УДК 519.712.3 **DOI** 10.17150/2500-2759.2020.30(4).575-585

ТЕОРЕТИКО-МНОЖЕСТВЕННЫЙ АНАЛИЗ ОРГАНИЗАЦИИ ДАННЫХ УЧЕБНОГО ПРОЦЕССА И АЛГОРИТМЫ ПРОЕКТИРОВАНИЯ РАСПИСАНИЯ С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА*

В.И. Мартьянов[1, 2]
[1] Иркутский национальный исследовательский технический университет, г. Иркутск, Российская Федерация
[2] Байкальский государственный университет, г. Иркутск, Российская Федерация

Информация о статье

Дата поступления
28 октября 2020 г.

Дата принятия к печати
18 декабря 2020 г.

Дата онлайн-размещения
30 декабря 2020 г.

Ключевые слова

NP-трудные задачи; системы искусственного интеллекта; логико-эвристические методы; реляционные базы данных; интеллектуальный возврат

Аннотация

Рассмотрены вопросы создания информационного и программного обеспечения организации учебного процесса, включая автоматизированный расчет качественных расписаний. Предложены две новые стратегии решения ***NP***-трудных задач, которые могут быть рекомендованы для систем искусственного интеллекта.

SET-THEORY ANALYSIS OF THE DATA ORGANIZATION OF THE EDUCATIONAL PROCESS AND ALGORITHMS FOR DESIGNING A SCHEDULE WITH ELEMENTS ARTIFICIAL INTELLIGENCE**

Vladimir I. Martyanov[1, 2]
[1] Irkutsk National Research Technical University, Irkutsk, the Russian Federation
[2] Baikal State University, Irkutsk, the Russian Federation

Article info

Received
October 28, 2020

Accepted
December 18, 2020

Available online
December 30, 2020

Keywords

NP-hard problems; artificial intelligence systems; logical-heuristic methods; relational databases; smart return

Abstract

The article examines the issues of creating information and software for organizing the educational process, including automated calculation of high-quality timetables. Two new strategies for solving ***NP***-hard problems are proposed, which can be recommended for systemic artificial intelligence

* Автор благодарен Д.В. Пахомову за подготовку данных для расчета расписания по стандартам, указанным автором.
** The author is grateful to D.V. Pakhomov for preparing data for calculating the schedule according to the standards specified by the author.

Введение

В статье рассматриваются вопросы разработки информационного и программного обеспечения организации учебного процесса, которые опробованы на опыте внедрения таких программ в двух крупных университетах Иркутска — ИрГУПС и ИРНИТУ. Этот программный комплекс для удобства будем называть сокращенно ***МаПа***.

Первым этапом организации учебного процесса является формирование учебных планов (УП) специальностей на основе федеральных государственных образовательных стандартов. В настоящее время необходимость использования своего программного обеспечения для этого этапа закрывается так называемой Шахтинской программой, которая заодно формирует учебные планы в формате, требуемой для отчетности перед Москвой, что и использует большая часть вузов (и вообще учебных заведений) России, включая ИрГУПС и ИРНИТУ, но ряд крупных университетов не использует, в том числе иркутский БГУ.

Второй этап организации учебного процесса — формирование рабочих учебных планов (РУП) специальностей на основе УП и контингента учащихся. Шахтинская программа закрывает и этот этап, но, например, ИРНИТУ данной возможностью Шахтинской программы не пользуется. Отметим, что программный комплекс ***МаПа*** имеет интерфейс перевода РУП Шахтинской программы в свою базу данных (БД).

Третьим этапом организации учебного процесса выступает расчет нагрузки на основе РУП и нормативов часов (для консультаций, зачетов, экзаменов и др.).

Четвертый и пятый этапы организации учебного процесса связаны с формированием учебных поручений кафедрам и преподавателям.

Шестой и седьмой этапы организации учебного процесса связаны с формированием типового и рабочего графиков учебного процесса.

Восьмым этапом является расчет расписания занятий, который, конечно, требует наличия БД по аудиторному фонду, включая привязку аудиторий к занятиям, которые в них могут проводиться, и многого другого (например, разбиение пар по сменам, определение времени освобождения от занятий для преподавателей, учащихся и аудиторий и пр.).

Более конкретно, в первой части статьи будет рассмотрен проект БД комплекса ***МаПа*** (без громоздких технических деталей) с позиции теоретико-множественного анализа, что естественно для реляционной БД и файловой системы решателя, основанной на представлении данных конечными деревьями [1].

Для современных информационных технологий реляционные базы данных обеспечивают вычислимость запросов определенных типов со скоростью, не зависящей от объема данных (более подробно ниже).

Во второй части статьи будут рассмотрены математические основы высокоэффективных алгоритмов автоматического проектирования расписания, которые обеспечивают его высокое качество (достаточно сказать, что для ИРНИТУ у бакалавров автоматически проектируется более 95 % занятий при условии наличия у преподавателя не более одного дня с единственным занятием в двухнедельном цикле, что совершенно недостижимо при ручном проектировании). Кроме того, несмотря на большой объем данных (около 10 тыс. занятий, более 1 тыс. преподавателей, более 1 тыс. подгрупп, групп, потоков студентов и весьма ограниченный аудиторный фонд), автоматический расчет основы расписания производится на офисном компьютере за неделю, хотя задачи сетевого планирования (включая расчет расписания) являются ***NP***-трудными (полными) [2], т.е. комбинаторными задачами высокой сложности.

Общую схему решения комбинаторных задач высокой сложности логико-эвристическими методами [3] можно трактовать как преобразование начальной (инициальной) многоосновной модели [4; 5]:

$$M_{ini} = <A_1, \ldots, A_s; p_1, \ldots, p_k>,$$

где A_i — основные множества; p_i — предикаты (отношения) на основных множествах, в конечную (финальную) M_{fin}, удовлетворяющую ограничениям $R_1, R_2, \ldots, R_m$.

Если искать аналоги, то последовательность таких преобразований можно считать допустимым (без оптимизации значений целевого функционала) управлением для задачи динамического программирования [6], где $R_1, R_2, \ldots, R_m$ — фазовые ограничения.

1. Теоретико-множественный анализ организации проекта БД комплекса *МаПа*

Как уже отмечалось во введении, БД комплекса ***МаПа*** является реляционной [7], что соответствует современным тенденциям и позволяет использовать технологии ***BigTable***[1], обеспечивающие эффективность выполнимости запросов на языке ***SQL*** [7] вне зависимости от объема данных реляционной БД.

Практическим подтверждением этого для любого человека из цивилизованного мира, имеющего банковскую электронную кар-

ISSN 2500-2759

точку, служит скорость работы банковских систем, использующих сетевые реляционные базы данных, с их мировыми сетями терминалов и банкоматов, где можно проводить операции с вкладами и денежными средствами в любой точке мира за считаные секунды. Практически эту же технологию используют полнотекстовые поисковые системы в интернет-пространстве («Гугл», «Яндекс» и др.), а также системы проверки плагиата.

В основе этого лежит следующий довольно просто доказываемый факт: сложность проверки принадлежности кортежа $(a_1, a_2, \ldots, a_n)$, где $a_1 \in A_1$, $a_2 \in A_2$, …, $a_n \in A_n$ отношению H, определенному на конечных множествах $A_1, A_2, \ldots, A_n$, линейна от n (т.е. не зависит от числа кортежей, составляющих отношение H).

При компьютерной реализации конечные множества $A_1, A_2, \ldots, A_n$ являются доменами, а отношение H — таблицей реляционной БД.

Соответственно, вопрос о принадлежности кортежа $(a_1, a_2, \ldots, a_n)$ отношению H сводится к проверке не пустоты курсора, полученного после выполнения ***SQL***-запроса: *SELECT * FROM* H *WHERE* $A_1 = a_1$ AND $A_2 = a_2$ AND … AND $A_n = a_n$, что выполняется со скоростью, линейно зависящей от n. Если искать истоки данных результатов, то следует обратиться к трудам Д. Кнута, где представлен древовидный способ организации данных [8].

1.1. Проект БД комплекса *МаПа*

Таблица Учебный план *Uchplan*

0. id_10 (счетчик)
1. Шифр учебного плана shifr (ссылка на таблицу шифров)
2. Цикл стандарта ciklst (char 20)
3. Название предмета pred (ссылка на словарь предметов vacpred)
4. Кафедра kaf (ссылка на словарь кафедр vackaf)
5. Кол-во часов по стандарту stvsegoh (int 4) (для заочной — число часов по УП)
6. Аудиторных часов по стандарту staudth (int 4)
7. Общее кол-во часов vsegoh (int 4)
8. Аудиторные часы audth (int 4) (для заочной — число очных часов)
9. Часы самостоятельной р-ты ст-та srsh (int 4)
10. Ссылка на предка parent (int 4)
11. Глубина дерева depth (int 2)
12. Структурный индекс structidx (char20)
13. Включение в РУП vklrup (binary) (1 — включен в РУП, 0 — не включен в РУП)
14. Факультатив facult (1 — факультатив, 0 — не факультатив)

Таблица Распределение часов по семестрам *Up_sem*

0. id_11 (счетчик)
1. Предмет up (ссылка на uchplan.id_1)
2. Семестр sem (ссылка на словарь семестров vacsem)
3. Кол-во часов в неделю hy (int 2)
4. Кол-во часов h (int 4)

Таблица Распределение контроля по семестрам *Up_k_sem*

0. id_12 (счетчик)
1. Предмет up (ссылка на uchplan.id_1)
2. Семестр sem (ссылка на словарь семестров vacsem)
3. Вид контроля vk (ссылка на словарь видов контроля vacvk)
4. Дополнительная информация (для курсовой работы — количество курсовых в семестре, для остальных 128 — па) dopinfo (int 4)

Таблица Распределение часов по видам работ и по семестрам *Up_vr_sem*

0. id_13 (счетчик)
1. Предмет up (ссылка на uchplan.id_1)
2. Вид работы vr (ссылка на словарь видов работ vacvr)
3. Семестр sem (ссылка на словарь семестров vacsem)
4. Кол-во часов h (int 4)
5. Кол-во часов в неделю hy (int 2)

Таблица шифров учебных планов *shup*

0. id_14
1. shifr (char 10)
2. Комитет komitet (char 100) * для отчета
3. Министерство minvo (char 100) * для отчета
4. Учебное заведение uchzav (char 100) * для отчета
5. Кем утвержден mutv (char 100) * для отчета
6. Должность утверждателя mdutv (char 50) * для отчета
7. Квалификация kval (char 100) * для отчета
8. Дата утверждения dutv (date)
9. Специальность spec (ссылка на словарь специальностей vacspclnt)
10. Специализация spclntion (char 100)
11. Цифровое обозначение специализации nspclntion (char 4)
12. Идентификатор учебного плана priznak (char 2)
13. Srobuch (int 2)
14. Форма обучения fobuch (int 4) (1 — дневная, 2 — заочная)
15. Адм. назв-е ф. обуч-я afobuch (int 4)
16. Srobuchstr (char 20) — Срок обучения строка (для отчета)

Таблица Словарь предметов *Vacpred*

0. id_15

ISSN 2500-2759

1. pred (char 100)

Таблица План практик *Planprakt*

0. id_16

1. Название практики prakt (ссылка на словарь практик)

2. Кафедра kaf (ссылка на словарь кафедр)

3. Шифр УП shifr (ссылка на таблицу шифров УП)

4. Семестр sem (ссылка на словарь семестров)

5. Кол-во недель week (int 4)

6 Ссылка на предка parent (int 4)

7. Глубина дерева depth (int 4)

Таблица Словарь кафедр *Vackaf*

0. id_17

1. kaf (char 100)

2. sokr (char 10)

Таблица Словарь семестров *Vacsem*

0. id_18

1. sem (char 20)

Таблица Словарь видов контроля *Vacvk*

0. id_19

1. vk (char 30)

Таблица Словарь видов работ *Vacvr*

0. id_20

1. vr (char 30)

Таблица Распределение часов по видам работ *Up_vr*

0. id_21

1. Ссылка на УП up (int 4)

2. Ссылка на вид работы vr (int 4)

3. Кол-во часов h (int 4)

Таблица распределение контроля по семестрам *G_up_vk_sem*

0. id_22

1. Ссылка на УП up (int 4)

2. Вид контроля 1 из словаря (char 30) (fk1)

…

N. Вид контроля N из словаря (char 30) (fkN)

Fk1 — зачеты, fk2 — экзамены, fk3 — КР, fk4 — КП

Таблица Распределение количества недель по семестрам *Up_w_sem*

0. id_23

1. Ссылка на шифр УП shup (int 4)

2. Ссылка на семестр sem (int 4)

3. Кол-во недель w (int 4)

4. Включение в РУП vklrup (int 4)

Таблица Словарь названий практик *Vacprakt*

0. id_24

1. prakt (char 100)

Таблица Типовой график *Typgraph*

0. id_30 (счетчик)

1. Шифр учебного плана shup (ссылка на shup.id_14)

2. Курс kurs (int 2)

3. Вид работы (цикл) vr (ссылка на словарь видов работ графика vacgrvr.id_gv)

4. Номер по порядку следования цикла npp (int 4)

5. Кол-во дней на цикл days (int 4)

Таблица Настройки для типового графика (ДФО) *Settgrd*

0. id_31

1. Шифр учебного плана shup (ссылка на shup.id_14)

2. Курс kurs (int 2)

3. Дней на подготовку к экзамену осень dfpo (int 2)

4. Дней на подготовку к экзамену весна dfpv (int 2)

5. Кол-во недель на зимние каникулы zkan (int 2) (по умолчанию 2)

6. Единица планирования dkont (int 2) (3 — поток, 2 — группа, 1 — подгруппа)

Таблица Настройки для типового графика (ЗФО) *Settgrz*

0. id_32

1. Шифр учебного плана shup (ссылка на shup.id_14)

2. Курс kurs (int 2)

3. Дней на подготовку к экзамену во 2 сессии dfpe2 (int 2)

4. Дней на подготовку к экзамену в 3 сессии dfpe3 (int 2)

5. Дней на сдачу зачета во 2 сессии dfpz2 (int 2)

6. Дней на сдачу зачета в 3 сессии dfpz3 (int 2)

7. Недель 1-й пустоты empty1 (int 2)

8. Недель 2-й пустоты empty2 (int 2)

9. Недель 3-й пустоты empty3 (int 2)

10. Часов в день на студента в 1-й сессии hdstud1 (int 2)

11. Часов в день на студента во 2-й сессии hdstud2 (int 2)

12. Недель каникул wkan (int 2)

13. Единица планирования dkont (int 2) (3 — поток, 2 — группа, 1 — подгруппа)

Таблица Словарь циклов для графика (ДФО) *Vacgrvrd*

0. id_grvr (формируется так: 1 — Теоретическое обучение (осень), 2 — Зимняя сессия, 3 — Зимние каникулы, 4 — Теоретическое обучение (весна), 5 — Летняя сессия, 6 — Летние каникулы, 128 + planprakt. id_16 — Практика) (int 4)

1. Название цикла cikl (char 100)

2. Сокращение для цикла (полная неделя) obozn (char 1)

3. Сокращение для цикла (половина недели) pobozn (char 1)

4. В какую колонку суммировать sumcol (ссылка на ltoggraph.id_33)

ISSN 2500-2759

Таблица Словарь циклов для графика (ЗФО) *Vacgrvrd*

0. id_grvr (формируется так: 1 — Пустота, 2 — Теоретическое обучение (1-я сессия), 3 — Теоретическое обучение (2-я сессия), 4 — Теоретическое обучение (3-я сессия), 5 — Каникулы, 128 + planprakt.id_16 — Практика) (int 4)

1. Название цикла cikl (char 100)

2. Сокращение для цикла (полная неделя) obozn (char 1)

3. Сокращение для цикла (половина недели) pobozn (char 1)

4. В какую колонку суммировать sumcol (ссылка на ltoggraph.id_33)

Таблица Рабочий график *Rabgraph*

1. Контингент (идентификатор потока) kont (int 4)

2. Идентификатор контингента (для потока — 0) kontid (int 4)

3. Вид работы (цикл) vr (ссылка на словарь видов работ графика vacgrvrd.id_gv или vacgrvrz.id_gv)

4. Дата начала проведения dbeg (int 2) (измеряется в количестве дней от начала учебного года)

5. Кол-во дней days (int 2)

6. Форма обучения (1 — дневная, 2 — заочная)

Таблица для подбивания итогов в графике *ltoggraph*

0. id_33

1. Название колонки colname (1 — Учебная практика; 2 — По профилю специальности; 3 — Преддипломная практика; 4 — Государственная аттестация; 5 — ‘ ‘)

Таблица Контингент учащихся по курсам *Kontkurs*

0. **id_1** (счетчик)

1. Шифр учебного плана **shup** (ссылка на таблицу шифров shup.id_14)

2. Филиал **fil** (ссылка на словарь филиалов vacfil)

3. Факультет **fac** (ссылка на словарь факультетов vacfac)

4. Контингент **aobozn** (ссылка на словарь обозначений vacaobozn)

5. Курс **kurs** (int 2)

6. Кол-во учащихся **stud** (int 2)

7. Кол-во групп **groups** (int 2)

8. Кол-во подгрупп **pgroups** (int 2)

9. Смена **smenao** (ссылка на словарь смен vacsmena) Осень

10. Смена **smenav** (ссылка на словарь смен vacsmena) Весна

11. Текстовый идентификатор **obozn** (char 20)

12. Поле для группирования в отчете **groupkey** (ссылка на словарь обозначений vacaobozn)

13. Строка со списком **undoworksps** (char 250)

14. Строка со списком **anothernumstudsps** (char 250)

15. Новое кол-во студентов **newnumstud** (int 2)

16. Необходимость переформирования графика **ntcgraph** (int 2)

Таблица Словарь смен *Vacsmena*

0. **id_2** (счетчик)

Смена s**mena** (char 50)

Таблица Словарь форм обучения *Vacfobuch*

0. **id_3** (счетчик)

Форма обучения **fobuch** (char 50)

Таблица Словарь филиалов *Vacfil*

0. **id_4** (счетчик)

Филиал **fil** (char 50)

Таблица Словарь факультетов *Vacfac*

0. **id_5** (счетчик)

Факультет **fac** (char 255)

Сокращение **sokr** (char 10)

Словарь контингентов *Vacaobozn*

0. **id_6** (счетчик)

Контингент **aobozn** (char 50)

Таблица Контингент по группам *Kontgrp*

0. **id_7** (счетчик)

1. Текстовый идентификатор **obozn** (char 20)

2. Номер группы (подгруппы) **ngroup** (int 2)

3. Кол-во учащихся **students** (int 2)

4. Ссылка на поток **kont** (kontkurs.id_1)

5. Глубина дерева **depth** (int 2) (или признак для определения контингента, например: 1 — группа, 2 — подгруппа)

6. Ссылка на родителя **parent** (int 2)

Таблица Данные для объединения и переформирования контингента *Potoklist*

0. Внутренний номер объединенного потока **op** (int 4)

1. Кол-во групп в потоке **groups** (int 2)

2. Кол-во подгрупп в потоке **pgroups** (int 2)

3. Кол-во студентов в потоке **stud** (int 4)

4. Будет ли объединение по лекциям **oblek** (int 2) (1 — будет, 0 — нет)

5. Будет ли объединение по семинарам **obsem** (int 2) (1 — будет, 0 — нет)

6. Будет ли объединение по лекциям **obsem** (int 2) (1 — будет, 0 — нет)

7. Список потоков **konts** (char 200)

8. Лекций sem (int 4)

9. Семинаров (int 4)

10. Лабораторных (int 4)

Таблица Список объединяемых потоков *Kontlist*

0. Номер объединенного потока **op** (int 4) (ссылка на номер потока Potoklist.op)

ISSN 2500-2759

1. Ссылка на административный поток **kont** (int 4) (kontkurs)

Таблица Список невыполненных работ *Undowork*

0. Ссылка на поток (kontkurs.id_1) **kont**

1. Ссылка на нормы времени (normtime. id_40) **vr** (int 4)

Таблица Список работ с другой численностью студентов *Anothernumstud*

0. Ссылка на поток (kontkurs.id_1) **kont**

1. Ссылка на нормы времени (normtime. id_40) **vr** (int 4)

1.2. Организация файловой системы решателя

Как уже отмечалось выше, проектирование расписания занятий является комбинаторной проблемой большой сложности, поэтому таблицы реляционной БД должны быть переведены в древовидные структуры специальной формы, которые будут рассмотрены в этом разделе. Конечно, может возникнуть вопрос, почему формат реляционных таблиц, столь эффективный для решения многих задач, недостаточен для решения комбинаторных проблем большой сложности. Действительно, это так и по теоретическим результатам, поскольку технология ***BigTable*** создана для решения задач полиноминальной сложности, а задачи проектирования расписания занятий таковыми не являются [9], хотя это пока не доказано и объявлено одной из математических проблем третьего тысячелетия (с миллионными долларовыми премиями от фондов, обществ и институтов).

Как уже отмечалось во введении, математическая формализация строится в рамках многоосновных моделей [4; 5]

$$M = <A_1, \ldots, A_s; p_1, \ldots, p_k>, \qquad (1)$$

где A_i — основные множества; p_i — предикаты (отношения) на основных множествах. При компьютерной реализации основные множества $A_1, A_2, \ldots, A_n$ становятся доменами, а отношения $p_1, p_2, \ldots, p_k$ — таблицами реляционной БД.

Для обеспечения эффективной реализации работы с данными, конечно, приходится использовать и стандартные средства, такие как сортировка, двоичный поиск, хеширование и др. Применение этих средств соответствует описаниям, приведенным в классических книгах Д. Кнута [8].

Структура, задающая вершину дерева (по терминологии Д. Кнута) — ***ссылками «левый сын — правый брат»***

```
struct Tree {
    int value;      //значение, приписанное вершине (операция, число, имя и др.)
    int down;       //ссылка на первого потомка (–1: отсутствие ссылки);
    int right;      //ссылка на следующего брата (–1: отсутствие ссылки);
};
```

Отсутствие ссылки на следующую вершину для компонент ***down*** и ***right*** дерева определяется значением **–1**.

Выделение ***k*** элементов структуры ***Tree*** по указателю ***tree*** задается оператором

```
tree = (struct Tree  *) calloc (k, sizeof (struct Tree)).   (2)
```

Функция ***sizeof*** задает объем памяти, необходимый под размещение одного экземпляра данных типа **struct *Tree*** в байтах. Таким образом, оператор ***(2)*** задает по указателю ***tree*** область в ***k * sizeof(*****struct *Tree)*** байт.

Доступ к ***i***-му экземпляру структуры ***Tree*** осуществляется командой ****(tree + i)***. Например, если необходимо переменной ***val*** (типа ***int***) присвоить значение элемента ***value i***-го экземпляра структуры ***Tree***, определенной оператором ***(2)***, то это задается оператором

```
val = ( tree + i)–> value.   (3)
```

Эквивалентные формы записи оператора ***(3)*** следующие:

```
val = * ( tree + i). value;
val = tree[ i]. value.   (4)
```

*Оператор **(4)** наиболее наглядно показывает эквивалентность понятий **указателей и массивов.***

Задание основных множеств $A_1, \ldots, A_s$ многоосновной модели ***M (1)*** (доменов реляционной БД) в оперативной памяти осуществляется выделением, соответственно, областей (**куч**)

```
Set1=(structSet1 *)calloc(k1,sizeof(structSet1));
Set2=(structSet2 *)calloc(k2,sizeof(structSet2));
.....
Sets=(structSets *)calloc(ks,sizeof(structSets)),
```

где, соответственно, структуры $Set_1, Set_2, \ldots, Set_s$ задают необходимые свойства элементов основных множеств $A_1, A_2, \ldots, A_s$, соответственно, $k_1, k_2, \ldots, k_s$ — размерности основных множеств $A_1, A_2, \ldots, A_s$.

Доступ к ***i***-му элементу основных множеств $A_1, A_2, \ldots, A_s$, соответственно, осу-

ISSN 2500-2759

ществляется операторами ***(Set $_1$ + i), (Set $_2$ + + i), …, (Set $_s$ + i).*** Доступ к компоненте ***value*** *i*-го элемента основных множеств A_1, A_2, …, A_s, соответственно, осуществляется операторами ***(Set $_1$ + i) –> value, (Set $_2$ + i) –> value, …, (Set $_s$ + i) –> value.***

Задание отношений p_1, …, p_k многоосновных моделей (таблиц реляционной БД) в оперативной памяти осуществляется выделением, соответственно, областей (**куч**)

Relat $_1$ = (struct Relat $_1$ *) calloc (m_1, sizeof(struct Relat $_1$));

Relat $_2$ = (struct Relat $_2$ *) calloc (m_2, sizeof(struct Relat $_2$));

……

Relat $_k$ = (struct Relat $_k$ *) calloc (m_k, sizeof(struct Relat $_k$)),

где, соответственно, структуры ***Relat $_1$, Relat $_2$, …, Relat $_k$*** служат для задания кортежей элементов основных множеств A_1, A_2, …, A_s, составляющих отношения p_1, …, p_k, а числа m_1, m_2, …, m_k определяют количество кортежей отношений p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, т.е. предполагаем, что отношения $p_1, \ldots, p_k \subset A_1 \times A_2 \times \ldots \times A_s$.

Доступ к *i*-му кортежу отношений p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, осуществляется, соответственно, операторами ***(Relat $_1$ + i), (Relat $_2$ + i), …, (Relat $_k$ + i).*** Доступ к *j*-му элементу *i*-го кортежа отношений p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, соответственно, осуществляется операторами ***(Relat $_1$ + i) –> value$_f$ (Relat $_2$ + i) –> value$_f$ …, (Relat $_k$ + i) –> value$_j$.***

Для обеспечения быстрого доступа к кортежам отношений p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, используется **древовидная** организация данных, формируемая по принципу расположения *j*-х элементов кортежей на *j*-м этаже древовидной структуры.

Пусть явное определение множества кортежей отношений p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, имеет вид

$$p_1 = \{(a_{11}, a_{12}, \ldots, a_{1n}) \mid a_{11} \in A_1, a_{12} \in A_2, \ldots, a_{1n} \in A_n\},$$

$$p_2 = \{(a_{21}, a_{22}, \ldots, a_{2n}) \mid a_{21} \in A_1, a_{22} \in A_2, \ldots, a_{2n} \in A_n\},$$

……

$$p_k = \{(a_{k1}, a_{k2}, \ldots, a_{kn}) \mid a_{k1} \in A_1, a_{k2} \in A_2, \ldots, a_{kn} \in A_n\}.$$

Задание в оперативной памяти древовидных структур Tre_1, Tre_2, …, Tre_k, задающих отношения p_1, …, p_k, определенных на основных множествах A_1, A_2, …, A_s, осуществляется выделением, соответственно, областей (**куч**)

Tr $_1$ = (struct Tree *) calloc (n * m_1, sizeof(struct Tree));

Tr $_2$ = (struct Tree *) calloc (n * m_2, sizeof(struct Tree));

……

Tr $_k$ = (struct Tree *) calloc (n * m_k, sizeof(struct Tree)),

где структура **struct *Tree*** определена выше для примера ***(2)***

Рассмотрим для произвольного *i*, где ***1 <= i <= k,*** построение по индукции от числа кортежей m_i в отношении p_i древовидной структуры Tre_i, задающей отношение p_i, определенное на основных множествах A_1, A_2, …, A_s.

Основание индукции. Пусть первый кортеж отношения p_i имеет вид ***(a_1, a_2, …, a_s),*** где $a_1 \in A_1$, $a_2 \in A_2$, …, $a_s \in A_s$ и элементы a_1, a_2, …, a_s имеют индексы, соответственно, t_1, t_2, …, t_s в представлении ***(Set $_1$ + t $_1$), (Set $_2$ + t $_2$), …, (Set $_s$ + t $_s$).*** Тогда начальные присвоения для элементов кучи по адресу (указателю) ***Tr $_i$*** будут:

– формирование 0-го этажа (корень дерева):

(Tr $_i$ + 0)–>value = –1; (Tr $_i$ + 0)–>down = = 1; (Tr $_i$ + 0)–>right = –1;

– формирование 1-го этажа:

(Tr $_i$ + 1)–>value = t_1; (Tr $_i$ + 1)–>down = = 2; (Tr $_i$ + 1)–>right = –1;

– формирование 2-го этажа:

(Tr $_i$ + 2)–>value = t_2; (Tr $_i$ + 2)–>down = = 3; (Tr $_i$ + 2)–>right = –1;

……

(Tr $_i$ + s) –>value = t_s; (Tr $_i$ + s) –>down = = –1; (Tr $_i$ + s) –>right = –1,

причем 1-й свободный элемент ***Fr1_Tr $_i$*** кучи по адресу (указателю) ***Tr $_i$*** будет ***s + 1.***

Индукционный шаг. Пусть очередной кортеж отношения p_i, включаемый в древовидную структуру, имеет вид ***(b_1, b_2, …, b_s),*** где $b_1 \in A_1$, $b_2 \in A_2$, …, $b_s \in A_s$ и элементы b_1, b_2, …, b_s имеют индексы, соответственно, u_1, u_2, …, u_s в представлении ***(Set $_1$ + u $_1$), (Set $_2$ + u $_2$), …, (Set $_s$ + u $_s$).***

Предположим далее, первые *l* (*l* < *s* обязательно!) элементов b_1, b_2, …, b_l совпадают с некоторой цепочкой w_1, w_2, …, w_l, идущей

ISSN 2500-2759

от начальной (нулевой) вершины древовидной структуры. Более точно это означает следующее:

– вершина w_1 — непосредственный потомок начальной (нулевой) вершины;

– вершина w_{o+1} — непосредственный потомок вершины w_o;

– $(Tr_i + w_1) \rightarrow value = u_1$; $(Tr_i + w_2) \rightarrow value = u_2$; ... $(Tr_i + w_l) \rightarrow value = u_l$;

– если ww самый правый непосредственный потомок вершины w_l, то

$(Tr_i + ww) \rightarrow right = Fr1\_Tr_i$;

$(Tr_i + Fr1\_Tr_i) \rightarrow right = -1$; $(Tr_i + Fr1\_Tr_i) \rightarrow value = u_{l+1}$;

.....

$(Tr_i + Fr1\_Tr_i + s - l) \rightarrow right = -1$; $(Tr_i + + Fr1\_Tr_i + s - l) \rightarrow value = u_s$,

причем 1-й свободный элемент $Fr1\_Tr_i$ кучи по адресу (указателю) Tr_i будет обновлен присвоением $Fr1\_Tr_i = Fr1\_Tr_i + s - l$.

Таким образом, полностью рассмотрена схема перевода таблиц в древовидные структуры, размещенные в оперативной памяти. А файловая система решателя состоит в основном из файлов, соответствующих древовидным структурам.

2. Математические основы алгоритмов автоматического проектирования расписания

2.1. Организация данных и структур управления проектированием

Одним из основных свойств алгоритмов проектирования расписания является одинаковый способ организации данных и структур управления, что позволяет легко реализовывать один из важнейших принципов систем искусственного интеллекта — ***принцип обратной связи*** (ранее считался свойством кибернетических систем). То есть данные формируют структуры управления пошагового решения, а на некоторых шагах часть структур управления переходит в данные и т.д. в итерационном режиме с возвратом на некоторые точки пошагового решения для выбора других вариантов. Отметим, что оптимизация переборов — одна из важнейших задач систем с элементами искусственного интеллекта.

Основными объектами задачи планирования расписания (по дневной форме обучения) являются множества:

– преподавателей $P = \{p_1, p_2, \ldots, p_{kp}\}$;

– контингентов учащихся $C = \{c_1, c_2, \ldots, c_{kc}\}$;

– предметов $D = \{d_1, d_2, \ldots, d_{kd}\}$;

– аудиторий $A = \{a_1, a_2, \ldots, a_{ka}\}$;

– пар (двухнедельный цикл) $M = \{m_1, m_2, \ldots, m_{km}\}$.

Совокупность планируемых занятий $Z \subseteq P \times C \times D$. Расписание занятий Sh — инъективное отображение $Z \rightarrow Res$, где $Res = A \times M$ — ресурс. В дальнейшем под объектом будем понимать либо преподавателя, либо контингент.

Расписание должно удовлетворять следующим (основным) ограничениям:

1. Занятия объектов не должны пересекаться по времени (для контингентов занятия по выбору проводятся одновременно).

2. В занятиях контингентов и преподавателей не должно быть окон.

3. Единственные в день занятия у контингентов и преподавателей должны отсутствовать.

4. Преподаватели и контингенты не могут иметь больше максимума занятий в день (для преподавателей эта константа может устанавливаться индивидуально) и больше максимума лекций в день (а также занятий других специализаций, если это необходимо).

5. Следует учитывать типизацию аудиторного фонда и предметов.

6. Необходимо учитывать временны́е интервалы для перемещения между корпусами преподавателей и контингентов.

Ограничение 1 учитывается за счет заполнения карточек недельной нагрузки преподавателей и контингентов. Ограничение 5 учитывается за счет сортировки аудиторного фонда и последующего «урезания» ресурса занятия. Ограничения 2, 4 поддерживаются демонами, которые проверяют их выполнимость при каждой попытке проектирования занятия (выделения ресурса). Ограничения 3, 6 (и некоторые другие) проверяются на этапе глобального проектирования (второй этап).

Задание ограничений $R_1, R_2, \ldots, R_m$, которым должны удовлетворять многоосновные модели, и методы проверки выполнимости этих ограничений имеют отличия программных решений для таблиц, что рассмотрены выше.

Эти отличия заключены в том, что ограничения $R_1, R_2, \ldots, R_m$ не имеют явного задания как конкретные отношения, определенные на основных множествах $A_1, A_2, \ldots, A_s$. Если от процедурного представления ограничений $R_1, R_2, \ldots, R_m$ удается перейти к перечислению кортежей, которые задают хотя бы некоторые из ограничений $R_1, R_2, \ldots, R_m$, то дальнейшая работа с ними практически не отличается от задания древовидными структурами отношений $p_1, \ldots, p_k$ с последующей организацией их вычислимости.

ISSN 2500-2759

Основных проблем обеспечения быстрой вычислимости ограничений $R_1, R_2, \ldots, R_m$ две:

– количество кортежей, которые задают некоторые из ограничений $R_1, R_2, \ldots, R_m$, может быть **очень большим**, что не позволяет все их одновременно держать в оперативной памяти;

– при преобразованиях многоосновной модели необходимо **редактировать совокупности кортежей**, представляющих ограничения $R_1, R_2, \ldots, R_m$, что резко усложняет все технические задачи построения древовидных структур.

Достаточно часто **первая проблема** может быть решена введением так называемых **декубов**, т.е. всеобщих значений (*) элементов из какого-либо основного множества $A_1, A_2, \ldots, A_s$ для кортежей. Расплатой за ввод **декубов** является переход к более сложной древовидной структуре, где вершина (узел) задается в виде

```
struct Tree_decube {
int value;      //значение, приписанное вершине (операция, число, имя и др.)
int down;       //ссылка на первого потомка (–1: отсутствие ссылки);
int right;      //ссылка на следующего брата (–1: отсутствие ссылки);
int decube;     //ссылка на декуб (–1: отсутствие ссылки);
}.
```

Кроме того, сложность вычислений для древовидной структуры с декубами не линейная, а квадратичная.

2.2. Организация проектирования расписания занятий

Задание операторов преобразований многоосновных моделей и организации перебора последовательностей преобразований наиболее выпукло можно представить на примере такой задачи, как планирование расписания учебных занятий, где каждому занятию из $Z \subseteq P \times C \times D$ необходимо сопоставить ресурс, т.е. элемент из $Res: A \times M$ (аудиторию и время проведения занятий). Таким образом, преобразование многоосновной алгебраической системы состоит в построении (формировании) одного кортежа отношения (либо его «уничтожения»). **Организация перебора** преобразований состоит в процедуре выбора **элементов**, которым сопоставляется **ресурс.**

Стратегии оптимизации перебора последовательностей преобразований состоят в возможно более быстром решении в каждой точке пространства поиска трех фундаментальных задач [10; 11]:

– просмотр вперед (**checking forward**) для уменьшения количества применяемых преобразований;

– определение точки возврата для тупика (**intelligent backtracking**, или глубокий возврат по принятой в России у ряда авторов терминологии);

– проверка выполнимости ограничений на многоосновной модели, полученной после выполнения выбранного преобразования (применение демонов для определения невязок — терминология специалистов по системам искусственного интеллекта).

В общем случае это производится модернизацией для древовидных структур следующего базового алгоритма:

```
steck[0] = top; // ИНИЦИАЛИЗАЦИЯ ЗНАЧЕНИЙ СТЕКА
for (i = 0, cur_top = top, offss = 1; i > –1; i = i + offss) {
  for ( ; ; ) {                          // ЦИКЛ ВПРАВО ПО БРАТЬЯМ
   if (offss == –1) {                    // ДВИЖЕНИЕ ВВЕРХ
    cur_top = steck[i];
   if (cur_top == top) { // КОНЕЦ РАБОТЫ, ПОДНЯЛИСЬ ДО КОРНЯ
     i = –2; break;
    }
   } else {
    steck[i]=cur_ top;
    if ((tree + cur_top)–> down > –1)) {
     cur_top = (tree + cur_top)–> down; break; //СПУСТИЛИСЬ ВНИЗ
    }
   }
   if (cur_top < 0) {
    offss=–1; break;
   }
  cur_top = ((tree + cur_top)–> right; // ВПРАВО ПО БРАТЬЯМ
   if (cur_top < 0) {
    offss=–1; break;
   }
   offss=1;
  }
 }
}.
```

В программном комплексе ***МаПа*** стратегии оптимизации переборов дополнены еще двумя стратегиями:

– неполным восстановлением среды точки возврата;

– ограничением глубины построения дерева невязок [3].

Стратегия **неполного восстановления среды точки возврата** позволяет сохранить часть вычислений между тупиком и точкой

ISSN 2500-2759

возврата, что, конечно, повышает скорость поиска решения, но резко усложняет программную реализацию из-за **только** вычислительного отката к точке возврата (нельзя пользоваться фиксацией ситуации в оперативной памяти) и необходимости проверки корректности данных. Другой проблемой является возможность пропуска решений (эта проблема характерна и для стратегии «**чем хуже, тем лучше**» [3]).

Стратегия **ограничения глубины построения дерева невязок** может считаться репликой стратегии «**чем хуже, тем лучше**» [там же] и не имеет особых проблем в реализации.

Заключение

Эффективность применения приведенных здесь методов для решения на больших объемах данных задачи построения качественных расписаний занятий, наверное, объясняется спецификой этой задачи, а именно: каждый объект (преподаватель, контингент учащихся), имеющий сложные ограничения, входит в ограниченное число занятий (ограничение занятий в день и количество дней в неделе), а аудитории, которые потенциально могут использоваться во многих занятиях, не имеют сложных ограничений, а их вместимость и специализацию можно учесть на этапе построения данных для задачи проектирования.

Таким образом, дерево невязок в основном неглубокое и стратегия **ограничения глубины построения дерева невязок** в сочетании со стратегией **неполного восстановления среды точки возврата** работает достаточно хорошо.

Считаю, что данные стратегии можно рекомендовать к использованию в системах искусственного интеллекта при решении конкретных ***NP***-трудных задач большой размерности.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Корольков Ю.Д. Дискретные модели: представление конечными деревьями и разрешимость формальных теорий / Ю.Д. Корольков, В.И. Мартьянов. — Иркутск : Изд-во ИРНИТУ, 2017. — 160 с.

2. Лорьер Ж.-Л. Системы искусственного интеллекта : пер. с фр. / Ж.-Л. Лорьер. — Москва : Мир, 1991. — 486 с.

3. Обзор приложений логико-эвристических методов решения комбинаторных задач высокой сложности / В.И. Мартьянов, В.В. Архипов, М.Д. Каташевцев, Д.В. Пахомов // Современные технологии. Системный анализ. Моделирование. — 2010. — № 4 (28). — С. 61–67.

4. Мальцев А.И. Алгебраические системы / А.И. Мальцев. — Москва : Наука, 1967. — 324 с.

5. Кокорин А.И. Вопросы разрешимости расширенных теорий / А.И. Кокорин, А.Г. Пинус // Успехи математических наук. — 1978. — Т. 33, вып. 2. — С. 49–84.

6. Беллман Р. Динамическое программирование : пер. с англ. / Р. Беллман. — Москва : Изд-во иностр. лит., 1960. — 400 с.

7. Codd E.F. The Relational Model For Database Management: Version 2 / E.F. Godd. — Reading : Addison-Wesley, 1990. — 538 p.

8. Кнут Д.Е. Искусство программирования для ЭВМ : пер. с англ. В 7 т. Т. 3 / Д. Кнут. — Москва : Мир, 1978. — 848 с.

9. Гери М. Вычислительные машины и труднорешаемые задачи / М. Гери, Д. Джонсон. — Москва : Мир, 1982. — 419 с.

10. Щербина О.А. Удовлетворение ограничений и программирование в ограничениях / О.А. Щербина // Интеллектуальные системы. — 2011. — Т. 15, вып. 1-4. — С. 53–170.

11. Hentenryck P. van. Constraint Satisfaction in Logic Programming / P. van Hentenryck. — Cambridge : MIT Press, 1989. — 224 p.

REFERENCES

1. Korolkov Yu.D., Martyanov V.I. *Diskretnye modeli: predstavlenie konechnymi derev'yami i razreshimost' formal'nykh teorii* [Discrete Models: Presentation by Finite Trees and Tractability of Formal Theories]. National Research Irkutsk State Technical University Publ., 2017. 160 p.

2. Laurière J.-L. *Intelligence artificielle — Résolution de problèmes par l'homme et par la machine*. Paris, Eyrolles, 1987. 473 p. (Russ. ed.: Laurière J.-L. *Sistemy iskusstvennogo intellekta*. Moscow, Mir Publ., 1991. 486 p.).

3. Martyanov V.I., Arkhipov V.V., Katashevzev M.D., Pakhomov D.V. Logic-Heuristic Methods for Solving Combinatorial Problems of High Complexity Applications Preview. *Sovremennye tekhnologii. Sistemnyi analiz. Modelirovanie = Modern Technologies. System Analysis. Modeling*, 2010, no. 4 (28), pp. 61–67. (In Russian).

4. Maltsev A.I. *Algebraicheskie sistemy* [Algebraic Systems]. Moscow, Nauka Publ., 1967. 324 p.

5. Kokorin A.I., Pinus A.G. Decidability Problems of Extended Theories. *Uspekhi matematicheskikh nauk = Russian Mathematical Surveys*, 1978, vol. 33, iss. 2, pp. 49–84. (In Russian).

6. Bellman R. *Dynamic Programming*. Princeton University Press, 1957. 365 p. (Russ. ed.: Bellman R. *Dinamicheskoe programmirovanie*. Moscow, Inostrannaya Literatura Publ., 1960. 400 p.).

7. Codd E.F. *The Relational Model For Database Management: Version 2*. Reading, Addison-Wesley, 1990. 538 p.

8. Knuth D.E. *The Art of Computer Programming. Volume 3: Sorting and Searching*. Reading, Addison-Wesley, 1973. 723 p. (Russ. ed.: Knuth D.E. *Iskusstvo programmirovaniya dlya EVM*. Moscow, Mir Publ., 1978. Vol. 3. 848 p.).

ISSN 2500-2759

9. Garey M.R., Johnson D.S. *Computers and Intractability*. San Fransisco, 1979. 338 p. (Russ. ed.: Garey M.R., Johnson D.S. *Vychislitel'nye mashiny i trudnoreshaemye zadachi*. Moscow, Mir Publ., 1982. 419 p.).
10. Shcherbina O.A. Constraint satisfaction and constraints-based programming. *Intellektual'nye sistemy = Intelligent Systems*, 2011, vol. 15, iss. 1-4, pp. 53–170. (In Russian).
11. Hentenryck P. van. *Constraint Satisfaction in Logic Programming*. Cambridge, MIT Press, 1989. 224 p.

Информация об авторе

Мартьянов Владимир Иванович — доктор физико-математических наук, профессор, кафедра автомобильных дорог, Иркутский национальный исследовательский технический университет, профконсультант, Байкальский государственный университет, г. Иркутск, Российская Федерация, e-mail: martvliv@mail.ru.

Для цитирования

Мартьянов В.И. Теоретико-множественный анализ организации данных учебного процесса и алгоритмы проектирования расписания с элементами искусственного интеллекта / В.И. Мартьянов. — DOI: 10.17150/2500-2759.2020.30(4).575-585 // Известия Байкальского государственного университета. — 2020. — Т. 30, № 4. — С. 575–585.

Author

Vladimir I. Martyanov — D.Sc. in Physics and Mathematics, Professor, Department of Highways, Irkutsk National Research Technical University, Professional Consultant, Baikal State University, Irkutsk, the Russian Federation, e-mail: martvliv@mail.ru.

For Citation

Martyanov V.I. Set-theory Analysis of the Data Organization of the Educational Process and Algorithms for Designing a Schedule with Elements Artificial Intelligence. *Izvestiya Baikal'skogo gosudarstvennogo universiteta = Bulletin of Baikal State University*, 2020, vol. 30, no. 4, pp. 575–585. DOI: 10.17150/2500-2759.2020.30(4).575-585. (In Russian).

ISSN 2500-2759