Научная статья
УДК 519.6(082)
EDN: QJQXZL
DOI: 10.21285/2227-2917-2024-3-556-569

Методы эффективной генерации машин Тьюринга, решающих NP-трудные задачи

В.И. Мартьянов[1,2,3]✉, А.С. Волков[2]
[1]Иркутский национальный исследовательский технический университет, г. Иркутск, Россия
[2]Байкальский государственный университет, г. Иркутск, Россия
[3]Иркутский государственный университет, г. Иркутск, Россия

Аннотация. Целью исследования является анализ программной реализации концепции генерации машин Тьюринга, решающих NP-трудные задачи. Формулируется математическая постановка проблемы генерации машин, представленная как решение задач ресурсного планирования для удобства адаптации к программным решениям, используемым для задач проектирования расписания занятий или, более широко, для задач сетевого планирования. Представлены программные решения представления данных, частично использующие технические решения для задачи проектирования расписания занятий. Приведено описание основных стратегий (процедурная семантика методов) сокращения переборов (глубокий возврат, «смотри вперед» и др.). В них также используются технические решения для задачи проектирования расписания занятий и других стратегий (например, применение оракула), которые не использовались для задачи проектирования расписания занятий. Целью исследования является анализ промежуточных результатов программной реализации концепции генерации машин Тьюринга, решающих NP-сложные задачи. Обсуждаются дальнейшие планы по разработке программного комплекса для генерации машин Тьюринга, который может стать открытой платформой для обучения студентов теории алгоритмов и информационным технологиям, проведения всеми желающими экспериментов по определению не полиномиальной сложности решения определенной серии NP-трудных задач, сгенерированными машинами Тьюринга.

Ключевые слова: задачи ресурсного планирования, удовлетворение ограничениям, программирование в ограничениях, глубокий возврат, неполное восстановление среды точки возврата

Для цитирования: Мартьянов В.И., Волков А.С. Методы эффективной генерации машин Тьюринга, решающих NP-трудные задачи // Известия вузов. Инвестиции. Строительство. Недвижимость. 2024. Т. 14. № 3. С. 556–569. https://doi.org/10.21285/2227-2917-2024-3-556-569. EDN: QJQXZL.

Original article

Methods for efficient generation of Turing machines to solve NP-hard problems

Vladimir I. Martyanov[1,2,3]✉, Andrey S. Volkov[2]
[1]Irkutsk National Research Technical University, Irkutsk, Russia
[2]Baikal State University, Irkutsk, Russia
[3]Irkutsk State University, Irkutsk, Russia

Abstract. The present study analyzes the software implementation of the concept of generating Turing machines that solve NP-hard problems. The paper describes a mathematical formulation of machine generation, presented as a solution of resource planning problems to be adapted to software for designing class schedules or, more generally, for network scheduling. The proposed software solutions for data representation are partially based on technical solutions for designing a class schedule. Basic strategies (procedural semantics of methods) for reducing searces (deep return, look ahead, etc.) utilize technical solutions for designing class schedules and other strategies (e.g., application of oracle) that have not been used for this purpose. The study is aimed at analyzing the intermediate results of software

implementation of the concept of generating Turing machines that solve NP-hard problems. The paper proposes further plans for developing a software package for generating Turing machines into an open platform for teaching students the theory of algorithms and information technology, conducting experiments to determine the non-polynomial complexity of solving certain NP-hard problems generated by Turing machines.

Keywords: resource planning tasks, satisfaction of constraints, programming in constraints, deep return, incomplete restoration of the return point environment

For citation: Martyanov V.I., Volkov A.S. Methods for efficient generation of Turing machines to solve NP-hard problems. *Proceedings of Universities. Investment. Construction. Real estate.* 2024;14(3):556-569. (In Russ.). https://doi.org/10.21285/2227-2917-2024-3-556-569. EDN: QJQXZL.

ВВЕДЕНИЕ

Общая концепция создания программного комплекса генерации машин Тьюринга (МТ) рассмотрена в статьях [1–4].

В статье [1] отмечается, что генерация МТ функционирует до некоторой степени подобно генераторам криптовалют [5–8], но формирует МТ, решающие математические задачи.

В данной работе рассмотрены более детально технические вопросы организации:

– данных (лента и запись на ней исходных задач, функций перехода состояний, печати, движения головки) и др.;

– стандартных переборов исходных задач на ленте и МТ;

– стратегии оптимизации переборов как исходных данных, так и МТ.

Кроме того, изучены результаты вычислительных экспериментов и планы решения задач большой размерности.

Также в качестве эталонных NP-сложных задач на ленте МТ записываются задачи покрытия [9].

Сравнительный анализ технических решений стратегий оптимизации переборов при генерации МТ и в задаче проектирования расписания занятий проводится только тогда, когда это возможно.

Генерации расписания занятий и МТ в формате задач ресурсного планирования

Для задачи генерации расписания основными являются множества:

– преподавателей – L = {l_1, l_2, …, l_{kp}};

– контингентов учащихся K = {k_1, k_2, …, k_{kc}};

– предметов P = {p_1, p_2, … p_{kd}};

– аудиторий M = {m_1, m_2, …, m_{ka}};

– пар (двухнедельный или иной цикл) C = {c_1, c_2, …, c_{km}}.

Совокупность планируемых занятий Za ⊆ L x K x P.

Расписание занятий Ih – инъективное отображение Za → Re, где Re: M x C – ресурс.

Таким образом, генерация расписания занятий – важный частный случай задачи ресурсного планирования. Следует отметить, что сетевой план пар (расписание) Ih должен учитывать условия как по группам студентов, так и по лекторам (смены для групп студентов, равномерное распределение занятий по учебной неделе и пр.), а также правила для исполнителей нагрузки (свободные от занятий дни недели или часы, ограниченное число пар в день, принимать во внимание число типов занятий в день, особенно лекций, и пр.).

Компоненты МТ

1. Лента L, разбитая на клетки (ячейки). При выполнении команд программы головка МТ может сдвигаться по ленте влево и право, формируя новые клетки так, что лента L может считаться потенциально бесконечной. В любой из клеток ленты указаны какие-то буквы из алфавита AL = {b_0 , b_1, b_2, …, b_w, …} , т.е. в клетках ленты b_{ij} являются буквами алфавита AL. При выполнении команд программы головка МТ движется по клеткам ленты L и буквы могут изменяться и не изменяться. Ленту L определяют также как внешнюю память МТ (в отличие от состояния, которое определяют как внутреннюю память МТ). При переходе на новые клетки слева или справа считаем, что в ней находится одна и та же буква алфавита AL = {b_0 , b_1, b_2, …, b_w, …}, а именно b_0, предполагая в данном случае клетку пустой.

2. Совокупность состояний МТ. QQ = {qq_0, qq_1, qq_2, …, qq_t}, составляющие эту совокупность называются внутренней памятью. Будем считать, что МТ постоянно существует лишь в единственном состоянии, поэтому определяется как детерминированная, но также используются не детерминированные МТ, которые могут иногда или постоянно использовать больше, чем одно состояние. Состояние qq_0 определяется как конечное (или стоп-состояние) при возникновении которого МТ стопорится.

3. Головка МТ читает букву одной клетки ленты L.

4. Правила функционирования МТ. На каждом шаге деятельности МТ головка воспринимает из клетки ленты букву алфавита МТ и записывает в данную клетку некоторую букву алфавита AL = {b_0, b_1, b_2, …, b_w, …}, а затем смещает головку МТ в ближайшую (левую или правую) клетку. МТ получает то или иное состояние из совокупности QQ = {qq_0, qq_1, qq_2, …, qq_t}. Следует учитывать, что МТ может зацикливаться, т. к. стоп-состояние qq_0 может оказаться недостигаемым. Сущностью (машинным словом) МТ определим множество, образованное словом aa_{j1}, aa_{j2}, …, букв всех клеток ленты, буквой qq_i и номером kk считываемой клетки aa_{jk}. Следовательно, каждая сущность (машинное слово) имеет всего одно применение буквы состояния qq_i. Это означает, что данное машинное слово (сущность) буква qq_i стоит на позиции kk, что можно представить словом

$$aa_{j1}\ aa_{j2}\ \ldots\ aa_{j\,k-1}\ qq_i\ aa_{i\,k}\ \ldots\ aa_{js} \qquad (1)$$

Напомним, что всякая сущность (машинное слово) обязательно в конце имеет букву алфавита AL = {b_0 , b_1, b_2, …, b_w, …} , но иногда начинается буквой из QQ = {qq_0, qq_1, qq_2, …, qq}.

5. Функции перехода, печати и сдвига:

– переход состояний: Trans : (QQ x AL) → QQ;

– печать: Write : (QQ x AL) → AL;

– сдвиг головки: Move : (QQ x AL) → {-1, 1}.

Изучим изменения формулы (1) при использовании функций Trans, Write, Move. Пусть Trans (qq_i, $aa_{i\,k}$) = qq_ξ, Write (qq_i, $aa_{i\,k}$) = aa_λ, Move (qq_i, $aa_{i\,k}$) = tt. Если tt = -1, то это производит сдвиг головки по ленте влево, а если tt = 1, то сдвиг головки вправо. Таким образом, получаем следующие два случая.

1. Пусть tt = -1. Тогда формула (1) будет переделана в слово $aa_{j1}\ aa_{j2}\ \ldots\ qq_\xi\ aa_{j\,k-1}\ aa_\lambda\ \ldots\ aa_{js}$

2. Пусть tt = 1. Тогда формула (1) будет переделана в слово $aa_{j1}\ aa_{j2}\ \ldots\ aa_{j\,k-1}\ aa_\lambda\ qq_\xi\ aa_{j\,k+1}\ \ldots\ aa_{js}$.

Отображения Trans, Write, Move определяют программу МТ и исходную ленту L – задачей. Лента МТ позволяет формулировать любые классические NP-трудные проблемы:

– приблизительные вычисления;

– маршрутизация;

– планирование и др. [10].

При постановке задачи генерации МТ в формате задач ресурсного планирования рассматривается построение отображений, удовлетворяющих ограничениям для функций Trans, Write, Move, получаемых из F естественным образом.

$$F: (QQ \times AA) \rightarrow (QQ \times AA \times \{-1, 1\}) \qquad (2)$$

Методы генерации МТ [11] можно применять для ускорения перебора входных данных (задачи на ленте МТ), а также для ускорения перебора программ МТ (функций Trans, Write, Move), что показывает возможность считать проблему генерации МТ еще одним примером важного частного случая задачи ресурсного планирования.

Генерация производится для МТ со следующими параметрами:

1. Алфавит xo_alphabet.slw = {0; 1; ☼; ◙; #; 5; 6}. Количество букв в алфавите Alp = 7. Буква ☼ – начальный символ, который стоит на нулевой позиции ленты и не изменяется при работе МТ (головка МТ не сдвигается влево от нулевой позиции); ◙ – конец формирования исходных данных; # – разделитель отрезков из 0 и 1, выборка из которых может формировать покрытие (отрезок, состоящий только из 1).

2. Словарь состояний xo state.slw = {q_begin; q0, q1, q2, q3}; q_begin – начальное состояние; q0 – стоп (конец работы); q1, q2, q3 – без свойств. Количество состояний Sta = 5.

3. Словарь сдвигов головки xo displace.slw = {1; -1}; 1 – сдвиг головки вправо; -1 – сдвиг головки влево; количество сдвигов k_displ = 2.

В статье [10] отмечается, что генерация программ, хотя бы только для МТ, является принципиально новым шагом, а в данной работе показаны результаты, которые могут быть значительно продвинуты вперед в ближайшее время.

Постановка проблемы генерации МТ в формате задач ресурсного планирования

Определяются основные программные конструкции представления данных, управляющих структур, приемов минимизации переборов и другие способы решения проблем ресурсного планирования.

В частности, это делается для алгоритмического представления следующих данных и процедур проверки выполнимости:

– множеств элементов (доменов реляционной БД);

– отношений, связывающих элементы множеств (таблиц реляционной БД);

– определения ограничений для кортежей множеств и средств определения истинности ограничений;

– определения процедур преобразований исходных данных и построения эффективных переборов наборов преобразований;

– стратегии оптимизации перебора наборов преобразований.

Представление множества элементов и структурных конструкций

Отметим, что в алгоритмическом представлении множества элементов и управляющие структуры реализуются одинаковыми структурами, а именно «деревьями», что гарантирует применение концепции «обратной связи», т.е. исходные или промежуточные данные постановки задачи можно преобразовывать в структуры управления и наоборот. Отметим, что принцип «обратной связи» является очень важным для кибернетики, а также для систем искусственного интеллекта. Программе генерации сетевых планов необходимо работать и с примитивной организацией данных. А именно, для памяти – выделение конкретных частей: куч (heap), а для внешней памяти – системы файлов, т. к. в системах управления базами данных практически невозможно хранить конструкции алгоритмов планирования решения задач, основанных на предлагаемых в работе [10] методах. В дальнейшем будем применять команды языка программирования C++, calloc или new, определяющие по дескриптору pointer часть машинной памяти – heap.

Для внешней памяти файлы создаются командой C++ Discrip = _creat(const char *File, 0), где файл с именем File имеет указатель файла Discrip. Команды считывания – _read, фиксации – _write, сдвига дескриптора – lseek.

В дальнейшем части программного кода, показывающие организацию задания исходных данных, программные конструкции задания структур планирования расчетов и задач, будут представлены на C++ или подобном языке программирования.

Основной программной конструкцией для задания данных и структур планирования расчетов будут «деревья» (tree) [12, 13]. Также будут применяться и классические методы обработки низкоуровневой организации данных:

– хеширование;

– сортировка;

– двоичный поиск и др.

Программная конструкция, определяющая узел дерева:

```
struct Tree {
        int value;      //число, соответствующее узлу (операция, наименование и др.)
        int down;       //адрес узла вниз (-1: нет пути вниз);
        int right;      // адрес узла вправо  (-1: нет пути вправо);
        };
```

Тупиковое состояние движения на другой узел для элементов дерева down и right задается числом -1.

Определение kk узлов структуры Tree по указателю trree задается оператором

```
trree = (struct Trree  *)
calloc (kk, sizeof(struct Trree));        (3)
```

Оператор sizeof задает раздел памяти, хватающий для задания узлов struct Trree. Таким образом, оператор (3) задает по указателю trree область величиной в kk * sizeof(struct Trree).

При работе с данными и структурами управления генерации решения ресурсных задач, представленных «деревьями», важное значение имеют следующие базовые алгоритмы:

– проход структуры (tree) «в глубину – направо»;

– установка включения (или наличия общих вершин) у данных «деревьев»;

– преобразования «деревьев» (замена значений переменных, перестройка и изменения «поддеревьев» на другие, изменения, связанные с выполнением вычислительных операций и по вопросам, возникающим с выполнением технологических преобразований для уменьшения возвратов, генерации графиков), такие как «просмотр вперед», «глубокий возврат», «быстрая проверка ограничений» и др. [12].

Отметим следующие положительные свойства «деревьев»:

1. При генерации расписаний и МТ «деревья» применяют как для формирования данных, так и для управляющих структур алгоритмов.

2. Динамическая интерпретация и преобразование структуры данных выполняется с высокой экономией вычислительных затрат и почти всегда при минимальных затратах на выделение дополнительной памяти.

3. Очищение оперативной памяти от отработанных данных и структур управления, представленных «деревьями», выполняется тоже достаточно просто.

Далее рассмотрим представление данных (основных множеств), отношений и ограничений, как программных конструкций для математических моделей, определенных следующим образом:

$MM = < AA_1, AA_2, \ldots, AA_s; pp_1, \ldots, pp_k >$,

где $AA_1, AA_2, \ldots, AA_s$ – домены (основные множества) и $pp_1, \ldots, pp_k$ – таблицы (отношения или предикаты), причем также заданы рестрикции (ограничения) – $RR_1, RR_2, \ldots, RR_m$. Следует отметить, что описанная математическая модель ММ является удобным формализмом для изложения математических основ эффективных стратегий решения задач ресурсного планирования. Например, отображение (2) может быть задано математической моделью MM_1 = < QQ, AA, {-1, 1}; Trans1, Write1,

Move1 >, где отношения Trans1, Write1 и Move1 соответствуют функциям Trans, Write и Move, а ограничения R_1, R_2,…, R_k определят необходимые требования для математической модели MM_1.

Определение доменов (основных множеств) AA_1, AA_2, …, AA_s математической модели ММ реализуется определением разделов памяти (heap или куч):

SS 1 = (struct Seet 1 *) calloc (kk_1, sizeof(struct Seet 1)),
SS 2 = (struct Seet 2 *) calloc (kk_2, sizeof(struct Seet 2)),
SS s = (struct Seet s *) calloc (kk_s, sizeof(struct Seet s)),

где, соответственно, конструкции Seet 1, Seet 2, …, Seet s определяют основные качества элементов доменов AA_1, AA_2, …, AA_s, соответственно, kk_1, kk_2, …., kk_s – размерности доменов AA_1, AA_2, …, AA_s.

Определение предикатов pp_1, …, pp_k математических моделей в компьютере реализуется заданием, соответственно, разделов памяти (heap или куч):

Rt 1 = (struct Rit 1 *) calloc (m_1, sizeof(struct Rit 1)),
Rt 2 = (struct Rit 2 *) calloc (m_2, sizeof(struct Rit 2)),
Rt k = (struct Rit k *) calloc (m_k, sizeof(struct Rit k)),

где, соответственно, конструкции Rit 1, Rit 2, …, Rit k используются для элементов декартова произведения множеств AA_1, AA_2, …, AA_s, образующих предикаты pp_1, …, pp_k, а числа m_1, m_2, …, m_k задают количество элементов декартова произведения в pp_1, …, pp_k, определенных на основных множествах AA_1, AA_2, …, AA_s, т. е. предполагаем, что отношения выстраиваются следующим образом:

$$pp_1, \ldots, pp_k \subseteq AA_1 \times AA_2 \times \ldots \times AA_s.$$

Определение рестрикций (ограничений) RR_1, RR_2, …, RR_m, которые обязаны исполняться на математической модели ММ, и алгоритмы их проверки

Приводимые в предшествующем пункте алгоритмы не подходят для этого, т. к. рестрикции RR_1, RR_2, …, RR_m не представлены в явном виде как предикаты, заданные на доменах AA_1, AA_2, …, AA_s.

Главные проблемы эффективной проверки рестрикций RR_1, RR_2, …, RR_m две:

– количество кортежей, которые задают некоторые из ограничений RR_1, RR_2, …, RR_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 RR_1, RR_2, …, RR_m, что 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 AA_1, AA_2, …, AA_s для кортежей. Из-за ввода декубов возникает 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);
};
```

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

Полученная древовидная структура называется «деревом решения» и позволяет проверить все ограничения одновременно, т. е. перебор по проверке ограничений исчезает.

Определение функционалов перестройки моделей и формирование подбора планов действий наглядно можно показать на задаче генерации графика пар по дневной форме обучения (расчет расписания занятий), что была подробно расписана выше.

Совокупность планируемых занятий $Za \subseteq L \times K \times P$. Расписание занятий Ih – инъективное отображение $Za \rightarrow Re$, где $Re: M \times C$ – ресурс. Таким образом, любой паре из $Za \subseteq L \times K \times P$ нужно сопоставить ресурс, т. е. элемент из $Re: M \times C$ (аудиторию и время проведения занятий).

Следовательно, преобразование многоосновной модели состоит в построении (формировании) одного кортежа отношения или 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:

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

– определению точки возврата для тупика (intelligent backtracking или глубокий возврат по принятой у ряда российских авторов терминологии);

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

Оптимизация переборов для этих трех задач для каждой предметной области производится весьма специфическим образом, но для рассматриваемых в статье задач есть общие черты, которые будут описаны ниже.

Оптимизация, реализуемая просмотром вперед (checking forward) для уменьшения количества применяемых преобразований, достигается уменьшением ресурса при применении каждого преобразования (большинство задач сетевого планирования происходит при убывающем ресурсе).

Для других задач используется стратегия «первой неудачи» (first fail) [12], которая, как правило, существенно уменьшает пространство перебора следующего шага решения.

Оптимизация, реализуемая определением точки возврата для тупика (intelligent backtracking или глубокий возврат), для задач сетевого планирования с убывающим ресурсом достигается рассмотрением либо правой точки, либо левой точки, где произошло преобразование с объектами тупика.

Если возврат осуществляется с правой точки, то это соответствует стандартному глубокому возврату, а если с левой точки, то глубокому возврату со стратегией «чем хуже, тем лучше», где решение может быть найдено быстрее, но данная стратегия не полна (может пропустить возможные решения, что для задачи генерации МТ неприемлемо). В рамках стратегии неполного восстановления среды точки возврата откат может осуществляться к точке пространства состояний, соответствующей применению слова $w_1 \dots w_s u_1 \dots u_n$, где слово $u_1 \dots u_n$ входит с точностью до вычеркивания букв в слово $w_{s+1} \dots w_v$.

Такой подход значительно усложняет многие технические моменты программной реализации метода как в плане обеспечения корректности решения, так и в организации данных, фиксирующих траекторию решения, сбор мусора и т. п. Но отключение этой возможности (неполное восстановление среды точки возврата) не позволяет в реальное время решать, например, задачу планирования расписания занятий.

Данная стратегия (неполное восстановление среды точки возврата) может применяться и в задаче генерации МТ, но пока оценка эффективности ее применения и возможности пропуска решений не проведена.

Основные стратегии ускорения генерации МТ в формате задач ресурсного планирования

Пусть генерация МТ, решающих NP-трудные задачи покрытия, производится при фиксированном алфавите A = {0; 1; ☼; ◙; #; 5; 6}, где ☼ – начальный символ, который стоит на нулевой позиции ленты и не изменяется при работе МТ (головка МТ не сдвигается влево от нулевой позиции); ◙ – конец формирования исходных данных; # – разделитель отрезков из 0 и 1, выборка из которых может формировать покрытие (отрезок, состоящий только из 1).

Пример исходных данных размерности (2, 2) задан табл. 1. Общий вид исходных данных размерности (m, n), где m – длина отрезков из 0 и 1, а n – количество отрезков из 0 и 1, может быть задан табл. 2, где $a_{i,j}$ являются 0 или 1.

Таблица 1. Исходные данные размерности (2, 2)
Table 1. The initial data of dimension (2, 2)

Позиция на ленте	0	1	2	3	4	5	6
Буквы алфавита	☼	0	0	#	0	0	◙

Таблица 2. Исходные данные размерности (m, n)
Table 2. Initial data of dimension (m, n)

№ строки	Буквы алфавита	Буквы алфавита	Буквы алфавита	…	Буквы алфавита	Буквы алфавита
1	☼	$a_{1,1}$	$a_{1,2}$	…	$a_{1,m}$	#
2		$a_{2,1}$	$a_{2,2}$	…	$a_{2,m}$	#
…	…	…	…	…	…	…
n - 1		$a_{n-1,1}$	$a_{n-1,2}$	…	$a_{n-1,m}$	#
n		$a_{n,1}$	$a_{n,2}$	…	$a_{n,m}$	◙

По определению данные табл. 2 имеют покрытие, если выборка строк с номерами i_1, i_2, …, i_w для любого столбца с номером k выполняется:

$$a_{i1,k} + a_{i2,k} + \dots + a_{iw,k} = 1. \quad (4)$$

Отметим, что, если для исходных данных выборка с условием (4) существует, то сгенерированная МТ должна оканчивать работу успехом (останавливаться при получении состояния q_0), а в противном случае должна работать бесконечно. Начальные данные размерности (m, n) представлены в табл. 3.

При сдвиге данные табл. 2 перестраиваются следующим образом.

Пусть элемент $a_{i,j}$ является первым 0 при проходе от $a_{n,m}$ налево (и вверх при необходимости). Тогда все пройденные элементы таблицы зануляются, а элемент $a_{i,j}$ превращается в 1. Все остальные элементы таблицы остаются неизменными. Таким образом, если изначальный вариант был как в табл. 4, то данные размерности (m, n) после сдвига будут как в табл. 5.

Таблица 3. Начальные данные размерности (m, n)
Table 3. Initial data of dimension (m, n)

№ строки	Буквы алфавита	Буквы алфавита	Буквы алфавита	…	Буквы алфавита	Буквы алфавита
1	☼	0	0	…	0	#
2		0	0	…	0	#
…	…	…	…	…	…	…
n - 1		0	0	…	0	#
n		0	0	…	0	◙

Таблица 4. Исходные данные размерности (m, n)
Table 4. Initial data of dimension (m, n)

№ строки	Буквы алфавита	Буквы алфавита	Буквы алфавита	…	Буквы алфавита	…	Буквы алфавита	Буквы алфавита
1	☼	$a_{1,1}$	$a_{1,2}$	…	$a_{1,j}$	…	$a_{1,m}$	#
2		$a_{2,1}$	$a_{2,2}$	…	$a_{2,j}$	…	$a_{2,m}$	#
…	…	…	…	…	…	…	…	…
i		$a_{i,1}$	$a_{i,2}$	…	$a_{i,j}$	…	$a_{i,m}$	
…	…	…	…	…	…	…	…	…
n - 1		$a_{n-1,1}$	$a_{n-1,2}$	…	$a_{n-1,2}$	…	$a_{n-1,m}$	#
n		$a_{n,1}$	$a_{n,2}$	…	$a_{n,2}$	…	$a_{n,m}$	◙

Таблица 5. Данные размерности (m, n) после сдвига
Table 5. Dimension data (m, n) after the shift

№ строки	Буквы алфавита	Буквы алфавита	Буквы алфавита	…	Буквы алфавита	…	Буквы алфавита	Буквы алфавита
1	☼	$a_{1,1}$	$a_{1,2}$	…	$a_{1,j}$	…	$a_{1,m}$	#
2		$a_{2,1}$	$a_{2,2}$	…	$a_{2,j}$	…	$a_{2,m}$	#
…	…	…	…	…	…	…	…	…
i		$a_{i,1}$	$a_{i,2}$	…	1	…	0	
…	…	…	…	…	…	…	…	…
n - 1		0	0	…	0	…	0	#
n		0	0	…	0	…	0	◙

Таблица 6. Данные размерности (m, n) конечные
Table 6. Finite dimension data (m, n)

№ строки	Буквы алфавита	Буквы алфавита	Буквы алфавита	…	Буквы алфавита	…	Буквы алфавита	Буквы алфавита
1	☼	1	1	…	1	…	1	#
2		1	1	…	1	…	1	#
…	…	…	…	…	…	…	…	…
i		1	1	…	1	…	0	#
…	…	…	…	…	…	…	…	…
n - 1		1	1	…	1	…	0	#
n		1	1	…	1	…	0	◙

Если на сдвиг данных вышли при таблице, состоящей из единиц, как представлено в табл. 6, то получена МТ, решающая задачу покрытия размерности (m, n), а это значит, что

можно осуществлять переход на генерацию следующей МТ.

Описан стандартный сдвиг, но при генерации МТ он может быть существенно более глубоким, а именно, благодаря тому, что МТ для получения ответа о существования покрытия прошла только часть данных. И тогда 0 ищется только в пройденной части и, соответственно, сдвиг будет не стандартным, а глубоким.

Переходим к рассмотрению стандартных сдвигов при генерации МТ, имеющих, как было указано выше, алфавит xo_alphabet.slw = {0; 1; ☼; ◙; #; 5; 6}, словарь состояний xo_state.slw = {q_begin; q0, q1, q2, q3}, словарь сдвигов головки xo_displace.slw = {1; -1}.

Следующая табл. 7 определяет программу МТ. Таким образом, четвертая строка табл. 7 с номером команды i определяет следующие команды МТ:

– печать: Write (a_i, s_i) = aa_i;

– переход состояния: Trans (a_i, s_i) = ss_i;

– движение головки: Move (a_i, s_i) = dd_i.

Если для буквы a и состояния s нет соответствующей строки в табл. 7, то это записывается следующим образом:

$$\text{Write}(a, s) = ?;\ \text{Trans}(a, s) = ?;\ \text{Move}(a, s) = ?. \quad (5)$$

Генерация МТ начинается со следующей исходной позиции (табл. 8).

Таблица 7. Программа МТ
Table 7. MT Program

№ команд	Аргументы команд			Значения команд		
	Буква	Состояние		Буква	Состояние	Сдвиг
1	a_1	s_1	→	aa_1	ss_1	dd_1
2	a_2	s_2	→	aa_2	ss_2	dd_2
…	…	…		…	…	…
i	a_i	s_i	→	aa_i	ss_i	dd_i
…	…	…		…	…	…
n	a_n	s_n	→	aa_n	ss_n	dd_n

Таблица 8. Программа исходной для генерации МТ
Table 8. The source program for generating MT

№ команд	Аргументы команд			Значения команд		
	Буква	Состояние		Буква	Состояние	Сдвиг
1	☼	q_begin	→	☼	q_begin	1

Таким образом, команды печати, перехода состояний и движения головки определены только для пары (☼, q_begin). Отметим также, что всегда:

☼	*	→	☼	q_begin	1

где * – любое состояние и, таким образом, лента имеет на нулевой позиции всегда букву ☼, а головка не может уходить влево от нулевой позиции (лента ограничена слева нулевой позицией).

При генерации (перестройки) МТ для решения NP-трудной задачи покрытия реконструкция МТ, заданной программой по табл. 7, может производиться следующими двумя способами:

– пополнением программы МТ новой командой;

– перестройкой программы МТ.

Пополнение программы МТ новой командой производится в случае, когда она, находясь в состоянии s считывает букву a, а в табл. 7 нет в аргументах команд пары (a, s), т. е. имеет место (5). Тогда табл. 7 перестраивается в табл. 9.

Таблица 9. Программа МТ после пополнения
Table 9. MT program after replenishment

№ команд	Аргументы команд			Значения команд		
	Буква	Состояние		Буква	Состояние	Сдвиг
1	a_1	s_1	→	aa_1	ss_1	dd_1
2	a_2	s_2	→	aa_2	ss_2	dd_2
…	…	…		…	…	…
i	a_i	s_i	→	aa_i	ss_i	dd_i
…	…	…		…	…	…
n	a_n	s_n	→	aa_n	ss_n	dd_n
n+1	a_{n+1}	s_{n+1}	→	aa_{n+1}	ss_{n+1}	dd_{n+1}

где $a_{n+1} = a$, $s_{n+1} = s$, $aa_{n+1} = 0$, $ss_{n+1} = q\_begin$, $dd_{n+1} = 1$.

Более наглядно n+1 строку можно отобразить таблицей.

a	S	→	0	q_begin	1

Перестройка (один шаг перебора вариантов) программы МТ производится для одной из строк табл. 7. Выбор строки табл. 7 отдельный сложный вопрос, который рассматривается в дальнейшем в стратегиях ускорения генерации МТ.

Стандартный шаг перестройки программы МТ производится для последней сформированной команды

n	a_n	s_n	→	aa_n	ss_n	dd_n

по следующим правилам: если dd_n не является последним элементом в словаре сдвигов головки xo_displace.slw = {1; -1}, то

n	a_n	s_n	→	aa_n	ss_n	-1

если dd_n является последним элементом в словаре сдвигов головки, а ss_n не является последним элементом в словаре состояний xo_state.slw = {q_begin; q0, q1, q2, q3}, то ss_n заменяется на следующий элемент $\underline{ss}_n$ и таким образом получаем

n	a_n	s_n	→	aa_n	$\underline{ss}_n$	-1

Если же ss_n является последним элементом в словаре состояний xo_state.slw = {q_begin; q0, q1, q2, q3}, т.е. $ss_n = q3$, то ss_n заменяется на q_begin, и получаем

n	a_n	s_n	→	aa_n	q_begin	-1

и далее надо сдвигать aa_n на следующий элемент $\underline{aa}_n$ алфавита xo_alphabet.slw = {0; 1; ☼; ◙; #; 5; 6}, причем элемент алфавита ☼ пропускается (конечно, если n больше 1) и в этом случае

n	a_n	s_n	→	$\underline{aa}_n$	q_begin	-1

Если же $aa_n = 6$, то переходим к работе с МТ с программой.

Таблица 10. Программа МТ, где n-ая строка (команда) сокращена
Table 10. The MT program, where the nth line (command) is shortened

№ команд	Аргументы команд			Значения команд		
	Буква	Состояние		Буква	Состояние	Сдвиг
1	a_1	s_1	→	aa_1	ss_1	dd_1
2	a_2	s_2	→	aa_2	ss_2	dd_2
…	…	…		…	…	…
i	a_i	s_i	→	aa_i	ss_i	dd_i
…	…	…		…	…	…
n-1	a_{n-1}	s_{n-1}	→	aa_{n-1}	ss_{n-1}	dd_{n-1}

Таким образом, описанный выше процесс необходимо запускать для n-1 строки МТ, программа которой задана табл. 10. Если будет сокращена и 1-ая строка, то процесс генерации МТ будет закончен для данной размерности МТ (количество букв в алфавите xo_alphabet.slw и словарь состояний xo_state.slw) и данных размерности (m, n).

Нестандартный шаг перестройки программы МТ может производиться с любой i-ой команды и тогда для МТ, заданной табл. 7 по стандартной перестройке, надо работать с МТ, заданной табл. 11. Далее, работа с МТ, заданной табл. 11 идет по стандартной перестройке, описанной выше.

Нестандартный шаг перестройки при генерации МТ является наиболее мощным механизмом сокращения переборов, но и наиболее опасным в плане пропуска МТ, решающих задачи покрытия, поэтому на настоящем этапе использование стратегии неполного восстановления среды точки возврата исключается (при сокращении переборов данных указанная стратегия может использоваться).

Таблица 11. Программа МТ, где строки (команды) сокращены до i
Table 11. The MT program, where the lines (commands) are shortened to i

№ команд	Аргументы команд			Значения команд		
	Буква	Состояние		Буква	Состояние	Сдвиг
1	a_1	s_1	→	aa_1	ss_1	dd_1
2	a_2	s_2	→	aa_2	ss_2	dd_2
…	…	…		…	…	…
i	a_i	s_i	→	aa_i	ss_i	dd_i

Далее будут рассмотрены основные стратегии ускорения генерации МТ при постановке задачи в формате ресурсного планирования.

Следует заметить, что эти стратегии были опробованы на реализации следующих проектов:

– системы управления региональной сетью автомобильных дорог (СУРАД) Иркутской области [14];

– генерации графа автомобильных дорог для системы взимания платы с большегрузного транспорта [15];

– программы сетевого планирование содержания сети автомобильных дорог Иркутской области [16];

– теоретико-множественные модели данных в задаче расчета вторичных структур РНК [17];

– программы планирование работ по нормативному содержанию сети автодорог [18];

– программы обработки видеорядов, снятых дорожными лабораториями на автодорогах [19];

– программы расчета (генерации) расписания [20], используемой с 2006 г. в Иркутском научно-исследовательском техническом университете.

Аналоги организации данных и требования к расчету можно найти в итоговых результатах расчета расписания.

Глубокий (глобальный) возврат по 1-ой решетке (Fir_Grid)

Структура данных struct Grid_new (gridd_new): файл Fi_Grid.fil

Верш ДВЛ (Tr_Full_L) ;

№ иниц ленты (inii_lent);

Позиция на ленте (poza);

№ началь отрезка иниц ленты (N_ini);

№ началь отрезка текущ ленты (N_cur 1 - 8);

№ началь отрезка текущ ленты (N_cur1 9 - 16);

№ началь отрезка текущ ленты (N_cur2 17 - 24);

Оракул № позы # где покрытие; -1 нет (oracl);

Шаг решения (vaar).

Ссылка на след (refe).

Кол-во вершин: k_Fir_Grid ; 1 free: fr_Fir_Grid

int jj0; // = (poi_grid + i_u)->poza;

jj2 = 2 - не совпад oracl (1 –совпад);

jj3 = 2 - не совпад N_ini (1 –совпад);

jj4 = 2 - не совпад N_cur (1 –совпад).

Данная стратегия обеспечивает до 10 % всех сокращений переборов.

Применение оракула (oracle)

При формировании данных, которые надо проверять, важное значение имеет знание о том, существует ли для них покрытие.

Действительно, если сгенерированная МТ зацикливается на данных, где покрытие существует, то надо генерировать следующую МТ, причем сдвиг может быть нестандартным (по результатам анализа процесса работы МТ).

Если же сгенерированная МТ достигает успеха на данных, где покрытие не существует, то надо генерировать следующую МТ, причем сдвиг может быть также нестандартным (по результатам анализа процесса работы МТ).

При формировании данных проверку, существует ли для них покрытие, осуществляет модуль oracle, что существенно ускоряет генерацию. Использование модуля oracle обеспечивает до 15 % всех сокращений переборов.

Применение стратегии «смотри вперед»

Если на каком-то этапе сгенерированная МТ W получает ленту, которая одинакова с лентой ранее рассмотренной МТ V, и можно построить такие исходные данные, что МТ V и W получают противоположные результаты, то надо генерировать (глубокий возврат) следующую МТ, причем сдвиг практически всегда нестандартный.

Данная стратегия обеспечивает до 20 % всех сокращений переборов, но требует весьма сложной подготовки организации данных.

Для примера представлен небольшой фрагмент табл. 12, загружаемой в оперативную память для отслеживания возможности применения данной стратегии.

Другие стратегии

Из других стратегий отметим:

– алгоритмы редактирования шагов решения между тупиковой точкой и точкой возврата (обеспечивают глубокий возврат без полного восстановления среды точки возврата);

– структуры управления, фиксирующие траекторию решения и обеспечивающие откат назад на любой шаг решения (т.е. обеспечивающие полное восстановление среды точки возврата);

– алгоритмы проверки выполнимости ограничений на каждом шаге решения («демоны»);

– алгоритмы полного восстановления среды точки возврата;

– алгоритмы сдвига в выборе следующего элемента ресурса;

– алгоритмы распознавания тупика и определения точки возврата (обеспечивают глубокий возврат).

Таблица 12. Часть таблицы мониторинга стратегии «смотри вперед»
Table 12. Part of the "look ahead" strategy monitoring table

i u 3							8						9				9		1
1-2	00	00	00	00	00	00	00	00		10	10	10	10	10	10	10	10		2
3-4	01	0*	00	0*	0*	0*	01	0*		0*	0*	0*	01	0*	0*	0*	01		3
m u							6						3				7		4
1-2	00	10	01	11	00	10	01	11		00	10	01	11	00	10	01	11		5
3-4	00	0*	0*	0*	1*	1*	10	1*		0*	0*	0*	00	10	10	10	10		6
																			7
i u 3			10										11				??		8
1-2	01	01	01	01	01	01	01	01		11	11	11	11	11	11	11	11		9
3-4	0*	0*	01	0*	0*	0*	0*	0*		0*	0*	0*	01	0*	0*	0*	01		10
m u			2										3						11
1-2	00	10	01	11	00	10	01	11		00	10	01	11	00	10	01	11		12

ЗАКЛЮЧЕНИЕ

Из всего вышесказанного можно сделать следующие выводы:

1. На настоящем этапе генерация МТ достигнута для размерности исходных данных при ограничении количества состояний МТ до 5 включительно. Для обеспечения генерации всех МТ необходимо увеличить количество состояний до 12 включительно, что пока является предметом исследований (уже есть успешные просчеты для такого количество состояний при некоторых ограничениях на организацию исходных данных).

2. Предположительно, генерация всех МТ при количестве состояний до 12 включительно будет достигнута при использовании многопоточных вычислений для этапа перебора всех исходных данных для сгенерированной МТ, что может быть достигнуто в ближайшем будущем (предположительно во втором полугодии 2024 г.).

Кроме того, автор предполагает, что данный подход является расширенной концепцией программирования в ограничениях [21, 22] и имеет шансы стать одним из возможных путей отрицательного решения проблемы P =? NP [23–25], так как проверять конкретные МТ на предмет алгоритмической сложности решения ими задачи покрытия, возможно, проще, чем доказывать соответствующую теорему.

После чего, как отмечалось в статье [1], можно будет приступить к практической реализации «…создания открытой платформы для обучения студентов теории алгоритмов и информационным технологиям; проведения всеми желающими экспериментов по определению не полиномиальной сложности решения определенной серии задач, сгенерированными машинами Тьюринга; участия всех желающих в генерации машин Тьюринга и решении NP-трудных задач на личных компьютерах».

А также, проводить, как отмечалось в статье [1], «…эксперименты по созданию решателей для других NP-трудных задач, коих, имеющих большое прикладное значение, очень и очень много».

Например, приблизительные вычисления, конфигурация, криптография, сбор данных, поддержка при принятии решения, филогенетика, планирование, расписания, маршрутизация.

СПИСОК ИСТОЧНИКОВ

1. Мартьянов В.И. Проект программного комплекса генерации машин Тьюринга, решающих NP-трудные задачи // Известия вузов. Инвестиции. Строительство. Недвижимость. 2023. Т. 13. № 2. С. 285–297. https://doi.org/10.21285/2227-2917-2023-2-285-297. EDN: UAEWDT.
2. Мартьянов В.И. Логико-эвристические методы сетевого планирования и распознавание ситуаций // Проблемы управления и моделирования в сложных системах. Самара, 2001. С. 203–215.
3. Мартьянов В.И., Архипов В.В., Каташевцев М.Д., Пахомов Д.В. Обзор приложений логико-эвристических методов решения комбинаторных задач высокой сложности // Современные технологии. Системный анализ. Моделирование. 2010. № 4 (28). С. 205–211. EDN: NRBKXP.
4. Мартьянов В.И., Сухорутченко В.В., Окунцов В.В. Планирование информационных потоков в иерархической системе // Прикладные системы. М.: ИАП РАН, 1992. С. 46–58.
5. Лагутенков А.В. Криптовалюты. Правила применения // Наука и жизнь. 2018. № 2. С. 22–26.
6. Kharpal A. All You Need to Know About the Top 5 Cryptocurrencies // Yahoo! Finance. 2017. Режим доступа: https://finance.yahoo.com/news/know-top-5-cryptocurrencies-093231173.html (дата обращения: 20.05.2024).

7. Wilson-Nunn D., Zenil H. On the Complexity and Behaviour of Cryptocurrencies Compared to Other Markets // Cornell University: arxiv. 2014. P. 1–16. https://doi.org/10.48550/arXiv.1411.1924.
8. Еремеев А.В., Заозерская Л.А., Колоколов А.А. Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования // Дискретный анализ и исследование операций. 2000. Т. 7. № 2. С. 22–46. EDN: IBBFDL.
9. Мартьянов В.И., Вяткин И.В., Могильницкий Э.Ю. Теоретический и реализационный аспекты некоторых классов задач сетевого управления // Проблемы управления и моделирования в сложных системах. Самара, 1999. С. 203–208.
10. Мартьянов В.И. NP-трудные задачи: автоматическое доказательство теорем и машины Тьюринга // Baikal Research Journal. 2021. Т. 12. № 4. С. 1–9. https://doi.org/10.17150/2411-6262.2021.12(4).11. EDN: HJXXGO.
11. Кнут Д.Э. Искусство программирования для ЭВМ: Основные алгоритмы. М.: Мир, 1976. 736 с.
12. Кнут Д.Э. Искусство программирования для ЭВМ: Сортировка и поиск. М.: Мир, 1978. 848 с.
13. Мартьянов В.И., Кулик Н.С., Пахомов Д.В., Большаков Э.А. Проект системы управления региональной сетью автомобильных дорог (СУРАД) Иркутской области // Вестник Иркутского государственного технического университета. 2014. № 4 (87). С. 118–183. EDN: SBNFNJ.
14. Кулик Н.С., Мартьянов В.И., Пахомов Д.В. Построение графа автомобильных дорог для системы взимания платы с большегрузного транспорта // Вестник Иркутского государственного технического университета. 2016. № 4 (111). С. 96–101. https://doi.org/10.21285/1814-3520-2016-4-96-101. EDN: VVSOFF.
15. Мартьянов В.И., Пахомов Д.В., Архипов В.В. Сетевое планирование содержания сети автомобильных дорог Иркутской области // Новые технологии в инвестиционно-строительной сфере и ЖКХ: сб. науч. трудов. (г. Иркутск, 14 апреля 2005 г.). Иркутск, 2005. Т. 1. С. 123–129.
16. Мартьянов В.И. Теоретико-множественные модели данных в задаче расчета вторичных структур РНК // System Analysis and Mathematical Modeling. 2022. Т. 4. № 4. С. 343–357. https://doi.org/10.17150/2713-1734.2022.4(4).343-357. EDN: VUNLFX.
17. Мартьянов В.И., Корольков Ю.Д. Теоретико-множественные модели сложных систем и алгоритмы интеллектуальной поддержки. Иркутск: Изд. дом БГУ, 2020. 125 с.
18. Martyanov V.I., Katashevtsev M.D. Computer Processing of Distorted Video Sequences Obtained by Mobile Road Laboratories // IOP Conference Series: Materials Science and Engineering. 2019. Vol. 667. Iss. 1. P. 1–9. http://doi.org/10.1088/1757-899X/667/1/012060.
19. Мартьянов В.И. Теоретико-множественный анализ организации данных учебного процесса и алгоритмы проектирования расписания с элементами искусственного интеллекта // Известия Байкальского государственного университета. 2020. Т. 30. № 4. С. 575–585. https://doi.org/10.17150/2500-2759.2020.30(4).575-585. EDN: QKOCYX.
20. Щербина О.А. Удовлетворение ограничений и программирование в ограничениях // Интеллектуальные системы. 2011. Т. 15. № 1-4. С. 53–170. EDN: PWUSTF.
21. Hentenryck van P. Constraint Satisfaction in Logic Programming. Cambridge: The MIT Press, 1989. 224 p.
22. Стюарт И. Величайшие математические задачи. М.: Альпина нон-фикшн, 2015. 584 с.
23. Разборов А.А. Алгебраическая сложность. М.: Московский центр непрерывного математического образования, 2016. 32 с.
24. Fortnow L. The Golden Ticket: P, NP, and the Search for the Impossible. Princeton: Princeton University Press. 2017. 176 p.
25. Leeuwen van J. Handbook of Theoretical Computer Science. Cambridge: The MIT Press, 1990. 1014 p.
26. Knuth D.E. Postscript about NP-hard problems // ACM SIGACT News. 1974. Vol. 6. Iss. 2. P. 15–16. https://doi.org/10.1145/1008304.1008305.
27. Aaronson S. The Scientific Case for P≠NP // Shtetl-Optimized. 2014. Режим доступа: https://scottaaronson.blog/?p=1720 (дата обращения: 20.05.2024).
28. Aaronson S. Quantum Computing Since Democritus Lecture 6: P, NP, and Friends // Shtetl-Optimized. 2006. Режим доступа: https://scottaaronson.blog/?p=149 (дата обращения: 20.05.2024).
29. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. New York: John Wiley & Sons, 1985. 476 p.
30. Wegener I. Complexity Theory: Exploring the Limits of Efficient Algorithms. New York: Springer, 2005. 320 p.

REFERENCES

1. Mart'yanov V.I. Draft Program Complex for Generating Turing Machines Solving Np-Hard Problems. *Proceedings of Universities. Investment. Construction. Real estate.* 2023;13(2):285-297. (In Russ.). https://doi.org/10.21285/2227-2917-2023-2-285-297. EDN: UAEWDT.

2. Mart'yanov V.I. Logical-Heuristic Methods of Network Planning and Situation Recognition. In: *Problemy upravleniya i modelirovaniya v slozhnykh sistemakh = Problems of Control and Modeling in Complex Systems.* Samara; 2001. p. 203–215. (In Russ.).
3. Mart'yanov V.I., Arkhipov V.V., Katashevtsev M.D., Pakhomov D.V. Logic-Heuristic Methods for Solving Combinatorial Problems of High Complexity Applications Preview. *Modern technologies. System analysis. Modeling.* 2010;4(28):205-211. (In Russ.). EDN: NRBKXP.
4. Martyanov V.I., Sukhorutchenko V.V., Okuntsov V.V. Planning of Information Flows in A Hierarchical System. In: *Prikladnye sistemy = Application Systems*. Moscow: Institute of Design Automation of the Russian Academy of Sciences; 1992. p. 46–58. (In Russ).
5. Lagutenkov A.V. Cryptocurrencies. Application Rules. *Nauka i zhizn'*. 2018;2:22-26. (In Russ.).
6. Kharpal A. All You Need to Know About the Top 5 Cryptocurrencies. *Yahoo! Finance*. 2017. Available from: https://finance.yahoo.com/news/know-top-5-cryptocurrencies-093231173.html [Accessed 20th May 2024].
7. Wilson-Nunn D., Zenil H. On the Complexity and Behaviour of Cryptocurrencies Compared to Other Markets. *Cornell University: arxiv.* 2014:1-16. https://doi.org/10.48550/arXiv.1411.1924.
8. Eremeev A.V., Zaozerskaya L.A., Kolokolov A.A. The Set Covering Problem: Complexity, Algorithms, and Experimental Investigations. *Discrete Analysis and Operations Research*. 2000;7(2):22-46. (In Russ). EDN: IBBFDL.
9. Mart'yanov V.I., Vyatkin I.V., Mogil'nitskii E.Yu. Theoretical and Implementation Aspects of Some Classes of Network Management Tasks. In: *Problemy upravleniya i modelirovaniya v slozhnykh sistemakh = Problems of Control and Modeling in Complex Systems.* Samara; 1999. p. 203–208. (In Russ.).
10. Mart'yanov V.I. NP-Difficult Tasks: Automatic Proof of Theorems and Turing's Machine. *Baikal Research Journal.* 2021;12(4):1-9. (In Russ.). https://doi.org/10.17150/2411-6262.2021.12(4).11. EDN: HJXXGO.
11. Knut D.E. *The Art of Computer Programming: Basic Algorithms.* Moscow: Mir, 1976. 736 p. (In Russ.).
12. Knut D.E. *The Art of Computer Programming: Sorting and Searching*. Moscow: Mir, 1978. 848 p. (In Russ.).
13. Mart'yanov V.I., Kulik N.S., Pakhomov D.V., Bol'shakov E.A. Project of The System to Control Irkutsk Regional Automobile Road Network. *Proceedings of Irkutsk State Technical University*. 2014;4(87):118-123. (In Russ.). EDN: SBNFNJ.
14. Kulik N.S., Mart'yanov V.I., Pakhomov D.V. Building A Motorway Graph for The Heavy Trucks Tolling System. *Proceedings of Irkutsk State Technical University*. 2016;4(111):96-101. (In Russ.). EDN: VVSOFF
15. Mart'yanov V.I., Pakhomov D.V., Arkhipov V.V. Network Planning of the Maintenance of the Highway Network of the Irkutsk Region. In: *Novye tekhnologii v investitsionno-stroitel'noi sfere i ZhKKh: sbornik nauchnykh trudov = New Technologies in The Investment and Construction Sector and Housing and Communal Services: A Collection of Scientific Papers.* 14 April 2005, Irkutsk. Irkutsk; 2005. vol. 1, p. 123–129. (In Russ.).
16. Mart'yanov V.I. Set-Theoretical Data Models in The Problem of Calculating Secondary Structures of RNA. *System Analysis and Mathematical Modeling.* 2022;4(4):343-357. (In Russ.). https://doi.org/10.17150/2713-1734.2022.4(4).343-357. EDN: VUNLFX.
17. Mart'yanov V.I., Korol'kov Yu.D. *Set-Theoretic Models of Complex Systems and Intellectual Support Algorithms.* Irkutsk: Publishing House of Baikal State University. 2020. 125 p. (In Russ.).
18. Mart'yanov V.I., Katashevtsev M.D. Computer Processing of Distorted Video Sequences Obtained by Mobile Road Laboratories. *IOP Conference Series: Materials Science and Engineering.* 2019;667(1):1-9. http://doi.org/10.1088/1757-899X/667/1/012060.
19. Mart'yanov V.I. Set-Theory Analysis of the Data Organization of the Educational Process and Algorithms for Designing a Schedule with Elements Artificial Intelligence. *Bulletin of Baikal State University*. 2020;30(4):575-585. (In Russ.). https://doi.org/10.17150/2500-2759.2020.30(4).575-585. EDN: QKOCYX.
20. Shcherbina O.A. Constraint Satisfaction and Constraint Programming. *Intelligent Systems.Theory and Applications.* 2011;15(1-4):53-170. (In Russ.). EDN: PWUSTF.
21. Hentenryck van P. *Constraint Satisfaction in Logic Programming.* Cambridge: The MIT Press, 1989. 224 p.
22. Styuart I. *The Greatest Math Problems*. Moscow: Alpina non-fiction, 2015. 584 p. (In Russ.).
23. Razborov A.A. *Algebraic Complexity*. Moscow: Moscow Center for Continuing Mathematical Education, 2016. 32 p.
24. Fortnow L. *The Golden Ticket: P, NP, and the Search for the Impossible.* Princeton: Princeton University Press. 2017. 176 p.
25. Leeuwen van J. *Handbook of Theoretical Computer Science.* Cambridge: The MIT Press, 1990. 1014 p.
26. *Knuth D.E. Postscript about NP-hard problems. ACM SIGACT News. 1974;6(2):15-16.* https://doi.org/10.1145/1008304.1008305.
27. Aaronson S. The Scientific Case for P≠NP. *Shtetl-Optimized*. 2014. Available from: https://scottaaronson.blog/?p=1720 [Accessed 20th May 2024].
28. Aaronson S. Quantum Computing Since Democritus Lecture 6: P, NP, and Friends. *Shtetl-Optimized.* 2006. Available from: https://scottaaronson.blog/?p=149 [Accessed 20th May 2024].
29. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* New York: John Wiley & Sons, 1985. 476 p.

30. Wegener I. *Complexity Theory: Exploring the Limits of Efficient Algorithms*. New York: Springer, 2005. 320 p.

Информация об авторах

Мартьянов Владимир Иванович,
д.ф-м.н., профессор кафедры
автомобильных дорог,
Иркутский национальный исследовательский
технический университет,
664074, г. Иркутск, ул. Лермонтова, 83, Россия,
Байкальский государственный университет,
664003, г. Иркутск, ул. Ленина, 11, Россия,
Иркутский государственный университет,
664003, г. Иркутск, ул. Карла Маркса, 1, Россия,
✉e-mail: martvliv@mail.ru
https://orcid.org/0000-0003-2659-0355
Author ID: 2477

Волков Андрей Сергеевич,
аспирант,
Байкальский государственный университет,
664003, г. Иркутск, ул. Ленина, 11, Россия,
e-mail: 0077707@mail.ru

Information about the authors

Vladimir I. Martyanov,
Dr. Sci. (Phys.-Math.), Professor
of the Department of Highways,
Irkutsk National Research
Technical University,
83 Lermontov St., Irkutsk 664074, Russia,
Baikal State University,
11 Lenin St., Irkutsk 664003, Russia,
Irkutsk State University,
1 Karl Marx St., Irkutsk 664003, Russia,
✉e-mail: martvliv@mail.ru
https://orcid.org/0000-0003-2659-0355
Author ID: 2477

Andrey S. Volkov,
Postgraduate Student,
Baikal State University,
11 Lenin St., Irkutsk 66403, Russia,
e-mail: 0077707@mail.ru

Вклад авторов

Все авторы сделали эквивалентный вклад в подготовку публикации.

Contribution of the authors

The authors contributed equally to this article.

Конфликт интересов

Авторы заявляют об отсутствии конфликта интересов.

Все авторы прочитали и одобрили окончательный вариант рукописи.

Conflict of interests

The authors declare no conflict of interests regarding the publication of this article.

The final manuscript has been read and approved by all the co-authors.

Информация о статье

Статья поступила в редакцию 31.05.2024.
Одобрена после рецензирования 14.06.2024.
Принята к публикации 17.06.2024.

Information about the article

The article was submitted 31.05.2024.
Approved after reviewing 14.06.2024.
Accepted for publication 17.06.2024.