	3	2	4	1	3	2	4	1	3	2	4	1	3	2
2	О	д	н	а	в	—	с	т	у	ю	—	з	ю	п	о	р
1	ж	д	ы	—	у	д	е	н	и	м	н	ю	у			

Формируемые потоки:
$p = 1$: ддм
$p = 2$: _нюп
$p = 3$: ыен
$p = 4$: жуиу
$p = 5$: д_юп
$p = 6$: атзр
$p = 7$: нс_о
$p = 8$: Овую

Такой же прием демонстрируется во многих других источниках. К сожалению, как уже говорилось, это наихудшая реализация метода с защищенностью $S!$ при получении всех потоков (увеличение числа строк и столбцов не меняет дела, поскольку длина ключа всегда совпадает с числом потоков). В вышеприведенном примере, в частности, закон рассечения является посимвольным с ключом {8,5,7,6,4,1,3,2}, т. е. $L = S$.

Таблица 2

H	e	l	l	o	,	w	o	r	l	d	!
01001000	01100101	01101100	01101100	01101111	00101100	01110111	01101111	01110010	01101100	01100100	00100001

Таблица 3

$	f	v	W	V	D
00100100	01100110	01110110	01010111	01010110	01000100

Таблица 4

Л	к	■	√	╩	б
10001011	10101010	10110010	11111011	11001010	10100001

Более интересным примером является метод защиты, представленный в патенте US 10216822B2 от 26.02.2019 "Data distribution methods and systems" [17, 18]. Согласно патенту исходная информация обрабатывается как последовательность *битов* — минимально возможных фрагментов информации. Исходные байты рассекаются на биты, которые направляются в различные потоки, например, по принципу "чет—нечет" для двух потоков. В результате бессмысленным становится не только "текст", но случайный характер приобретают и образующиеся байты. В простейшем варианте, когда потоки нормальны, получается следующий результат.

Исходное сообщение представлено в табл. 2, поток 1 — в табл. 3, поток 2 — в табл. 4.

Предлагаются и более сложные разнесения на несколько потоков с перемешиванием (рис. 1, 2). Как видно из рис. 1 и 2, рассечение—разнесение здесь не является нормальным, что повышает защищенность.

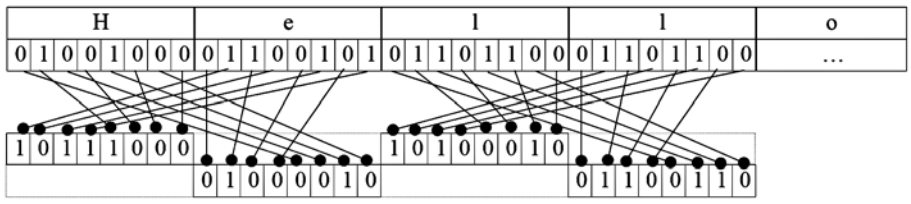


Рис. 1. Усложненное формирование потоков при побитовом рассечении по принципу "чет—нечет" с перемешиванием

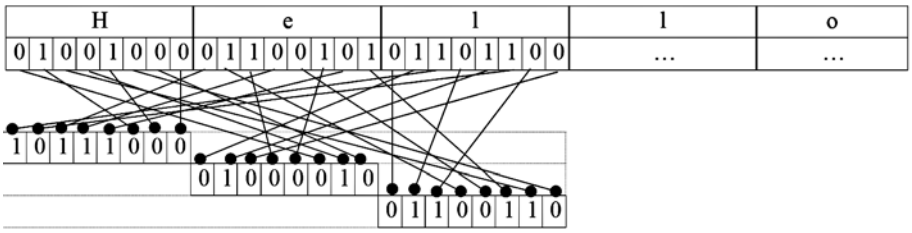


Рис. 2. Формирование потоков при побитовом рассечении с тремя потоками и перемешиванием

Для рис. 1 закон рассечения—разнесения выглядит как побитовое рассечение с 16-элементным ключом:

$$P(16,2) = \\ = \{(2,5),(1,5),(2,6),(1,6),\ (2,7),(1,7),(2,8),(1,8),\\ (2,1),(1,1),(2,2),(1,2),(2,3),(1,3),(2,4),(1,4)\}.$$

Для рис. 2 ключ 24-элементный:

$$P(16,2) = \{(2,6),(1,6),(3,7),(2,7),(1,7),(3,8),(2,8),(1,8),\\ (1,3),(3,4),(2,4),(1,4),(3,5),(2,5),(1,5),(3,6),\\ (3,1),(2,1),(1,1),(3,2),(2,2),(1,2),(3,3),(2,3)\}.$$

Особенностью реализации метода в работах [17, 18] является создание параллельно основным потокам потоков четности, получаемых из основных, их сложением посредством исключающего ИЛИ: *x*XOR*y*. Это повышает надежность, позволяя восстанавливать информацию при потере части потоков, но требует, чтобы потоки имели равную длину, что уменьшает вариативность ключа (хотя в случае перемешивания защищенность остается по-прежнему высокой). При нормальном порядке разнесения битов по потокам число возможных ключей равно (сравните с (2))

$$|\boldsymbol{P}(\boldsymbol{L},\boldsymbol{S})| = \frac{L!}{((L/S)!)^S} \sim q\left(L,S\right)^L, \text{ где } q > 1.$$

Для ключей с перемешиванием:

$$|\boldsymbol{P}(\boldsymbol{L},\boldsymbol{S})| = L!.$$

В обоих случаях работает только одна композиция числа L, а именно такая, что $n_1 = \ldots = n_S = \frac{L}{S}$. Несложно видеть, что требование равенства потоков снижает защищенность в $C_{L-1}^{S-1} = \frac{(L-1)!}{(S-1)!(L-S)!}$ раз.

Заметим, что регулярное (не случайное) разнесение битов по потокам при небольшой длине ключа и равных долях идентификаторов снижает защищенность при атаке "владение всеми потоками".

Если длина ключа не известна, показатель защищенности содержит умножение

на L (перебор всех вариантов ключей разной длины).

Общий принцип побитового рассечения—разнесения можно сформулировать как способ защищенной передачи и хранения информации, размещаемой в облачных и внешних хранилищах данных, который включает в себя формирование и преобразование первичной цифровой информации, отличающийся тем, что биты первичной цифровой информации образуют несколько битовых последовательностей так, *что существует не менее одного байта первичной цифровой информации, биты которого были помещены в разные последовательности*, с последующей передачей и размещением последовательностей в разных хранилищах данных, причем обеспечивается обратимый и контролируемый процесс. Здесь отсутствуют какие-либо ограничения на характер рассечения, позволяя генерировать ключи любой сложности и потоки различной длины. Например, фраза "Hello, world!" может быть разбита на два неравных потока:

1) Qп8у€{н9ў@

2) °¬Ђ

Программная реализация такого подхода даже в простейшем случае двух потоков с нормальным порядком согласно фоормуле (5) дает защищенность

$$D = \frac{2^E}{\varepsilon^S}(2^L - 2) - 1.$$

Главным отличием такого более общего взгляда служит то, что потоки могут создаваться путем побитового рассечения исходного файла любым образом. Принцип равенства потоков при этом может не соблюдаться.

Заключение

Подытоживая, можно заключить следующее:

1. В условиях повсеместного внедрения технологий облачного хранения данных остро встал вопрос защищенности размещаемой в них информации.

2. Облачные провайдеры недостаточно внимания уделяют вопросам защиты клиентских данных, что существенно снижает интерес к этой технологии.

3. В случае криптографической защиты на стороне провайдера по-прежнему остается вопрос доверия к нему в случае размещения критически важной информации.

4. Одним из перспективных способов защиты информации в облаках является метод рассечения—разнесения, заключающийся в том, что информация делится на фрагменты, из которых по определенному алгоритму формируются потоки данных, направляемые в различные хранилища так, что владение одним потоком даже в незашифрованном виде не позволяет атакующему воспользоваться полученными сведениями.

5. Защищенность тем выше, чем выше коэффициент дробления информации и больше длина ключа. Число потоков при этом существенной роли не играют.

6. Наибольшую защищенность обеспечивает побитовое рассечение, притом так, что потоки формируются из битов исходной информации случайным образом. Какой-либо регулярности следует избегать. Это обеспечивает высокий уровень защиты даже при получении атакующим всех потоков. При завладении лишь частью потоков перед ним стоит вообще труднореализуемая задача заполнения неизвестных битов, особенно если файлы имеют нетекстовый характер.

Список литературы

1. **Ступина М. В., Шпаков Д. В.** К вопросу безопасности облачных технологий // Молодой ученый. 2016. № 9(113). С. 85—88.

2. **Cloud Security:** взгляд на российский рынок // Information Security = Информационная безопасность [Электронный ресурс]. URL: https://lib.itsec.ru/articles2/cloud-security/cloud-security-vzglyad-na-rossiiskii-rinok (дата обращения: 02.02.2021).

3. **Довгаль В. А.** Методы повышения безопасности в сфере "облачных" технологий // Вестник АГУ. 2014. № 4(147). С. 170—174.

4. **Шпаков Д. В.** Изучение проблемы развития и безопасности облачных технологий [Электронный ресурс]. URL: https://docplayer.ru/57291877-Izuchenie-problemy-razvitiya-i-bezopasnosti-oblachnyh-tehnologiy.html (дата обращения: 02.02.2021).

5. **Вишняков А. С., Макаров А. Е., Уткин А. В.** и др. Обеспечение защиты данных, представленных в облачных сервисах [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/obespechenie-zaschity-dannyh-predstavlennyh-v-oblachnyh-servisah/viewer (дата обращения: 02.02.2021).

6. **Дремач К.** Облачные сервисы: проблемы в доверии // Information Security = Информационная безопасность [Электронный ресурс]. URL: https://lib.itsec.ru/articles2/cloud-security/oblachnie-servisi-problemi-v-doverii (дата обращения: 02.02.2021).

7. **Прудникова А. А., Садовникова Т. М.** Анализ облачных сервисов с точки зрения информационной безопасности. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/analiz-oblachnyh-servisov-s-tochki-zreniya-informatsionnoy-bezopasnosti/viewer (дата обращения: 02.02.2021).

8. **Подколозина Л. А., Коваленко Д. К.** Обеспечение безопасности мобильных облачных хранилищ на основе методов классификации данных [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/obespechenie-bezopasnosti-mobilnyh-oblachnyh-hranilisch-na-osnove-metodov-klassifikatsii-dannyh/viewer (дата обращения: 02.02.2021).

9. **Сахаров Д. В., Левин М. В., Фостач Е. С., Виткова Л. А.** Исследование механизмов обеспечения защищенного доступа к данным, размещенным в облачной инфраструктуре // Наукоемкие технологии в космических исследованиях Земли. 2017. Т. 9, № 2. С. 40—46.

10. **Алексеев А. П., Орлов В. В.** Сокрытие сообщений путем их распыления в пространстве // Инфокоммуникационные технологии. 2008. Т.6, N. 3. С. 52—56.

11. **Алексеев А. П., Макаров М. И.** Принципы многоуровневой защиты информации // Инфокоммуникационные технологии. 2012. Т. 10, № 2. С. 88—93.

12. **Алексеев А. П.** Многоуровневая защита информации. Самара: ПГУ-ТИ-ИУНЛ, 2017. 128 с.

13. **Безбогов А. А., Яковлев А. В., Шамкин В. Н.** Методы и средства защиты компьютерной информации: Учеб. пособ. Тамбов: Издательство ТГТУ, 2006. 196 с.

14. **Лыгин Е. А.** Тайнопись. Практическое пособие по ручному шифрованию. Саратов: Новый ветер, 2010. 206 с.

15. **Виды** и способы криптографических преобразований [Электронный ресурс]. URL: https://www.securitylab.ru/blog/personal/aguryanov/29980.php (дата обращения: 01.02.2021).

16. **Древние** информационные системы: защита информации [Электронный ресурс]. URL: https://zen.yandex.ru/media/info_law_society/drevnie-informacionnye-sistemy-zascita-informacii-5c83d8eae2ca0400b31c154f (дата обращения: 02.02.2021).

17. **Data** distribution methods and systems [Электронный ресурс]. URL: http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&s1=%2220150293986%22.PGNR.&OS=DN/20150293986&RS=DN/20150293986 (дата обращения: 02.02.2021).

18. **Data** distribution methods and systems [Электронный ресурс]. URL: https://patents.google.com/patent/US10216822B2/en?oq = US + 10216822B2 (дата обращения: 02.02.2021).

L. V. Arshinskiy, Professor, e-mail: larsh@mail.ru,
G. N. Shurkhovetsky, Postgraduate Student, e-mail: gshn5@yandex.ru,
Irkutsk State Transport University, Irkutsk, Russian Federation

Features Application of the Dissection-Placing Method for Secure Data Storage in External Data Warehouses

The article considers the application of the dissection-placing method for secure storage of information in external, primarily cloud-based data warehouses. Various approaches to the implementation of the method, including patent material, are analyzed. It is shown that the method is most effective for bitwise dissection of files and random placement of bits in data streams sent to separate warehouses.

Keywords: *information security, data storage, cloud computing, dissection-placing method*

DOI: 10.17587/it.27.259-266

References

1. **Stupina M. V., Shpakov D. V.** On the issue of cloud security, *Young Scientist*, 2016, no. 9 (113), pp. 85—88 (in Russian).

2. **Cloud Security:** a look at the Russian market, *Information Security*, https://lib.itsec.ru/articles2/cloud-security/cloud-security-vzglyad-na-rossiiskii-rinok (date of access: 01/02/2021) (in Russian).

3. **Dovgal V. A.** Methods of safety increase in the sphere of "cloud" technologies, *The Bulletin of the Adyghe State University*, 2014, no. 4(147), pp. 170—174 (in Russian).

4. **Shpakov D. V.** Study of the problem of development and security of cloud technologies, https://docplayer.ru/57291877-Izuchenie-problemy-razvitiya-i-bezopasnosti-oblachnyh-tehnologiy.html (date of access: 02.02.2021) (in Russian).

5. **Vishnyakov A. S., Makarov A. E., Utkin A. V.** et. al. Ensuring the protection of data presented in cloud services, available at: https://cyberleninka.ru/article/n/obespechenie-zaschity-dannyh-predstavlennyh-v-oblachnyh-servisah/viewer (date of access: 02.02.2021) (in Russian).

6. **Dremach K.** Cloud services: Problems in trust, *Information Security,* available at: https://lib.itsec.ru/articles2/cloud-security/oblachnie-servisi-problemi-v-doverii (date of access: 02.02.2021) (in Russian).

7. **Prudnikova A. A., Sadovnikova T. M.** Analysis of cloud services from the point of view of information security, available at: https://cyberleninka.ru/article/n/analiz-oblachnyh-servisov-s-tochki-zreniya-informatsionnoy-bezopasnosti/viewer (date of access: 02.02.2021) (in Russian).

8. **Podkolozina L. A., Kovalenko D. K.** Ensuring the security of mobile cloud warehouses based on data classification methods, available at: https://cyberleninka.ru/article/n/obespechenie-bezopasnosti-mobilnyh-oblachnyh-hranilisch-na-osnove-metodov-klassifikatsii-dannyh/viewer (date of access: 02.02.2021) (in Russian).

9. **Sakharov D. V., Levin M. V., Fostach E. S., Vitkova L. A.** Research of mechanisms of the protected access problem to cloud data storage, *H&ES Research*, 2017, vol. 9, no. 2, pp. 40—46 (in Russian).

10. **Alekseev A. P., Orlov V. V.** Hiding messages by scattering them in space, *Infocommunication Technologies*, 2008, vol. 6, no. 3, pp. 52—56 (in Russian).

11. **Alekseev A. P., Makarov M. I.** Principles of multilevel protection of the information, *Infocommunication Technologies*, 2012, vol. 10, no. 2, pp. 88—93 (in Russian).

12. **Alekseev A. P.** Multilevel protection of the information. Samara, PGU-TI-IUNL, 2017, 128 p. (in Russian).

13. **Bezbogov A. A., Yakovlev A. V., Shamkin V. N.** Methods and means of protection of computer information systems: textbook. Tambov, TSU, 2006. 196 p. (in Russian).

14. **Lygin E. A.** Secret writing. A practical guide to manual encryption. Saratov, Novyy veter, 2010, 206 p. (in Russian).

15. **Types** and methods of cryptographic transformations, available at: https://www.securitylab.ru/blog/personal/aguryanov/29980.php (date of access: 01.02.2021) (in Russian).

16. **Ancient** information systems: Information security, available at: https://zen.yandex.ru/media/info_law_society/drevnie-informacionnye-sistemy-zascita-informacii-5c83d8eae2ca0400b-31c154f (date of access: 01.02.2021) (in Russian).

17. **Data** distribution methods and systems, available at: http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.html&r=1&f=G&l=50&s1=%2220150293986%22.PGNR.&OS=DN/20150293986&RS=DN/20150293986(date of access: 02/02/2021).

18. **Data** distribution methods and systems, available at: https://patents.google.com/patent/US10216822B2/en?oq = US + 10216822B2 (date of access: 02/02/2021).

УДК 004.822 DOI: 10.17587/it.27.267-274

С. М. Салибекян, канд. техн. наук, e-mail: ssalibekyan@hse.ru,
Национальный исследовательский университет "Высшая школа экономики",
Московский институт электроники и математики, г. Москва

Объектно-атрибутный подход для семантического анализа естественного языка*

Описывается методика семантического анализа естественного языка (ЕЯ) и семантического поиска в нем, включающая в себя: основные этапы анализа ЕЯ, формат семантической сети для представления смысла текста, работу с полисемией (многозначностью) слов, семантико-синтаксическое согласование слов и т. д. Методика основывается на применении объектно-атрибутного принципа организации вычислений и структур данных, относящегося к классу dataflow (вычислительные системы с управлением данными).

***Ключевые слова**: семантический анализ естественного языка, семантическая сеть, граф-трансформирующая система, вычислительная система с управлением потоком данных, семантико-синтаксический анализ естественного языка*

Введение

Предметом рассмотрения данной статьи является методика семантического анализа естественного языка (ЕЯ) с помощью объектно-атрибутного (ОА) подхода к организации вычислительного процесса, разрабатываемая с 2013 г. Проблематика семантического анализа ЕЯ становится все более актуальной: объем информации в мире увеличивается экспоненциальными темпами, а широко применяемый сейчас поиск по ключевым фразам (или частотный анализ текста) уже не в состоянии дать релевантные (соответствующие замыслу пользователя) результаты; требуются все бóльшие объемы автоматического перевода иностранных текстов, но повсеместно используемые сейчас статические алгоритмы не обеспечивают необходимого качества; все в бóльших объемах требуется автоматическое реферирование текстов и т. д. Однако реализация подобных систем связана с большими трудностями, такими как многозначность (полисемия) слов ЕЯ, сложные синтаксические и семантические конструкции, необходимость применения многостадийного анализа и т. д. За всю историю компьютерной лингвистики было предложено достаточно много методик семантического анализа ЕЯ: генеративные грамматики Хомского, теория "Смысл <-> текст" Игоря Мельчука, концепция "семантический Web" и т.д. Хороший обзор методов синтаксического и семантического анализа ЕЯ приведен в работах [1, 2]. Появляются различные программные продукты, из которых особо следует отметить ABBYY Compreno [3], осуществляющий глубинный синтаксический и семантический анализы ЕЯ. Однако до сих пор нет общепризнанной методики семантического анализа ЕЯ, где на всех этапах разбора языка применяется единый подход к анализу и единые форматы данных. Одна из причин этого — разделение анализа синтаксиса и семантики ЕЯ, а ведь на практике было доказано, что без учета семантики невозможно провести полноценный синтаксический анализ, и ошибки, допущенные на изолированном этапе синтаксического анализа, затем вносят искажения на этапе семантического анализа. Еще одной причиной является тот факт, что получаемые в процессе синтеза итоговые информационные конструкции имеют ограничения по топологии и информационному содержанию (т. е. ограничения в описании смыслов). Например, генеративная грамматика Н. Хомского, основы которой изложены в работе [4], получила на Западе большую популярность. Однако оказалось, что она не очень хорошо подходит для задач анализа ЕЯ. Причины тому следующие: семантическая конструкция языка представляется в виде дерева (а не семантической сети), а такая структура не может описывать все многообразие семантических конструкций ЕЯ; существуют ограничения в описании смысла текста (узлы и дуги графа-дерева, описывающего синтаксис и семантику предложения, помечаются только метками, но не могут содержать структурированную информацию). По этой же причине оказались неэффективными и другие виды грамматик: грамматик-

*Работа выполнена при финансовой поддержке гранта РФФИ в рамках научного проекта №20-07-00958, реализуемого проектной группой "Децентрализованные данные" НИУ ВШЭ.

ки зависимостей, грамматика непосредственно составляющих, категориальная грамматика, вершинная грамматика составляющих (Head-driven Phrase Structure Grammar, HPSG) [5], лексико-функциональная грамматика (Lexical Functional Grammar, LFG), вероятностная контекстно-свободная грамматика (Probabalistik ContextFree Grammar, PCFG) и многие другие. Не подтвердила свою эффективность и LinkGrammar [6], являющаяся дальнейшим развитием генеративной грамматики, хотя данной тематике было посвящено множество исследований. Единственная грамматика, работающая со сложным информационным содержанием семантической сети — это атрибутная транслирующая грамматика Н. Хомского (АТ-грамматика) [7]. В отличие от обычной грамматики к символам языка приписывается один или несколько атрибутов. Атрибутом может быть: символ, число, строка и т. д. Грамматика описывает преобразование не только строки символов, но и информационного содержания символов. Однако АТ-грамматика применяется в основном для семантического анализа ЕЯ, так как она не в состоянии проводить синтез семантической сети, описывающей смысл текста.

Применение объектно-ориентированного (ОО) программирования (ООП) для распознания ЕЯ не дало ожидаемого эффекта. Так, ООП не нашла широкого применения даже в области баз данных, не говоря уже о базах знаний (надежды на то, что ООП составит конкуренцию реляционной парадигме, сошли на нет в 1990-х гг.) [8]. Причина тому — громоздкость и негибкость ООП [9]: класс как структура данных создается заранее и не изменяется во время вычислительного процесса. ООП успешно применяется при представлении статической семантической сети, например ОО-язык OWL, используемый в Semantics Web. Семантический же анализ предполагает синтез семантической сети с заранее неизвестной структурой.

Для анализа ЕЯ также применяются и методы машинного обучения, например, кластеризация, нейронные сети и т. д. Один из подходов к машинному обучению — это вероятностная комбинаторная категориальная грамматика (Probabilisti Combinatorial Categorial Grammars, CCG) [10], которая обеспечивает обучение вычислительной системы для задач кластеризации предложений ЕЯ. Однако такие методы неспособны осуществить полный семантический анализ, и ограничиваются только поверхностным — выделением сегментов текста (группы слов, связанной по смыслу глаголом или другим языковым элементом). Еще один недостаток таких методик анализа — необходимость обучения на больших обучающих выборках.

Достаточно удачной в области семантического анализа ЕЯ можно считать теорию "Смысл <-> текст" И. Мельчука. Теория [11, 12] предполагает анализ текста на нескольких уровнях, с постепенным переходом от одного уровня к другому: фонологический (уровень текста), поверхностно-морфологический, глубинно-морфологический, поверхностно-синтаксический, глубинно-синтаксический и семантический (уровень смысла). Морфологические и фонологические конструкции языка представляются с помощью линейных конструкций, синтаксическая — в виде дерева, остальные — виде семантической сети. Особенностью этой методики является то, что она объединяет в себе все стадии анализа языка от синтаксического до семантического. Недостаток такого подхода состоит в том, что на каждом уровне анализа конструкций языка используются свои форматы представления семантической сети, что вызывает необходимость применения сложных алгоритмов преобразования форматов представления информации при переходе с одного уровня на другой.

Теория "Смысл <-> текст" относится к классу систем, которые производят синтез семантической сети, описывающей смысл, заложенный в тексте. Это достаточно перспективный подход, поскольку такую сеть можно применять затем для различных целей: семантического поиска информации, трансляции на другой ЕЯ и т. д. В настоящем исследовании предлагается использование ОА подхода к организации структуры данных и организации вычислительного процесса [13], который также нацелен на синтез семантической сети (или ОА графа в терминологии ОА) в качестве результата анализа текста. Подход обеспечивает выполнение всех стадий анализа ЕЯ: представление правил анализа ЕЯ, построение семантической сети и поиск информации в ней, процесс преобразования текста в семантическую сеть, а также использование специализированного языка программирования для описания алгоритма преобразования и т. д. Благодаря тому, что все уровни анализа ЕЯ работают по единым принципам, значительно повышается эффективность системы в целом, так как сокращается расход вычислительных мощностей на преобразование формата данных при переходе от одного уровня к другому. ОА подход, относящийся к классу dataflow (вычисления с управлением потоком данных) [14, 15], обладает массой положительных качеств: объектный принцип построения программы и данных, способность синтеза и модификации семантической сети непосредственно во время вычислительного процесса, удобство организации параллельных и распределенных вычислений и т. д. Данные качества позволяют эффективно реализовывать системы семантического анализа ЕЯ. Еще одним преимуществом ОА подхода можно считать разработанный для него формализм, описанный в работе [16].

Итак, целью настоящего исследования является разработка методики семантического анализа текста на ЕЯ на базе ОА подхода к организации вычислительного процесса и структур данных. Отличительной особенностью такой системы являет-

ся единообразие представления данных и методики их обработки на всех стадиях анализа текста. Задачами исследования является разработка:

- форматов данных и методов их обработки;
- методики представления информации в семантической сети;
- методики анализа полисемических (многозначных) слов,
  а также
- выделение стадий анализа ЕЯ;
- выделение основных информационных примитивов и конструкций и их формализация для них;
- программная реализация прототипа системы анализа ЕЯ.

1. Синтез семантической сети

Данные на всех уровнях анализа представляют собой ОА граф. Эта конструкция несколько напоминает фреймовую структуру данных [17] или классовую структуру в ООП. Единицей данных является информационная пара (ИП), представляющая собой двойку $c = <a,l>$, где a — атрибут, представляющий собой идентификатор операнда (нагрузки); l — нагрузка (данные или указатель). ИП похожа на слот во фрейме. ИП собираются во множество, называемое информационной капсулой (ИК). Это — аналог фрейма. В нагрузках ИП могут располагаться указатели на другие ИК, и таким образом ИК могут объединяться в семантическую сеть, где каждая ИК является ее вершиной, а ссылки в нагрузках ИП выполняют роль семантических связей. В самом начале анализа текст представляется в виде ОА графа, представляющего собой список толкований слов, а далее в процессе многостадийного анализа список трансформируется в семантическую сеть, хранящую смысл текста. Преобразования ОА графа осуществляются с помощью граф-трансформирующей системы [18], а правила трансформации описываются посредством нотации ОА грамматики [19], специально разработанной для описания преобразований структуры и информационного содержания ОА графа. Граф-трансформирующая система, основанная на ОА подходе, осуществляет процесс синтеза семантической сети на всех уровнях (этапах) семантико-синтаксического анализа ЕЯ, что обеспечивает однородность анализа ЕЯ без необходимости преобразования данных при переходе от одного уровня анализа к другому.

Основой системы анализа ЕЯ на базе ОА подхода является семантико-морфологический словарь, хранящий описания лексем (слов). Описание одной лексемы представляет собой ОА список [20] всех возможных ее толкований, ведь, лексемы ЕЯ обладают свойством полисемии (многозначности). Каждое толкование лексемы — это совокупность трех связанных ссылками ИК: ИК с описанием морфологических свойств толкования лексемы (падеж, род, число и т. п.), ИК семантических свойств толкования лексемы и ИК с признаками для семантического согласования толкований слов (именно благодаря такому согласованию происходит выбор нужного толкования полисемичного слова исходя из его контекста). Семантическое согласование слов включает основные признаки для согласования: "контейнер", "вместилище", "абстрактный объект", "физический объект" и т. д. Семантические признаки толкования слова могут включить в себя и различные семантические связи с другими объектами.

Во время анализа текста осуществляется поиск описаний лексем в этом словаре, и из них формируется список толкований слов анализируемого текста (рис. 1, см. вторую сторону обложки): в ИК, содержащей атрибут POS (Part Of Spech — часть речи), находится перечисление синтаксических свойств толкования слова, а в нагрузке ИП с атрибутом SemProp (семантические свойства) находится указатель на ИП с описанием семантических свойств толкования слова. Наиболее оптимально формировать и затем осуществлять семантико-синтаксический анализ одного предложения текста, затем формировать список толкований для следующего предложения текста, анализировать его и т. д. После формирования списка толкований слов происходит многостадийное преобразование списка в семантическую сеть (семантический ОА граф), описывающую смысл текста (ОА граф текста). Эту семантическую сеть впоследствии можно будет использовать для поиска необходимой информации, автоматического перевода, автоматического реферирования и т. п. Преобразование в семантическую сеть происходит в несколько этапов (стадий), на каждом из которых осуществляется анализ определенной части речи, синтаксической или семантической конструкции — таким образом, подобный анализ ЕЯ можно назвать семантико-синтаксическим. Для русского и английского языков нами было выделено порядка 20 этапов. Анализ происходит от простого к сложному: сначала анализируются самые зависимые части речи (для русского и английского языков — это наречие степени), после анализа частей речи начинается анализ синтаксических и семантических конструкций. Приведем более подробное описание процесса анализа ЕЯ при ОА подходе.

На каждом этапе анализа происходит поиск второстепенных лексем определенного типа, а затем выполняется их "склейка" с близлежащим главным словом. Операция склейки подразумевает, что семантические свойства зависимого слова (синтаксической конструкции) присоединяются к семантическому описанию главного слова (синтаксической конструкции), а описание толкования зависимого слова удаляется из списка толкований слов. Так, на первом проходе анализа ЕЯ осуществляется "склейка" наречий степени с качественными наречиями и глаголами. Например,

для словосочетания "очень зеленый" в ИК с семантическим описанием слова "зеленый" добавится информационная пара (ИП) с атрибутом "степень" и нагрузкой, в которой будет храниться описание свойства степени "зелености" (для задания степени свойства можно, например, применить лингвистическую переменную [21]). Описание слова "очень" удаляется из списка описания слов. Данный процесс иллюстрируется на рис. 1 (см. вторую сторону обложки), где применяются обозначения: POS (Part Of Spech) — атрибут части речи), SemProp — атрибут семантических свойств толкования слова, DEGREE — атрибут степени, ADVERB_DERGREE — обозначение наречия степени, ADJECT — обозначение прилагательного.

2. ОА-грамматика

Для описания преобразований ОА графа была разработана нотация ОА грамматики. Формально ОА грамматика представляет собой четверку: OAG = {*A,L,P,G*}, где *A* — алфавит атрибутов; *L* — алфавит нагрузок ИП (алфавит включает в себя не только числа и строки, но и ссылки на ИК); *G* — ОА граф (список описаний лексем исходного языка); *P* — правила преобразования ОА графа (продукция). ОА грамматика, по сути, является разновидностью атрибутной графовой грамматики [18], где к вершинам графа приписываются один или несколько атрибутов. ОА грамматика же оперирует с ИП и ИК ОА графа: с помощью операций добавления ИП к ИК, удаления ИП из ИК, создания ИК, изменения атрибута или нагрузки ИП можно осуществлять модификацию как структуры, так и информационного содержания ОА графа.

Для ОА грамматики была разработана следующая нотация. Знак "=" в нотации описания правила преобразования обозначает ИП: слева от него указывается атрибут, справа — нагрузка. С помощью знаков "{"и"}" выделяется ИК. ИК или нагрузка могут быть именованы (имя в ОА вычислительной системе является мнемоникой указателя на ИК или нагрузку). В левой части правила преобразования (продукции) перед ИК может указываться тип множества ИК шаблона искомого подграфа. Если тип не указывается, то он по умолчанию устанавливается как "И". Множество "И" подразумевает, что все ИП из данной ИК шаблона поиска должны совпадать с ИП из ИК графа текста. Перечислим лишь некоторые типы ИК (их было выделено порядка 20): "ИЛИ" ("OR") — хотя бы одна ИП должна совпадать с ИП из ИК графа текста; "обратное И" ("INVERS AND") — все ИП из ИК графа-текста должны совпадать с ИП из ИК шаблона; "исключающее ИЛИ" ("XOR") — только одна ИП должна совпадать в ИК текста и ИК шаблона и т. д. С помощью аппарата множеств ИП можно, например, описывать поиск среди различных вариантов пространственно-временных отношений объектов физического мира [22].

Например, преобразование на рис. 1 (см. вторую сторону обложки) описывается с помощью продукции ОА-грамматики следующим образом:

{POS = ADVERB_DERGREE SemProp = temp} {POS = ADJECT SemProp = temp2} -> {POS = ADJECT SemProp = *temp}. (1)

Правая часть правила преобразования (1) задает шаблон искомого подграфа семантической сети, который необходимо преобразовать; после знака "->" идет описание трансформированного подграфа. Значком "=" разделяются атрибут и нагрузка ИП, фигурными скобками обозначается ИК. В данном случае описание наречия степени удаляется из списка толкований слов, а в ИК с описанием семантических свойств (адрес этого описания хранится в указателе "temp") добавляется в ИК с описанием семантических свойств прилагательного с атрибутом DERGREE (степень). Знак "*" обозначает конкатенацию (соединение) ИК, что было в нагрузке с ИК по указателю "temp". Второй проход анализа осуществляет "склейку" прилагательных с существительными. Например, при разборе выражения "очень зеленый стул" на первом проходе происходит склейка слов "очень" и "зеленый" (описание слова "очень" удаляется из списка, а его семантические свойства "склеиваются" со свойствами слова "зеленый"). После такой операции в списке остаются описания слов "зеленый" и "стул". На втором проходе осуществляется следующее преобразование: к описанию семантических свойств слова "стул" добавляется ИП с семантическими свойствами прилагательного, а именно, атрибутом "цвет" ("Color"), нагрузкой "зеленый" и степенью "очень", а описание слова "зеленый" удаляется из списка. В результате получается следующий семантический ОА граф: {POS = NOUN SemProp = {Color = GREEN DEGREE = Very}}, где GREEN — обозначение зеленого цвета, Very — обозначение степени "очень". При склейке к главному слову добавляется только ИК с описанием семантических свойств второстепенной лексемы, ИК с описанием ее морфологических свойств удаляется за ненадобностью, так как оно не понадобится для дальнейшего семантико-синтаксического анализа. На заключительных этапах анализа осуществляется разбор синтаксических конструкций с союзами, склейки существительных и глаголов и, наконец, склейка предикативных частей сложносочиненного или сложноподчиненного предложений. В конце концов в семантической сети остаются только ИК с описанием семантических свойств объектов и отношений между ними.

3. Обработка многозначности лексем

Анализ смысловых связей между предложениями в ОА системе осуществляется с применением

так называемого тематического словаря. Он представляет собой список, куда помещаются описания объектов, упоминающихся в тексте ранее. Например, при анализе первого предложения из фрагмента "В комнате стоял **стул**. **Стул** был зеленым." описание объекта "стул" передается в тематический словарь; при анализе же второго предложения описание слова "стул" будет обнаружено в тематическом словаре, и ссылка на это описание будет включена в список толкований слов. И впоследствии свойство "зелености" будет добавлено именно в описание семантических свойств того слова, которое встретилось в первом предложении. Описания объектов, попадающие в тематический словарь, через некоторое время удаляются из него, поскольку они становятся уже неактуальными по причине устаревания (читатель уже забывает о тех словах, которые он встретил в тексте достаточно давно) или иной причине. Процесс синтеза семантического графа из текста на ЕЯ приведена на рис. 2 (см. вторую сторону обложки).

ОА анализ успешно решает проблему работы с полисемией (многозначностью) слов. На рис. 3 (см. третью сторону обложки) представлен процесс обработки двух лексем с двумя толкованиями каждая. Пусть согласно правилам преобразования списка толкований слов необходимо провести "склейку" 2-го и 3-го толкований лексем (выделено штриховым прямоугольником в левой части рис. 3). Тогда происходит так называемое расщепление списка толкований. В результате образуется список из всех возможных альтернативных толкований этого фрагмента текста, содержащих все возможные комбинации толкований этих двух слов, точнее, четыре ветки с описанием двух лексем. На ветви, соответствующей склеиваемым толкованиям, осуществляется модификация списка толкований слов, согласно применяемому правилу преобразования (на рис. 3 справа внизу выделено штриховым прямоугольником). Альтернативные ветви толкований фрагментов текста могут удаляться из списка в процессе дальнейшего анализа, если обнаруживается синтаксическое или семантическое несогласование лексем в них. Таким образом, процесс преобразования списка толкований слов в семантическую сеть представляет собой процесс многократного создания и уничтожения альтернативных ветвей толкований фрагментов текста. Семантико-синтаксическое согласование представляет собой сравнение синтаксических и семантических атрибутов слов. Так, после союза "в" ("in") должно идти слово с атрибутом "вместилище" ("container"), а за союзом "на" ("Under") должно следовать слово с атрибутом "поверхность" ("surface") и т. д. В том случае, если не будет согласовано ни одно из толкований слова, текст считается несогласованным и не может быть проанализирован.

4. Семантическая сеть и методика поиска информации в ней

Теоретической основой семантического ОА графа является теория семантических падежей (валентностей) Ч. Филмора [23], в которой вводится перечень возможных типов связей между объектами, описываемыми в тексте. Так, Филлмором было выделено 9 ролей (связей), а Ю. Д. Апресяном — 25 [24], например, "объект", "субъект", "инструмент". Так, для предложения "Мальчик ударил большую змею палкой." будет синтезирован следующий ОА-граф: {**Объект** = мальчик **Субъект** = { **Объект** = змея **Свойство** = {**РазмерОтносительный** = большой}} **Действие** = ударить **Инструмент** = палка} (жирным выделены атрибуты ИП, описывающие семантические роли/валентности). Обзор основных семантических связей между объектами приведен в работе [25]. Нами во время разработки методики семантического анализа русского языка было выделено более ста различных семантических ролей (валентностей по Ю. Д. Апресяну), так как ролей, выделенных Филлмором, Апресяном и другими лингвистами, не хватало для описания семантики текста. И, скорее всего, в дальнейшем их список и еще пополнится.

Также был разработан формат семантической сети, представляющей смысл текста. Такая сеть должна, по сути, представлять собой универсальную сетевую базу данных, способную описывать реальный мир. В результате анализа была разработана структура сети, описанная в работе [22]. Сеть такого формата содержит три уровня: описания множеств объектов, описания свойств и состояний объектов и описание (в том числе и пространственно-временных) отношений объектов. В ней можно описать не только объекты, их свойства и отношения, но и множества, которые они образуют.

В ходе исследования также была разработана методика семантического поиска в тексте. В сетевых базах данных информационный поиск представляет собой поиск определенного подграфа. Для этого пользовательский запрос (вводится на ЕЯ) преобразуется в ОА граф и используется в качестве шаблона для поиска (рис. 4, см. третью сторону обложки). Назовем его шаблоном. Если в ОА графе текста находится подграф, изоморфный шаблону, то пользователю выдается сообщение об успехе поиска, причем в изоморфном подграфе должны совпадать с шаблоном не только узлы и связи, но информационное содержание в узлах.

Нами было выделено три типа вопросительных предложений которые могут выступать в качестве информационного запроса: от данных типов зависит формат представления графа-шаблона. Первый — общий вопрос, например "Мама уже ходила в магазин?". Ответ на данный вопрос может быть только "Да" или "Нет". На такой вопрос синтезируется простой граф-шаблон, и далее система проверяет,

имеется ли в графе-тексте подграф, совпадающий с шаблоном. Если хотя бы один такой подграф есть, то выдается сообщение "Да", иначе "Нет". Второй — запрос с "дыркой". "Дыркой" (рис. 4, см. третью сторону обложки) называется та часть шаблона, где подразумевается ответ на вопрос. Например, "Где Маша потеряла мячик?". Вопросительный союз "где" подразумевает локатив (т. е. место, где происходило действие). Поэтому в семантической сети шаблона на месте локатива будет находиться узел, обозначающий пустоту. Данный узел будет совпадать с любым локативом в графе текста. При нахождении подграфа ответом на вопрос как раз и будет тот локатив, который совпадает с "дыркой". Третий тип — вопрос с шаблоном ответа. Такие вопросы задают шаблон ответа (довольно часто в них применяются частицы "ли", "или", "и"), например, "Коля или Вася пошли гулять?". По результату такого запроса будет сформирован запрос с "дыркой" на месте главного действующего лица, и также будет задан шаблон ответа, представляющий собой список из двух имен. На такой вопрос может быть три варианта ответа: "Коля", "Петя" или ответ не найден. Последний вариант выдается в том случае, если в графе текста не будет найден подграф, изоморфный шаблону, или ответ на месте "дырки" не будет соответствовать шаблону.

Семантический поиск также необходим и в процессе синтеза семантической сети. Дело в том, что правая часть правила преобразования ОА графа описывает шаблон поиска подграфа, который необходимо модифицировать (правила сравнения информационного содержания узлов ОА графа описываются в левой части правила преобразования ОА графа). Поэтому наиболее рационально разделить трансформирующую граф систему на

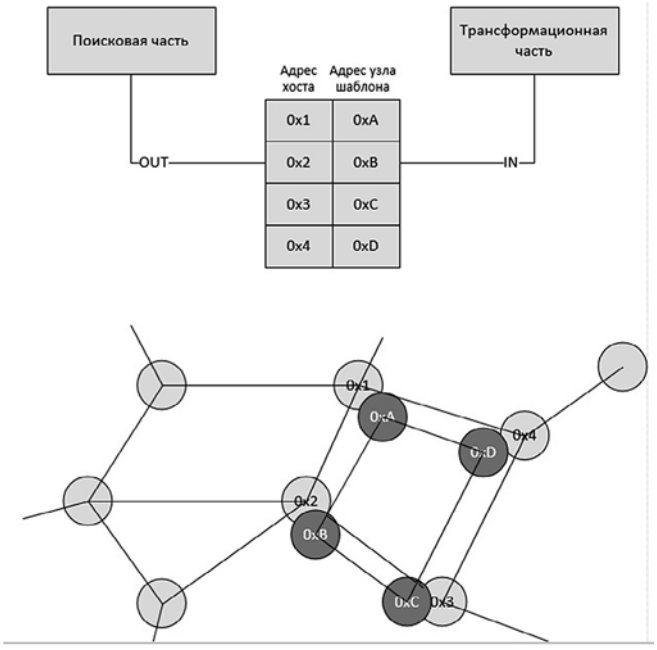


Рис. 5. Граф-трансформирующая система

две части: устройство поиска и устройство трансформации графа. Первое из них находит подграф, соответствующий шаблону из левой части правила преобразования, передает его на второе устройство (рис. 5). Но здесь возникает одна трудность: продукция описана относительно шаблона. Например, в правиле (1) ссылки temp и temp2 указывают на граф-шаблон, а трансформировать необходимо найденный подграф графа текста. Решением проблемы стало применение таблицы преобразования адресов — устройство поиска формирует ее после нахождения подграфа и передает устройству трансформации. В этой таблице прописываются соответствия адресов узлов шаблона и найденного подграфа. Во время преобразования подграфа устройство трансформации заменяет адреса узлов шаблона на адреса узлов подграфа и проводит модификацию структуры информационного содержания подграфа. Устройство трансформации графа было реализовано программно. Оно способно изменять ИП, вставлять новые ИП в ИК, удалять ИП из ИК, создавать новые ИК. Данный инструментарий дает возможность модификации любых ОА графов. Описание продукций осуществляется с помощью специализированного языка программирования (ОА язык). Посредством его команд можно задать шаблон поиска, все необходимые ссылки на ИК и ИП шаблона, а также задать действия, которые необходимо сделать в случае нахождения подграфа, соответствующего левой части продукции ОА грамматики.

Заключение

Разработанная методика ОА анализа ЕЯ реализована на практике — создана экспериментальная база знаний для анализа русского и английского языков, включающая в себя словарь из описаний порядка двух сотен слов и около ста правил преобразования списка исходных лексем. Это доказало работоспособность предложенной методики анализа ЕЯ. В дальнейших планах работы стоит расширение семантико-морфологического словаря и числа правил преобразования списка толкований слов, а также критериев семантического согласования лексем в экспериментальной базе данных. Кроме того, планируется добавление вероятностного анализа при семантическом согласовании слов.

Разработанная методика обладает однородностью представления и обработки данных на всех уровнях семантического анализа ЕЯ, что сводит к минимуму преобразования информационных конструкций при переходе от одного уровня анализа к другому. Все преобразования осуществляются с помощью одной и той же трансформирующей граф системы, обеспечивающей целостность данных при их изменении. Трансформирующая система способна изменять не только структуру семантической сети, но и ее информационное содержание. Для описания как исходных данных, так и правил преобразования

семантической сети существует такой инструмент, как ОА язык — специализированный язык программирования для описания данных и алгоритма работы ОА вычислительной сети. Сама же ОА система может быть реализована как программно, так и аппаратно. Еще одно преимущество ОА анализа ЕЯ в том, что он позволяет работать в одной парадигме буквально всем специалистам, занимающимся созданием системы семантического анализа ЕЯ: разработчикам аппаратной части, программистам, лингвистам, математикам.

В планы на будущее входят разработка алгоритма и программная реализация поиска изоморфного подграфа. Семантический поиск необходим не только для реализации пользовательских поисковых запросов, но и является неотъемлемой частью ОА трансформирующей граф системы, необходимой для осуществления левой части продукции правила ОА грамматики.

Эффективность работы создаваемой системы анализа ЕЯ требует проверки. Эту проверку можно будет провести путем статистического анализа работы системы на больших текстах (в настоящее время реализована только экспериментальная система анализа ЕЯ). Для этого будет необходимо программно реализовать алгоритм поиска изоморфного подграфа в семантической сети (в настоящее время применяются различные "заглушки", эмулирующие работу поисковой системы). Аналитические расчеты сложности алгоритма поиска показывают, что в худшем случае временная сложность алгоритма равна N^n, где N — число узлов в графе текста, n — число узлов в графе-шаблоне. Данная сложность полиномиальна при условии ограниченности n (т. е. когда поисковые запросы пользователя ограничены по объему текста, а на практике так и происходит), и поисковый алгоритм выполним на ЭВМ. Но ожидается, что на практике сложность будет намного меньше, чем данная оценка.

Результаты исследования найдут применение в области семантического анализа текста, семантического поиска, автоматического перевода и реферирования. Для автоматического перевода необходима разработка методики синтеза текста из семантической сети; однако в настоящее время исследования по данной тематике нами не проводятся.

Список литературы

1. **Смирнов И. В., Шелманов А. О.** Семантико-синтаксический анализ естественных языков часть I. Обзор методов синтаксического и семантического анализа текстов // Искусственный интеллект и принятие решений. 2013. № 1. С. 41—54.

2. **Смирнов И. В., Шелманов А. О., Кузнецова Е. С., Храмоин И. В.** Семантико-синтаксический анализ естественных языков. Часть II. Метод семантико-синтаксического анализа текстов // Искусственный интеллект и принятие решений. 2014. № 1. С. 11—24.

3. **Anisimovich K. V., Druzhkin K. Ju., Minlos F. R., Petrova M. A., Selegey V. P., Zuev K. A.** Syntactic and semantic parser based on abbyy compreno linguistic technologies, computational linguistics and intellectual technologies // Proceedings of the International Conference Dialog. 2012. P. 90—103.

4. **Chomsky N.** Three models for the description of language // IRE Transactions on Information Theory. 1956. Vol. 2, N. 3. P. 113—124.

5. **Müller S.** Unifying everything: Some remarks on simpler syntax, construction grammar, minimalism and HPSG // Language. 2013. Vol.89, N.4. P.920—950.

6. **Link** grammar. 2013. feb. URL: http://www.abisource.com/projects/link-grammar/

7. **Ахо А., Сети Р., Ульман Дж.** Компиляторы. Принципы, технологии, инструменты. М.: Издательский дом "Вильямс", 2008. С. 383—398.

8. **Дейт К. Дж.** Введение в системы баз данных. М.: Издательский дом "Вильямс", 2005.

9. **Gabriel R.** Objects Have Failed: Notes for a Debate. (retrieved 17 May 2009). URL: http://www.dreamsongs.com/Files/ObjectsHaveFailed.pdf.

10. **Luke S.** Zettlemoyer and Mi hael Collins. Learning to map sentences to logical form: Structured lassication with Probabilisti Categorial Grammars // Proc. of 21th Conf. on Uncertainty in Articial Intelligence (UAI-2005). Edinburgh, Scotland, 2005. P. 658—666.

11. **Мельчук И. А.** Опыт теории лингвистических моделей "СМЫСЛ <—>ТЕКСТ". М.: Школа "Языки русской литературы", 1999.

12. **Мельчук И. А.** Русский язык в модели "СМЫСЛ <—>ТЕКСТ". Москва-Вена: Школа "Языки русской культуры", Венский славистический альманах, 1995.

13. **Салибекян С. М., Панфилов П. Б.** Объектно-атрибутная архитектура — новый подход к созданию объектных систем // Информационные технологии. 2012. № 2. С. 8—13.

14. **Data** flow computing: theory and practice / edited by John A. Sharp. Ablex Publishing Corp. Norwood, NJ, USA, 1992. 569 c.

15. **Milutinovic V., Trifunovic N., Salom J., Giorgi R.** The guide to dataflow supercomputing. USA: Springer; 2015.

16. **Салибекян С. М., Панфилов П. Б.** Вопросы автоматно-сетевого моделирования вычислительных систем с управлением потоком данных // Информационные технологии и вычислительные системы. 2015. № 1. С. 3—9

17. **Minsky M.** A framework for representing knowledge. MIT AI Laboratory Memo 306, June, 1974.

18. **König B., Nolte D., Padberg J., Rensink A.** A tutorial on graph transformation // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics10800 LNCS. P. 83—104.

19. **Salibekyan S., Panfilov P.** Linguistic Processor Based on Object-Attribute Grammar // The 6th International Conference on Analysis of Images, Social Networks, and Texts (AIST-2017), Moscow, Russia, July 27-29, 2017, CEUR Workshop Proceedings, Vol-1975. P. 134—145. URL: http://ceur-ws.org/Vol-1975/paper15.pdf.

20. **Салибекян С. М., Панфилов П. Б.** Анализ языка с помощью объектно-атрибутного подхода к организации вычислений // Программная инженерия. 2013. № 1. С. 9—16.

21. **Заде Л.** Понятие о лингвистической переменной и его применение к принятию решений. М.: Мир, 1976.

22. **Салибекян С. М., Петрова С. Б.** Объектно-атрибутная модель представления пространственно-временных отношений между объектами // Прикладная информатика. 2016. Т. 11, № 3 (63). С. 103—115.

23. **Филлмор Ч.** Дело о падеже // Новое в.зарубежной лингвистике. 1981. Вып.X. С. 369—495.

24. **Апресян Ю. Д.** Избранные труды. Т. 1. Лексическая семантика. М.: Языки русской культуры, 1995. 472 с.

25. **Найханова Л. В.** Основные типы семантических отношений между терминами предметной области // Известия высших учебных заведений. Поволжский регион. Технические науки. 2008. № 1. С. 62—71.

S. M. Salibekyan, Assistant Professor, e-mail: ssalibekyan@hse.ru,
National Research University Higher School of Economics, Moscow, Russian Federation

Object-Attribute Approach for Semantic Analysis of Natural Language

The article describes the methodology of semantic analysis of natural language (NL) and semantic search in it, which includes: the general stages of analysis of NL, the format of the semantic network for representing the meaning of the text, words polysemy analysis, semantic and syntactic agreement of words, etc. The method is based on the object-attribute principle of organization of calculations and data structures, belonging to the dataflow class.

Keywords: *Semantic analysis of natural language, semantic network, graph-rewriting system, dataflow computing system, semantic-syntactic analysis of natural language, graph-transforming system*

Acknowledgements: This work was carried out with the financial support of the RFBR grant in the framework of the scientific project No. 20-07-00958

DOI: 10.17587/it.27.267-274

References

1. **Smirnov I. V., Shelmanov A. O.** Semantiko-sintaksicheskij analiz estestvennyh yazykov chast' I. Obzor metodov sintaksicheskogo i semanticheskogo analiza tekstov, *Iskusstvennyj Intellekt i Prinyatie Reshenij*, 2013, no. 1, pp. 41—54 (in Russian).

2. **Smirnov I. V., Shelmanov A. O., Kuznecova E. S., Hramoin I. V.** Semantiko-sintaksicheskij analiz estestvennyh yazykov. CHast' II. Metod semantiko-sintaksicheskogo analiza tekstov, *Iskusstvennyj Intellekt i Prinyatie Reshenij*, 2014, no. 1, pp. 11—24 (in Russian).

3. **Anisimovich K. V., Druzhkin K. Ju., Minlos F. R., Petrova M. A., Selegey V. P., Zuev K. A.** Syntactic and semantic parser based on ABBY Compreno linguistic technologies, computational linguistics and intellectual technologies, *Proceedings of the International Conference Dialog*, 2012, pp. 90—103.

4. **Chomsky N.** Three models for the description of language, *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113—124.

5. **Müller S.** Unifying everything: Some remarks on simpler syntax, construction grammar, minimalism and HPSG, *Language,* 2013, vol. 89, no. 4, pp.920—950.

6. **Link** grammar, 2013, feb., available at: http://www.abisource.com/projects/link-grammar/

7. **Aho A., Seti R., Ul'man J.** Kompilyatory. Principy, tekhnologii, instrument, Moscow, Vil'yams, 2008, pp. 383—398 (in Russian).

8. **Dejt K. Dzh.** Vvedenie v sistemy baz dannyh, Moscow, Vil'yams, 2005 (in Russian).

9. **Gabriel R.** Objects Have Failed: Notes for a Debate. (retrieved 17 May 2009), available at: http://www.dreamsongs.com/Files/ObjectsHaveFailed.pdf.

10. **Luke S.** Zettlemoyer and Mi hael Collins. Learning to map sentences to logical form: Structured lassication with Probabilisti Categorial Grammars, *Proc. of 21th Conf. on Uncertainty in Articial Intelligence (UAI-2005),* Edinburgh, Scotland, 2005, pp. 658—666.

11. **Mel'chuk I. A.** Opyt teorii lingvisticheskih modelej "SMYSL <—>TEKST", Moscow, Shkola "YAzyki russkoj literatury", 1999 (in Russian).

12. **Mel'chuk I. A.** Russkij yazyk v modeli "SMYSL <—>TEKST", Moscow,Vena, Shkola "Yazyki russkoj kul'tury", Venskij slavisticheskij al'manah, 1995 (in Russian).

13. **Salibekyan S. M., Panfilov P. B.** Ob"ektno-atributnaya arhitektura — novyj podhod k sozdaniyu ob"ektnyh system, *Informacionnye Tekhnologii*, 2012, no. 2, pp. 8—13 (in Russian).

14. **Data** flow computing: theory and practice, Ablex Publishing Corp. Norwood, NJ, USA, 1992, 569 p.

15. **Milutinovic V., Trifunovic N., Salom J., Giorgi R.** The guide to dataflow supercomputing, USA, Springer, 2015.

16. **Salibekyan S. M., Panfilov P. B.** Voprosy avtomatno-setevogo modelirovaniya vychislitel'nyh sistem s upravleniem potokom dannyh, *Informacionnye Tekhnologii i Vychislitel'nye Sistemy,* 2015, no. 1, pp. 3—9 (in Russian).

17. **Minsky M.** A framework for representing knowledge, MIT AI Laboratory Memo 306, June, 1974.

18. **König B., Nolte D., Padberg J., Rensink A.** A tutorial on graph transformation, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics10800 LNCS*, pp. 83—104.

19. **Salibekyan S., Panfilov P.** Linguistic Processor Based on Object-Attribute Grammar, *The 6th International Conference on Analysis of Images, Social Networks, and Texts (AIST-2017)*, Moscow, Russia, July 27—29, 2017, CEUR Workshop Proceedings, ISSN 1613-0073, Vol. 1975, pp. 134—145, available at: http://ceur-ws.org/Vol-1975/paper15.pdf (in Russian).

20. **Salibekyan S. M., Panfilov P. B.** Analiz yazyka s pomoshch'yu ob"ektno-atributnogo podhoda k organizacii vychislenij, *Programmnaya Inzheneriya*, 2013, no. 1, pp. 9—16 (in Russian).

21. **Zade L.** Ponyatie o lingvisticheskoj peremennoj i ego primenenie k prinyatiyu reshenij, Moscow, Mir, 1976 (in Russian).

22. **Salibekyan S. M., Petrova S. B.** Ob"ektno-atributnaya model' predstavleniya prostranstvenno-vremennyh otnoshenij mezhdu ob"ektami*, Prikladnaya Informatika*, 2016, vol. 11, no. 3 (63), pp. 103—115 (in Russian).

23. **Fillmor Ch.** Delo o padezhe, *Novoe v.zarubezhnoj lingvistike*, Moscow, Progress, 1981, iss. H, pp. 369—495 (in Russian).

24. **Apresyan Yu. D.** Izbrannye trudy. Vol. 1. Leksicheskaya semantika, Moscow, Yazyki russkoj kul'tury, 1995, 472 p. (in Russian).

25. **Najhanova L. V.** Osnovnye tipy semanticheskih otnoshenij mezhdu terminami predmetnoj oblasti, *Izvestiya Vysshih Uchebnyh Zavedenij. Povolzhskij Region. Tekhnicheskie Nauki,* 2008, no. 1, pp. 62—71 (in Russian).

УДК 004.415.2 DOI: 10.17587/it.27.275-280

Н. М. Куляшова, канд. физ.-мат. наук, доц., e-mail: kafivt@mail.ru,
Национальный исследовательский Мордовский государственный университет
имени Н. П. Огарева, г. Саранск

Программное обеспечение и алгоритмы распознавания адресных структур

Обсуждается проблема распознавания почтовых адресов из строки произвольного формата. Обоснована необходимость создания программного обеспечения для автоматизации распознавания адресных структур из строки произвольного формата. На основе сведений о правилах формирования почтовых адресов проанализированы их основные закономерности и нюансы.

В качестве подхода к разработке программного обеспечения в виде программной библиотеки и веб-сервера с открытым интерфейсом взаимодействия по технологии HTTP REST API для автоматического распознавания адресов была использована парадигма объектно-ориентированного программирования. Предпочтение отдается языку программирования C# с использованием платформы ASP.NET Core 3.1 и Entity Framework Core 3.1.

Разработаны принципиально новые алгоритмы распознавания почтовых адресов на языке программирования C#.

Ключевые слова: *распознавание адресов, программное обеспечение, адресные структуры, алгоритмы, сравнение строк, ASP.NET Core, Entity framework, Waterfall*

Введение

В современном мире множество автоматизированных информационных систем встречается с проблемой распознавания почтового адреса и его структур из строки, вводимой пользователем в произвольном формате [1, 4]. Задачу можно сформулировать следующим образом: автоматизировать деятельность почтальона, который, прочитав адрес на почтовом конверте, мысленно разбивает его на составляющие и определяет, в какой местности располагается адресат, основываясь на специфике формирования, вложенности административно-территориальных единиц и своем профессиональном опыте.

Потребность в распознавании почтового адреса может возникнуть в автоматизированных курьерских или почтовых системах, диспетчерских или логистических программных продуктах. Заметим, что ошибки в таком программном обеспечении (ПО) для некоторых организаций (например, оказывающих экстренную и неотложную помощь) могут быть критическими.

На практике при детальном рассмотрении правил формирования адресов и прочих внешних факторов становится понятно, что разработка такого программного приложения требует привлечения большого количества ресурсов, и это существенно превышает выгоду от использования решения, особенно для небольших проектов. Поэтому в настоящий момент успешно развиваются online-сервисы, специализирующиеся на обработке адресов и предлагающие свои услуги на рынке программных продуктов с многочисленными ограничениями, однако все они либо платные, либо не свободны в использовании [2, с. 356].

Актуальность работы состоит в необходимости создания универсального программного обеспечения для распознавания адресных структур.

1. Материалы и методы

В качестве подхода к решению выберем доминирующую в настоящее время парадигму объектно-ориентированного программирования. Это значит, что все сущности в программной системе будут представлены в виде объектов. Для поддержки большинства БД лучше всего использовать одну из технологий ORM (Object-Relational Mapping) [9].

С учетом вышеперечисленных особенностей предпочтение отдается объектно-ориентированному языку программирования C# версии 8.0 с использованием платформы ASP.NET Core 3.1 и Entity Framework Core 3.1 [3, с. 133]. Эти технологии разрабатываются и поддерживаются корпорацией Microsoft, что обеспечивает их совместимость не только в настоящий момент, но и в долгосрочном будущем.

2. Постановка задачи

Задача заключается в разработке веб-приложения по распознаванию адресных структур, удовлетворяющего следующим требованиям:

- распознавание адресных структур из строки произвольного формата и выдача всех вариантов распознавания за разумное время (менее 3 с);
- использование эталонного справочника (например, ФИАС);

- возможность обновления адресной базы, поддержка актуальности адресных объектов;
- точность распознавания адреса, записанного в корректном формате, не менее 90 %;
- возможность использования программного обеспечения как в виде программной библиотеки, так и в виде отдельно настраиваемого веб-сервера;
- кросс-платформенность.

3. Основные компоненты программного обеспечения

Для начала рассмотрим модель разрабатываемой программной библиотеки (рис. 1).

Библиотека представляет собой монолитный блок, имеющий свойства "черного ящика". Использование программного обеспечения происходит в ряд этапов:

1. Загрузка массива адресных объектов.
2. Непосредственное использование библиотеки для распознавания адресов.
3. Обновление массива адресных структур.
4. Завершение работы со словарем.

Существует несколько ограничений в использовании ПО для распознавания адресов в виде библиотеки, особенно в проектах, написанных с использованием иных информационных технологий [10], поэтому решено дополнительно создать программное обеспечение (обертку), позволяющее развернуть отдельный веб-сервер с предоставлением открытого HTTP REST API.

Рис. 1. Использование программной библиотеки

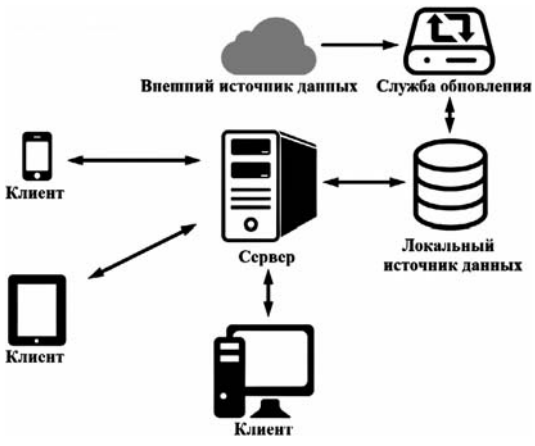


Рис. 2. Организация взаимосвязей компонентов ПО верхнего уровня

С таким ПО смогут взаимодействовать самые требовательные по ресурсам программы, например, приложения для микроконтроллеров или умных часов.

В силу вышесказанного можно выделить следующие компоненты верхнего уровня взаимодействия такой программной системы (рис. 2):

- Клиент. ПО, обращающееся к серверу для распознавания адресов.
- Сервер. Осуществляет обработку входящих запросов от клиентов и возвращает результаты распознавания. Взаимодействует с локальным источником данных для получения адресных объектов по мере необходимости.
- Локальный источник данных. БД или иная технология (файл, облако), обеспечивающая доступ к адресным объектам и их постоянное хранение.
- Служба обновления. Отдельное приложение для обеспечения актуальности адресных объектов.
- Внешний источник данных. Сервер данных Федеральной информационной адресной системы (ФИАС), располагающийся по адресу: https://fias.nalog.ru/Updates [7].

4. Архитектура программного обеспечения

На рис. 3 представлена архитектура приложения, в основе которой лежит классическая трехзвенная модель разработки приложений [8].

Ниже представлено подробное описание каждой отдельной сборки:

1. Domain. Содержит основные модели и интерфейсы для взаимодействия с ними.
2. Data. Содержит реализацию слоя доступа к данным.
3. Algorithms. Программная библиотека с разработанными алгоритмами.
4. Services. Содержит реализацию слоя бизнес-логики ПО.
5. Import.FIAS. Отдельная сборка, предоставляющая методы для работы с внешним источником данных ФИАС.

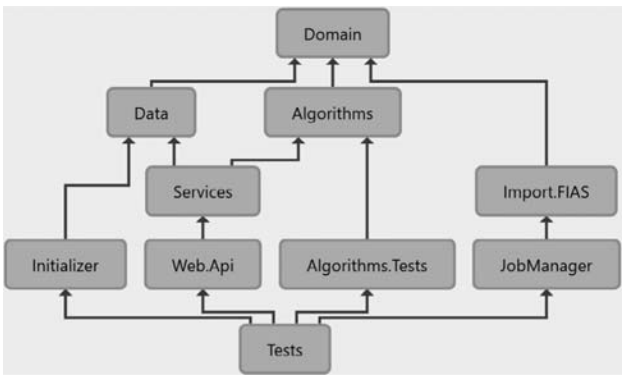


Рис. 3. Архитектура приложения

6. JobManager. Служба обновления.

7. Web.Api. Приложение-сервер с предоставлением HTTP REST API.

8. Initializer. Консольное приложение, позволяющее осуществлять миграции базы данных.

9. Algorithms.Tests. Модульные и интеграционные тесты для сборки Algorithms.

10. Tests. Модульные и интеграционные тесты для других сборок ПО.

```
public interface IAddressObject {
    Guid AoId { get; }
    Guid AoGuid { get; }
    AoLevel AoLevel { get; }
    string ShortName { get; }
    string FormalName { get; }
    Guid? ParentGuid { get; }
    string? RegionCode { get; }
    Guid? PrevId { get; }
    Guid? NextId { get; }
}
```

```
public class AddressInputData {
    public string AddressText { get; set; }
    public double? TextSimilarity { get; set; }
    public RecognitionType? RecognitionType { get; set; }
    public Guid? ParentGuid { get; set; }
}
```

```
public interface IHouse {
    Guid HouseId { get; }
    Guid HouseGuid { get; }
    Guid ParentGuid { get; }
    string? HouseNumber { get; }
    string? CorpusNumber { get; }
    string? StructureNumber { get; }
    string? RegionCode { get; }
}
```

```
public class AddressRecognitionResult {
    public AddressInputData AddressInputData { get; set; }
    public List<AddressRecognitionResultItem> Items { get; set; }
}
```

```
public class AddressRecognitionResultItem {
    public double Percent { get; set; }
    public List<IAddressObject> AddressObjects { get; set; }
    public IHouse House { get; set; }
    public string RecognizedAddress { get; set; }
}
```

Рис. 4. Интерфейсы IAddressObject и IHouse и классы AddressInputData, AddressRecognitionResult и AddressRecognitionResultItem

5. Характеристика объектов для взаимодействия с библиотекой

Ключевой сборкой в рассматриваемой схеме является программная библиотека Algorithms. Именно в ней содержится вся необходимая функциональность для распознавания адресных объектов. Сборка Algorithms зависит только от сборки Domain, поэтому последнюю также необходимо импортировать для поддержания работоспособности сборки Algorithms.

Таблица 1

Описание свойств интерфейса IAddressObject

Свойство	Описание
AoId	Уникальный идентификатор адресного объекта
AoGuid	Идентификатор адресного объекта с учетом переименований
AoLevel	Перечисление, обозначающее уровень адресной структуры: 1 — Субъект РФ, 3 — Район 4 — Город 5 — Внутригородская территория 6 — Населенный пункт 7 — Уровень планировочной структуры (улицы, проспекты и т.д.)
ShortName	Наименование префикса адресной структуры
FormalName	Наименование адресной структуры
ParentGuid	Уникальный идентификатор "родительской" адресной структуры
RegionCode	Уникальный код "родительского" субъекта РФ
PrevId	Уникальный идентификатор адресного объекта, который имеет предыдущее наименование в исторической записи
NextId	Уникальный идентификатор адресного объекта, который имеет следующее наименование в исторической записи

Базовыми элементами при работе с программной библиотекой являются модели адресного объекта, здания, входных данных, результата распознавания и элементов результата распознавания. На языке C# эти модели описываются интерфейсами IAddressObject и IHouse и классами AddressInputData, AddressRecognitionResult и AddressRecognitionResultItem соответственно. Определения сущностей и их свойств представлены на рис. 4 и в табл. 1—5.

Структура хранения адресных объектов. Перейдем к описанию алгоритма работы программной библиотеки. Одной из ключевых особенностей распознавания адресов является наиболее удобное представление адресных объектов и взаимосвязей между ними. Лучшим решением является организация адресных объектов в виде дерева.

Листьями дерева выступают объекты-дома как самые мелкие адресные единицы, а корневой вершиной является адресный объект "Российская Федерация". Любой маршрут с началом в корневой вершине и окончанием в любом листе представляет собой совокупность вложенных друг в друга адресных структур. Если соединить наименования адресных объектов маршрута, получится правильно записанный адрес какого-либо здания, в порядке убывания административных единиц.

Таблица 2

Описание свойств интерфейса IHouse

Свойство	Описание
HouseId	Идентификатор здания с учетом переименований
HouseGuid	Уникальный идентификатор здания
ParentGuid	Уникальный идентификатор "родительской" адресной структуры
HouseNumber	Номер дома
CorpusNumber	Номер корпуса
StructureNumber	Номер строения
RegionCode	Уникальный код "родительского" субъекта РФ

Таблица 3

Описание свойств класса AddressRecognitionResult

Свойство	Описание
AddressInputData	Объект AddressInputData, переданный на вход распознавания
Items	Всевозможные результаты распознавания адреса, отсортированные по проценту точности и однозначности

Таблица 4

Описание свойств класса AddressInputData

Свойство	Описание
AddressText	Строка для распознавания адреса
TextSimilarity	Допустимый процент схожести строк между собой. Учитывается в обработке ошибок и опечаток. Варьируется в пределах от 0 до 100
RecognitionType	Перечисление, определяющее тип алгоритма для распознавания адреса. Если поле не проставлено, библиотека выберет наиболее подходящий алгоритм
ParentGuid	Уникальный идентификатор адресной структуры для распознавания в пределах конкретной адресной структуры. Используется, если потребителю библиотеки достоверно известно, что адрес точно находится в пределах указанного адресного объекта. Это экономит вычислительные ресурсы и ускоряет процесс распознавания, поскольку использует только часть поддерева

Таблица 5

Описание свойств класса AddressRecognitionResultItem

Свойство	Описание
Percent	Процент точности и однозначности распознавания адреса
AddressObjects	Массив объектов, записанный в порядке убывания административных единиц, формирующий результат распознавания
House	Объект здания, соответствующий распознанному объекту
RecognizedAddress	Распознанный адрес, записанный в формате, рекомендуемом Почтой России

Рис. 5. Пример дерева адресных объектов

Если в одном узле дерева содержатся несколько адресных объектов, это всегда означает, что объект содержит несколько наименований. В реальной жизни адресные объекты постоянно изменяются, в том числе и переименовываются. Пример дерева с переименованиями представлен на рис. 5.

Чаще всего это происходит с относительно мелкими административными единицами, например, улицами и районами городов. Поэтому у одного адресного объекта может быть несколько адресов, но программному обеспечению необходимо постоянно учитывать такие исторические изменения, ведь они могут происходить и в будущем. Именно поэтому каждую неделю справочник ФИАС обновляется, чтобы сведения были максимально актуальными.

Также в технической реализации в каждом узле дерева находится гораздо большее количество информации, чем просто сведения из объекта IAddressObject. Здесь содержатся данные о наследниках и родителях и прочая информация для обновления дерева. Само дерево не является классической структурой данных в традиционном понимании. Оно состоит не только из узлов и взаимосвязей, сюда включается кеш по последним использованным веткам и добавлена организация быстрого доступа к каждому элементу трудоемкостью O(1), что позволяет не обходить дерево от корня всякий раз при поиске совпадения. Дополнительно настроена оптимизация для поиска вхождений подстрок в наименования адресных структур в узлах дерева.

Модификации распознавания адреса. Непосредственно перед процессом распознавания адреса программной библиотеке могут быть переданы параметры, изменяющие поведение алгоритмов. На основе этих параметров библиотека включает или выключает определенные модификации, описанные ниже.

Нечувствительность к регистру символов. Относительно простой функционал, который невозможно отключить. Механизм учета регистра не различает прописные и строчные символы. Это не увеличивает число ошибок распознавания.

Учет переименований адресных объектов используется только в том случае, если при загрузке словаря адресных объектов передаются непустые свойства PrevId и NextId. При построении дерева адресных объектов переименования будут добавлены в соответствующий узел. Таким образом, поддерживается распознавание исторических наименований.

Обработка сокращений в записи адреса также призвана снизить число ошибок распознавания и включена по умолчанию без возможности отключения. Программное обеспечение

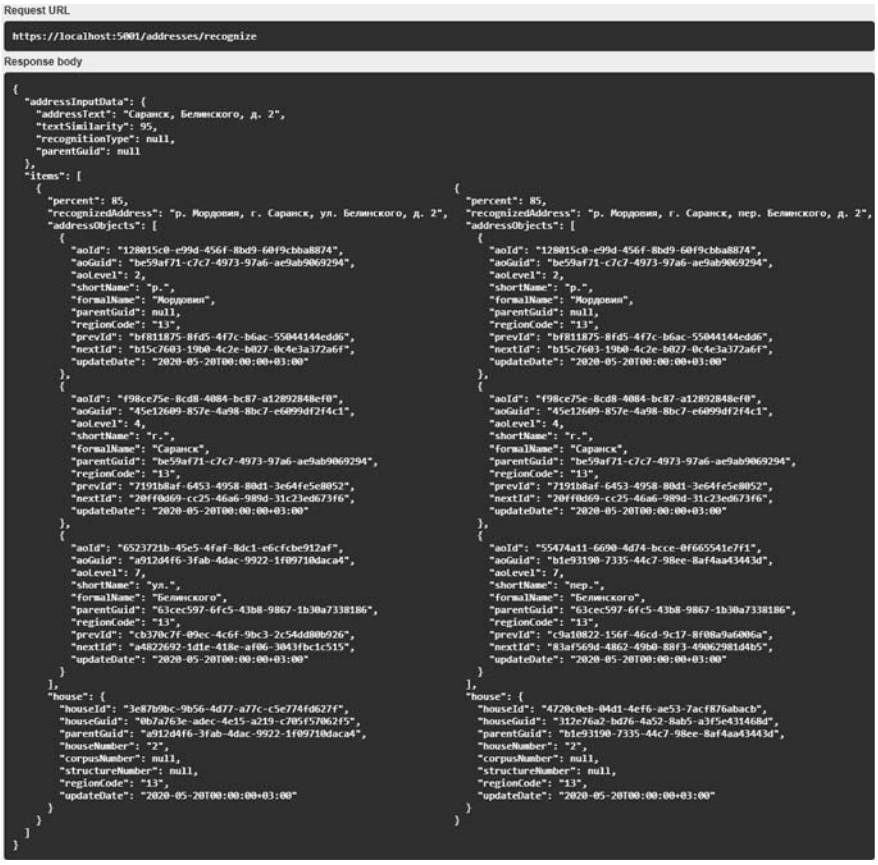


Рис. 6. Пример работы программы с неоднозначным распознаванием

обрабатывает некоторые сокращения (например, *корпус, корп., к.*) с помощью внутреннего словаря, поскольку число префиксов адресных объектов и их сокращений ограничено.

Исправление опечаток. Это могут быть и орфографические ошибки, и обычные опечатки, связанные с близким расположением букв на клавиатуре компьютера или смартфона. В основе модификации при сравнении строк лежит алгоритм вычисления редакционного расстояния Левенштейна [5].

На рис. 6 представлена работа программного обеспечения на конкретных примерах. Для визуализации взаимодействия с API используется программная библиотека Swagger.

URL-адрес распознавания прописывается в поле Request URL автоматически, с указанием метода HTTP POST. Локальный сервер слушает входящие запросы на порту 5001. В качестве параметра запроса передается объект класса AddressInputData, а возвращаемым значением является объект класса AddressRecognitionResult, который содержит входной параметр и результаты распознавания в порядке убывания точности распознавания.

URL-адрес распознавания прописывается в поле Request URL автоматически, с указанием метода HTTP POST. В данном случае локальный сервер слушает входящие запросы на порту 5001. В качестве параметра запроса передается объект класса AddressInputData, а возвращаемым значением является объект класса AddressRecognitionResult, который содержит входной параметр и результаты распознавания в порядке убывания точности распознавания.

Заключение

В работе обоснована необходимость создания программного обеспечения для автоматизации распознавания адресных структур из строки произвольного формата.

На основе сведений о правилах формирования почтовых адресов проанализированы их основные закономерности и нюансы, разработано программное обеспечение, доступное к использованию в виде программной библиотеки и в виде отдельного веб-сервера с открытым интерфейсом взаимодействия по технологии HTTP REST API.

В качестве дополнительных возможностей в программе предусмотрена работа с эталонным словарем ФИАС и его автоматические ежесуточные обновления посредством загрузки данных с официального сайта Федеральной налоговой службы.

Тестирование системы выявило, что уровень ошибок распознавания почтовых адресов составляет не более 15 % от общего числа тестовых данных. В силу специфики адресов дальнейшее снижение уровня ошибок экспоненциально увеличивает объем трудозатрат. Также было установлено, что часть ошибок распознавания может быть связана с внешними факторами (например, с отсутствием записей о некоторых адресных объектах в эталонном словаре).

Одним из приоритетных направлений развития проекта является интеграция с открытой картографической технологией OpenStreetMap для привязки географического местоположения к адресным объектам эталонного словаря.

Список литературы

1. **Бегтин И.** Сложные решения простых задач. Структуризация почтового адреса [Электронный ресурс]. URL: https:// https://old.begtin.tech/2008/07/23/сложные-решения-простых-задач-структ (дата обращения: 30.10.2020).

2. **Макаров О. С.** Обзор online-сервисов по формализации адресов // XLVII Огаревские чтения: материалы науч. конф. Саранск, 2019. С. 352—357.

3. **Макаров О. С.** Сравнение платформ.NET Core и.NET Framework // Фундаментальная и прикладная наука: состояние и тенденции развития: материалы межд. научно-практ. конф. Петрозаводск, 2019. С. 131—134.

4. **Павлюц А.** Работаем с почтовыми адресами и их качеством. Как правильно [Электронный ресурс]. URL: https://iqsystems.ru/postaddress-do-it-right/ (дата обращения: 30.10.2020).

5. **Расстояние** Левенштейна [Электронный ресурс]. URL: https://programm.top/c-sharp/algorithm/levenshtein-distance (дата обращения: 30.10.2020).

6. **Тестирование** служб и веб-приложений ASP.NET Core [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps (дата обращения: 30.10.2020).

7. **Федеральная** информационная адресная система [Электронный ресурс]. URL: https://fias.nalog.ru/Updates (дата обращения: 30.10.2020).

8. **3-Tier** Architecture: A Complete Overview [Электронный ресурс]. URL: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview (дата обращения: 30.10.2020).

9. **Micro** ORM vs ORM [Электронный ресурс]. URL: https://www.yaplex.com/blog/micro-orm-vs-orm (дата обращения: 30.10.2020).

10. **Python vs. C#:** Comparison of the Programming Languages [Электронный ресурс]. URL: https://www.netguru.com/blog/python-vs-c-sharp (дата обращения: 30.10.2020).

N. M. Kulyashova, Associate Professor, e-mail: kafivt@mail.ru,
Ogarev Mordovia State University, Saransk, 430005, Russian Federation

Algorithms and Software for Recognition of Address Structures

The article is devoted to the problem of recognizing mail addresses from an arbitrary format string. The purpose of the research is a developing software for automatic mail address recognition in two forms: as a library and as a web server.

As an approach to development, object-oriented programming paradigm was used. Preference is given to the C# programming language version 8.0 and ASP.NET Core 3.1 and Entity Framework Core 3.1.

Address recognition software has been developed. It can be used as a programming library and as a separate web server with open HTTP REST API.

The paper substantiates the need to create software for automating the recognition of address structures from a string of arbitrary format. According to the rules for the contruction of addresses, their basic patterns and nuances are analyzed. During development, fundamentally new mail address recognition algorithms were invented in the C# programming language.

Keywords*: address recognition, software, address structures, algorithms, string comparison, ASP.NET Core, Entity Framework, Waterfall*

DOI: 10.17587/it.27.275-280

References

1. **Begtin I.** Complex solutions to simple problems. Postal address structuring, available at: https:// https://old.begtin.tech/2008/07/23/сложные-решени-простых-задач-структ/ (date of access: 30.10.2020).

2. **Makarov O. C.** Review of online services for the formalization of addresses, *XLVII Ogarev readings: scientific conference materials*, Saransk, 2019, pp. 352—357 (in Russian).

3. **Makarov O. C.** Comparison between.NET Core and.NET Framework, *Fundamental and applied science: state and development trends: scientific conference materials*., Petrozavodsk, 2019, pp. 131—134 (in Russian).

4. **Pavlyuts A.** We work with postal addresses and their quality. How right, available at: https://iqsystems.ru/postaddress-do-it-right (date of access: 30.10.2020).

5. **Levenshtein** distance, available at: https://programm.top/c-sharp/algorithm/levenshtein-distance (date of access: 30.10.2020).

6. **Testing** services and web applications ASP.NET Core, available at: https://docs.microsoft.com/ru-ru/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps (date of access: 30.10.2020).

7. **Federal** information address system, available at: https://fias.nalog.ru/Updates (date of access: 30.10.2020).

8. **3-Tier** Architecture: A Complete Overview, available at: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview (date of access: 30.10.2020).

9. **Micro** ORM vs ORM, available at: https://www.yaplex.com/blog/micro-orm-vs-orm (date of access: 30.10.2020).

10. **Python vs. C#:** Comparison of the Programming Languages, available at: https://www.netguru.com/blog/python-vs-c-sharp (date of the application: 30.10.2020).

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**
E-mail: it@novtex.ru

Технический редактор *Е. В. Конова*.
Корректор *М. Ю. Безменова*.
Сдано в набор 07.03.2021. Подписано в печать 26.04.2021. Формат 60×88 1/8. Бумага офсетная.
Усл. печ. л. 8,86. Заказ IT521. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

Рисунки к статье С. М. Салибекяна

«ОБЪЕКТНО-АТРИБУТНЫЙ ПОДХОД ДЛЯ СЕМАНТИЧЕСКОГО АНАЛИЗА ЕСТЕСТВЕННОГО ЯЗЫКА»

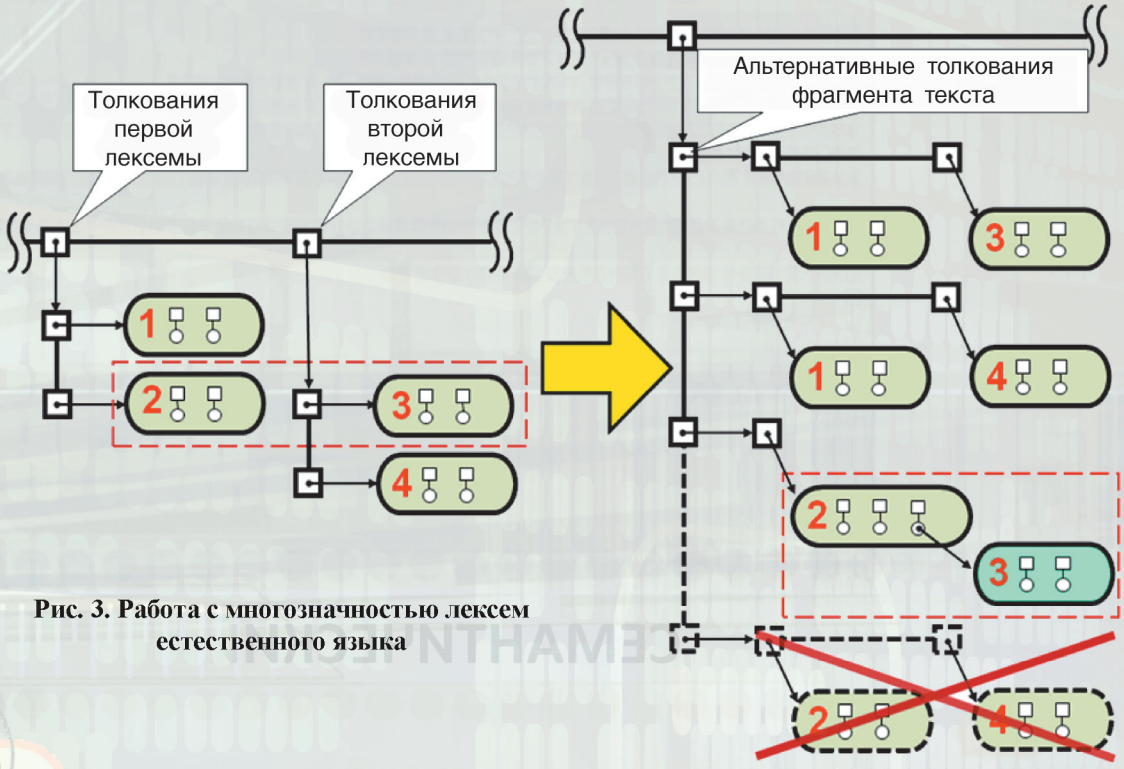


Рис. 3. Работа с многозначностью лексем естественного языка

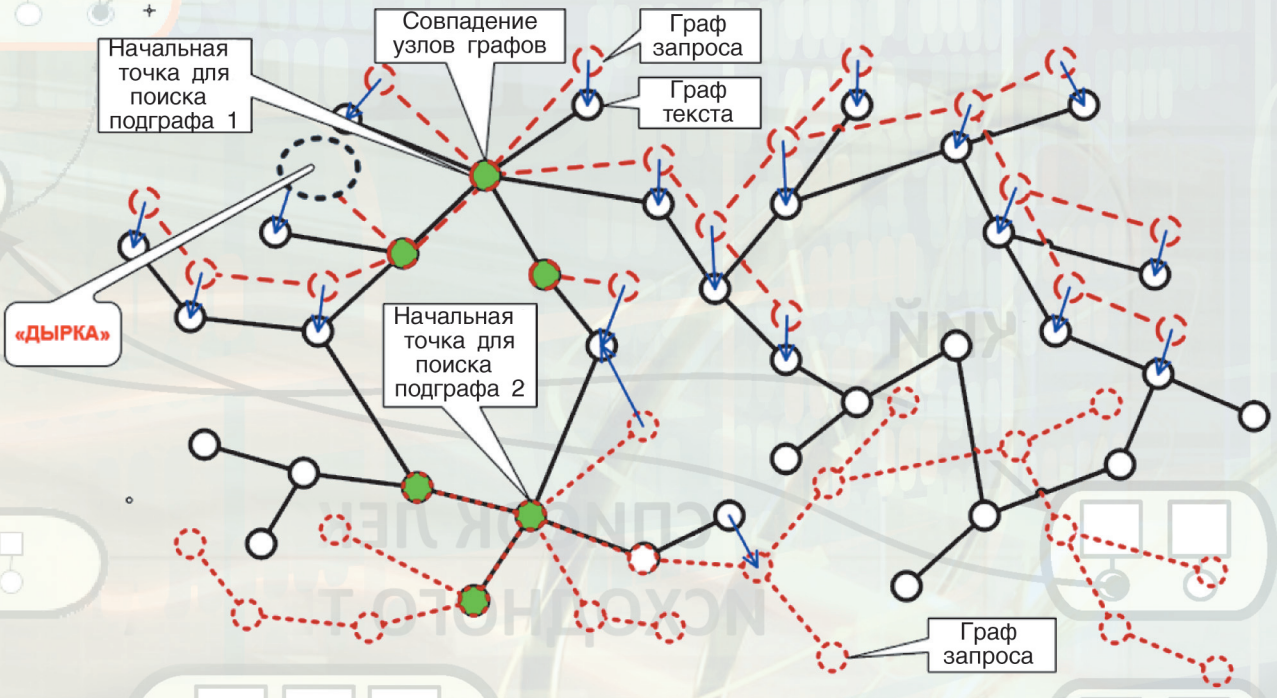


Рис. 4. Поиск подграфа в семантическом ОА-графе текста

Издательство «НОВЫЕ ТЕХНОЛОГИИ»

выпускает *научно-технические журналы*

Ежемесячный теоретический
и прикладной научно-технический журнал

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В журнале освещаются современное состояние, тенденции и перспективы развития основных направлений в области разработки, производства и применения информационных технологий.

Подписной индекс по Объединенному каталогу
«Пресса России» – 72656

Подписной индекс по Объединенному каталогу
«Пресса России» – 79963

Научно-практический
и учебно-методический журнал

БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

В журнале освещаются достижения и перспективы в области исследований, обеспечения и совершенствования защиты человека от всех видов опасностей производственной и природной среды, их контроля, мониторинга, предотвращения, ликвидации последствий аварий и катастроф, образования в сфере безопасности жизнедеятельности.

Междисциплинарный
теоретический и прикладной
научно-технический журнал

НАНО- и МИКРОСИСТЕМНАЯ ТЕХНИКА

В журнале освещаются современное состояние, тенденции и перспективы развития нано- и микросистемной техники, рассматриваются вопрос разработки и внедрения нанои микросистем в различные области науки, технологии и производства.

Подписной индекс по Объединенному каталогу
«Пресса России» – 79493

Подписной индекс по Объединенному каталогу
«Пресса России» – 79492

Ежемесячный теоретический
и прикладной
научно-технический журнал

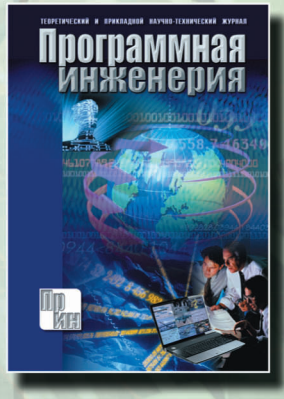
МЕХАТРОНИКА, АВТОМАТИЗАЦИЯ, УПРАВЛЕНИЕ

В журнале освещаются достижения в области мехатроники, интегрирующей механику, электронику, автоматику и информатику в целях совершенствования технологий производства и создания техники новых поколений. Рассматриваются актуальные проблемы теории и практики автоматического и автоматизированного управления техническими объектами и технологическими процессами в промышленности, энергетике и на транспорте.

Теоретический
и прикладной
научно-технический журнал

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

В журнале освещаются состояние и тенденции развития основных направлений индустрии программного обеспечения, связанных с проектированием, конструированием, архитектурой, обеспечением качества и сопровождением жизненного цикла программного обеспечения, а также рассматриваются достижения в области создания и эксплуатации прикладных программно-информационных систем во всех областях человеческой деятельности.



Подписной индекс по Объединенному каталогу
«Пресса России» – 22765

Адрес редакции журналов для авторов и подписчиков:
107076, Москва, Стромынский пер., 4. Издательство "НОВЫЕ ТЕХНОЛОГИИ".
Тел.: (499) 269-55-10, 269-53-97. Факс: (499) 269-55-10. E-mail: antonov@novtex.ru

ISSN 1684-6400. Информационные технологии. 2021. Том 27. № 5. 225–280. Индекс: 72656 (Объединенный каталог «Пресса России»)