МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Системное программирование и компьютерные технологии
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Реализация алгоритмов метода Гомори для задачи целочисленного линейного программирования»

Студент	Д.А. Бердников	
	(И.О. Фамилия)	(личная подпись)
Руководитель	Г.А. Тырыгина	
	(И.О. Фамилия)	(личная подпись)
Консультант	Н.В. Ященко	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _______________
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____» _________________ 2018 г.

Тольятти 2018

АННОТАЦИЯ

Тема: Реализация алгоритмов метода Гомори для задачи целочисленного линейного программирования.

Ключевые слова: ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ, АЛГОРИМ ГОМОРИ, АНАЛИЗ, РЕАЛИЗАЦИЯ.

Дипломная работа состоит из введения, трех глав, заключения, списка использованной литературы и приложения. Объект исследования бакалаврской работы – целочисленное линейное программирование. Предмет исследования бакалаврской работы – алгоритм Гомори. Цель бакалаврской работы - анализ и реализация алгоритма Гомори.

Дипломную работу можно разделить на следующие логически связанные части: Теоретические основы метода Гомори; Первый алгоритм Гомори; Второй алгоритм Гомори.

Методы исследования: методы решения задач целочисленного программирования, математический аппарат теории алгоритмов.

Проанализированы проблемы целочисленного линейного программирования. Дано описание метода отсекающих плоскостей и алгоритма Гомори, позволяющего решить задачу целочисленного программирования с высокой точностью.

Описаны свойства и произведен анализ алгоритма Гомори.

Выполнена реализация алгоритма на основе современных Web-технологий и проведено тестирование разработанного решения.

Структура бакалаврской работы: страниц 54 с приложением, рисунков 8, таблиц 21, источников 25.

ABSTRACT

The title of the graduation work is The Implementation of Gomory Method Algorithms for the Solving the Integer Linear Programming. The graduation work consists of an introduction, three parts, conclusions, the list of references and an appendix. The object of the graduation work is the integer linear programming. The subject of the graduation work is the algorithm Gomory. The aim of the work is to give some information about analysis and implementation of the algorithm Gomory.

The graduation work may be divided into several logically connected parts which are: Theoretical basis of the method Gomory; The first algorithm Gomory; The second algorithm Gomory.

The methods of research are methods for solving the integer programming problems, mathematical apparatus of the theory of the algorithms.

The problems of the integer linear programming are analyzed. A description is given of the method of cutting planes and the algorithm Gomory, which allows to solv the integer programming problem with high accuracy.

Properties and analysis of the algorithm Gomory are described.

The algorithm is implemented on the basis of modern Web technologies and the developed solution is tested.

The graduation work consists of the text on 54 pages, 8 figures, 21 tables, the list of 25 references including 5 foreign sources.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целочисленное программирование широко применяется для решения задач в таких областях человеческой деятельности, как производственное планирование, составление расписаний, сети передачи данных и сотовая связь и других, т.е. там, где приходится оперировать штучными физическими или логическими величинами (0 или 1).

Следует отметить, что в отличие от задач линейного программирования, задачи целочисленного программирования очень сложны для решения. Фактически, для их решения не известен эффективный общий алгоритм [18].

Вместе с тем, для некоторых задач известен ряд алгоритмов, позволяющих решить задачу целочисленного программирования с высокой точностью. Одним из таких алгоритмов, анализ и реализация которых представляют научно-практический интерес, является алгоритм Гомори. Этим определяется **актуальность** бакалаврской работы.

Объект исследования бакалаврской работы – целочисленное линейное программирование.

Предмет исследования бакалаврской работы – алгоритм Гомори.

Цель бакалаврской работы - анализ и реализация алгоритма Гомори.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать теоретические основы метода Гомори;
- осуществить реализацию первого алгоритма метода Гомори;
- осуществить реализацию второго алгоритма метода Гомори
- произвести сравнительный анализ алгоритмов метода Гомори;
- выбрать средство программной реализации алгоритмов Гомори;

Представленная бакалаврская работа состоит из введения, трех глав, заключения, приложения и списка литературы.

Во введение обозначается тема работы и ее актуальность, описывается объект и предмет исследования, цели и задачи, которые необходимо решить в данной работе.

В первой главе представлены теоретические основы.

Вторая глава посвящена первому алгоритму Гомори и его реализации.

В третьей главе рассматривается второй алгоритм Гомори и его реализация и произведено сравнение с первым алгоритмом Гомори.

В заключении представлены результаты выполнения бакалаврской работы.

В приложении приведен кода программной реализации алгоритма Гомори.

Структура бакалаврской работы: страниц 54 с приложением, рисунков 8, таблиц 21, источников 26.

Глава 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МЕТОДА ГОМОРИ

1.1 Методы линейного программирования

Линейное программирование (ЛП) – вид математического моделирования, в котором находится экстремум (максимум или минимум) линейной функции при различных ограничениях [20].

Этот метод полезен для определения количественных решений в области бизнес-планирования, в промышленном машиностроении и, в меньшей степени, в социальных и естественных науках.

Более формально ЛП рассматривается как метод оптимизации линейной целевой функции с учетом линейного равенства и ограничений линейного неравенства.

Допустимая область линейного программирования является выпуклым многогранником, который является множеством, определяемым как пересечение конечного числа полупространств, каждое из которых определяется линейным неравенством.

Целевая функция ЛП - это вещественная линейная функция, определенная на этом многограннике. С помощью алгоритма ЛП находится точка в многограннике, где эта функция имеет наименьшее (или наибольшее) значение, если такая точка существует [3].

Задачи линейного программирования - это задачи, которые могут быть представлены в канонической форме следующим образом:

найти максимум целевой функции $c^T x$ при условии $Ax \leq b$ и $x \geq 0$,

где:

x - вектор переменных (определяется);

c и b - векторами известных коэффициентов;

A - известная матрица коэффициентов;

c^T - транспонированная матрица.

Неравенства $Ax \leq b$ и $x \geq 0$ являются ограничениями, определяющими выпуклый многогранник, над которым должна быть оптимизирована целевая функция $c^T x$. В данном контексте два вектора сравнимы, если они имеют одинаковые размеры. Если каждая запись в первом векторе меньше или равна соответствующей записи во втором векторе, то можно утверждать, что первый вектор меньше или равен второму вектору.

Задачи линейного программирования решаются с помощью симплекс-метода (simplex method) (рисунок 1.1).

«Симплекс-метод (с лат. – простой) или метод последовательного улучшения решений зада линейного программирования заключается в определении координаты только тех угловых точек, в которых значение целевой функции лучше предыдущего (больше для задачи максимизации и меньше для задачи минимизации)» [7].

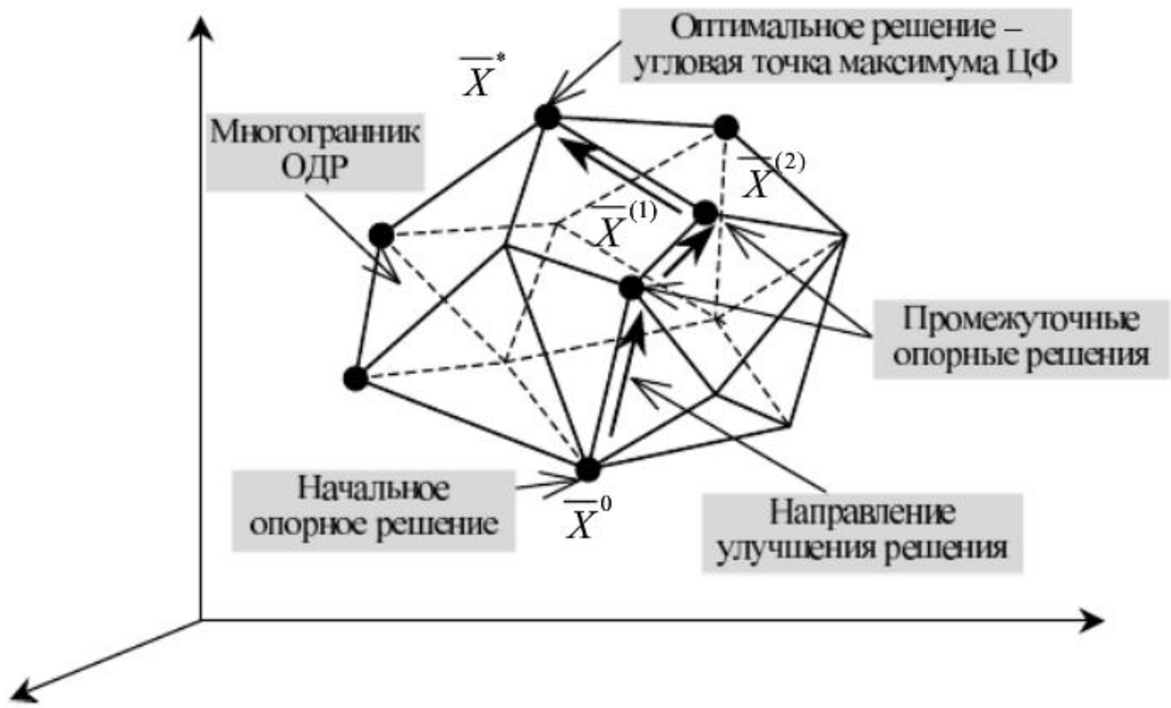

Рисунок 1.1- Графическое представление симплекс-метода

При реализации симплекс-метода на каждой итерации в новый базис вводится переменная с номером t, который представляет собой номер столбца матрицы ограничений канонической задачи линейного программирования. При этом из базиса выводится переменная с номером s, представляющим номер строки данной таблицы. Таким образом, вычислительную процедуру симплекс-метода можно реализовать с помощью таблицы, которая называется симлекс-таблицей (simplex tableau), чтобы на каждой итерации процедуры использовать определенные строку и столбец указанной таблицы (рисунок 1.2).

		c_1	c_2	…	c_j	…	c_n	c_{n+}	…	c_{n+m}		
Базис	$C^Б$	x_1	x_2	…	x_j	…	x_n	x_{n+}	…	x_{n+m}	A_0	α
x	$c_1^Б$	a	a_2	…	a_{1j}	…	a_n	$a_{1,n+1}$	…	$a_{1,n+m}$	a_0	
x_2	$c_2^Б$	a_2	a_{22}	…	a_{2j}	…	a_{2n}	$a_{2,n+1}$	…	$a_{2,n+m}$	a_2	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	
$x_{i_{вп}}$	$c_j^Б$	$a_{i_{вп}1}$	$a_{i_{вп}2}$	…	$a_{i_{вп}j}$	…	$a_{i_{вп}n}$	$a_{i_{вп},n+}$	…	$a_{i_{вп},n+m}$	$a_{i_{вп}0}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	
x_m	$c_m^Б$	a_{m1}	a_{m2}	…	a_{mj}	…	a_{mn}	$a_{m,n+1}$	…	$a_{m,n+m}$	a_{m0}	
$\Delta_j = z_j - c_j$		$Z_1 - c_1$	$Z_2 - c_2$	…	$Z_j - c_j$	…	$Z_n - c_n$	$Z_{n+1} - c_{n+1}$	…	$Z_{n+m} - c_{n+m}$	$Z(X^Б)$	

Рисунок 1.2 - Структура симплекс-таблицы

В двойственным симплекс-методе (dual simplex method) фактически не используются упрощения, но одна его интерпретация заключается в том, что он работает на симплициальных конусах, и они становятся подходящими симплексами с дополнительным ограничением.

Симплициальными конусами являются углы (окрестности вершин) геометрического объекта, называемого многогранником. Форма этого многогранника определяется ограничениями, применяемыми к целевой функции.

1.2 Целочисленное программирование

Целочисленное программирование (ЦЛП)- это математическая программа оптимизации или выполнимости, в которой некоторые или все переменные представлены только целыми числами. Данный термин очень часто относят к целочисленному линейному программированию, в котором целевая функция и ограничения (кроме ограничений целостности) являются линейными.

Иными словами, целочисленное программирование - это подмножество линейного программирования.

Целочисленному программированию присущи все особенности линейного программирования, за исключением одной: решение задачи должно быть ограничено целыми числами или булевыми значения.

Наиболее известными задачами целочисленного программирования являются: задача о размещение (задача о рюкзаке), задача о назначениях и задача о коммивояжере [6].

Проблема получения наилучшего целочисленного решения для линейной программы в различных контекстах поднималась в 50-х годах 20 века Данцигом, Марковицем, Манном и Дрейфусом.

Решить ее удалось в 1958г. американскому ученому Ральфу Е. Гомори.

1.2.1 Постановка задачи целочисленного программирования

Рассмотрим каноническую задачу ЦЛП.

Необходимо максимизировать функцию:

$$F(x) = \sum_{j=1}^{n} c_j x_j \qquad (1.1)$$

при условиях:

$$\sum_{j=1}^{n} a_{ij} x_j = b_j,\ i = \overline{1,m}; \qquad (1.2)$$

$$x_j \geq 0,\ j = \overline{1,n}; \qquad (1.3)$$

$$x_j - \text{целые}, j = \overline{1, n_1}; n_1 \le n \; (1.4)$$

Если $n_1=n$, то задача (1.1)-(1.4) - полностью целочисленная задача ЛП.

Если $n_1<n$, то задача (1.1)-(1.4) - частично целочисленная задача ЛП.

1.2.2 Условие целочисленности многогранных множеств

Для задачи ЦЛП введем ряд понятий, аналогичных задаче ЛП.

Множество векторов $X = (x_1, x_2, \ldots, x_n)$, удовлетворяющих условиям (1.2) – (1.4), называется областью определения задачи ЦЛП (1.1) – (1.4).

Вектор X, удовлетворяющий условиям (1.2) – (1.4), представляет собой план (допустимое решение) задачи ЦЛП (1.1) – (1.4).

Крайняя точка выпуклого многогранника множества D называется опорным планом.

План $X = (x_1^*, x_2^*, \ldots, x_n^*)$, который обращает в максимум линейную проблемы (1.1), называется оптимальным планом (решением) задачи ЦЛП (1.1) – (1.4).

Если $X = (x_1, x_2, \ldots, x_n)$ – оптимальный план и $x_0 = \sum_{j=1}^{n} c_j x_j$, то вектор $\tilde{X} = (x_0, x_1, \ldots, x_n)$ называется расширенным оптимальным планом задачи ЦЛП (1.1) – (1.4).

Введем следующие обозначения:

D – область определения задачи ЦЛП (1.2)-(1.3);

$D^ч$ – область определения задачи ЦЛП (1.2)-(1.4);

(D, F) – задача ЦЛП (1.1) - (1.3);

$(D^ч, F)$ – задача ЦЛП (1.1) - (1.4);

$X(D, F)$- оптимальный план задачи ЦЛП (1.1)-(1.3)

$X(D^ч, F)$- оптимальный план задачи ЦЛП (1.1)-(1.4)

$\tilde{X}(D^ч, F)$-расширенный оптимальный план задачи ЦЛП (1.1)-(1.4)

Задача ЦЛП является разрешимой, если оптимальный план X* существует.

Если все опорные планы (вершины) многогранника D целочисленные (все компоненты каждого из опорных планов - целые), то такой многогранник называется целочисленным многогранником.

Решить задачу ЦЛП ($D^ц$, F) - означает указать оптимальный план данной задачи или убедиться в том, что такой план не существует ($D^ц = \varnothing$, если F не ограничена сверху на $D^ц$).

Решая задачу ЦЛП ($D^ц$, F) необходимо определить возможность использования уже известных вычислительных процедур.

Рассмотрим следующие теоремы:

Теорема 1.1. Если целочисленны множества допустимых планов задачи ЦЛП (D, F), то справедливы следующие утверждения:

1) Каждому допустимому базису B задачи ЦЛП (D, F) соответствует целочисленный базисный план X(B).

2) Если **X***(B) является базисным решением задачи ЦЛП (D, F), то **X***(B) также является оптимальным планом задачи ЦЛП ($D^ц$, F).

3) Оптимальный план задачи ЦЛП (D, F) также является решением задачи ЦЛП ($D^ц$, F).

Доказательство:

1) Базисный план X(B) есть вершина целочисленного многогранника выпуклого множества D, и, следовательно, X(B) является целочисленным вектором.

2) Утверждение следует из включения $D^ц \subset D$.

3. Поскольку оптимальный план задачи ЦЛП (D, F) является базисным, то **X***(B) является решением задачи ЦЛП (D, F) и также является решением задачи ЦЛП ($D^ц$, F).

Замечание. Если $D = \varnothing$, то и $D^ц = \varnothing$. Однако задача ЦЛП ($D^ц$, F) может иметь решение и в случае, когда задача ЦЛП (D, F) не имеет решения.

Рассмотрим многогранное множество следующего вида: D = $\{Ax=b, x\geq 0\}$.

Задача: Необходимо определить, каким условиям должны удовлетворять матрица **A**=$(a_{ij},\ i=\overline{1,m},\ j=\overline{1,n})$ и вектор правых частей $b=(b_1,b_2,...,b_m)^T$, чтобы многогранное множество D (1.2)-(1.3) было численным?

Матрица **A** называется унимодулярной**,** если каждый ее минор равен +1, –1 или 0.

Решение дает теорема:

Теорема 1.2. Если матрица **A** (m x n) - унимодулярна, вектор $b\in R^m$ целочисленный и $D\neq\varnothing$, то многогранное выпуклое множество D -целочисленно.

Доказательство:

Известно, что $x_0\in D$ является экстремальной точкой множества D в том и только в том случае, если система $B_0=\{a^j,\ j\in[1..n]\setminus J(x_0)\}$ $(J(x_0)=\{j\in[1..n]\setminus X_0(j)=0\})$ столбцов матрицы **A**, индексы которых отвечают положительным компонентам вектора X_0, линейно независимы.

Введем следующие обозначения:

k - число векторов в множестве $\mathbf{B}_0$;

$\mathbf{B}_0$ – квадратная подматрица матрицы **A** k–го порядка, составленная из элементов множества B_0 и таких k-строк матрицы **A**, что det $\mathbf{B}_0\neq 0$;

b_0– вектор, включающий k соответствующих компонентов вектора b.

Тогда вектор z_0, составленный из k положительных компонентов вектора X_0, является решением системы линейных уравнений $\mathbf{B}_0 z=b_0$.

Ввиду того, что det $\mathbf{B}_0\neq 0$, в силу унимодулярности матрицы **A** он равен ± 1. Следовательно, $z_0=\mathbf{B}_0^{-1}b$ целочисленный вектор, то целочислен вектор X_0.

Следствие. Если в канонической задаче ЦЛП (1.1) – (1.4) матрица **A** унимодулярна и вектор b целочислен, то оптимальный план задачи ЦЛП (D, F), полученный любым методом, является решением задачи ЦЛП ($D^ц$, F).

Теорема 1.2 дает достаточные условия целочисленности для некоторых классов многогранных множеств. Но эти условия очень трудно проверить.

Вместе с тем, если множество D не является целочисленным, то оптимальный план X^* задачи ЦЛП (D, F) может также не быть целочисленным. Во многих случаях нельзя ограничиться и приближенным решением $[\mathbf{X}^*]$, в виду невозможности гарантировать допустимость вектора $[\mathbf{X}^*]$.

1.3 Метод отсекающих плоскостей Гомори

В основу алгоритмов Гомори положен предложенный им метод отсекающих плоскостей или метод отсечений (Cutting-plane method) [22].

В математической оптимизации метод отсекающих плоскостей представляет собой любой из множества методов оптимизации, которые итеративно уточняют допустимую совокупность или целевую функцию с помощью линейных неравенств, называемых отсечениями [24].

Такие процедуры обычно используются для нахождения целочисленных решений задач смешанного ЦЛП, а также для решения общих, не обязательно дифференцируемых выпуклых задач оптимизации.

Методы отсекающих плоскостей для задач смешанного ЦЛП программирования работают, решая нецелую линейную программу, линейную релаксацию данной целочисленной программы.

Согласно теории линейного программирования, при слабых предположениях (если линейная программа имеет оптимальное решение и если допустимая область не содержит прямой), всегда можно найти экстремальную точку или оптимальную точку угла.

Полученный оптимум тестируется как целочисленное решение.

Если это не так, то существует линейное неравенство, отделяющее оптимум от выпуклой оболочки истинного допустимого множества. Поиск такого неравенства является проблемой разделения, и такое неравенство является отсечением. В расслабленную линейную программу можно добавить отсечение. Тогда текущее нецелое решение перестает быть возможным для релаксации. Этот процесс повторяется до тех пор, пока не будет найдено оптимальное целочисленное решение.

Дадим математическое описание метода отсекающих плоскостей Гомори на примере следующей задачи ЦЛП [4].

$$f(x) = (c, x) \to \max, \tag{1.5}$$

$$Ax = b \tag{1.6}$$

$$x \geq 0 \tag{1.7}$$

$$x \equiv 0 \ (\mathrm{mod}\ 1), \tag{1.8}$$

где:

c, b - целочисленные векторы;

A - целочисленная матрица.

Допустимое множество задачи (1.5-1.8), являющееся пересечением выпуклого многогранного множества X и целочисленной решетки Z_n , обозначим через X_Z.

Пусть x^*_R, x^*_Z оптимальное непрерывное и оптимальное целочисленное решения задачи соответственно.

Предположим, что задача разрешима и $x^*_R \neq x^*_Z$.

Представим систему (1.6) в виде:

$$x_B = B^{-1}b - B^{-1}N\, x_N, \tag{1.9}$$

где:

B - оптимальный базис задачи ЦЛП (1.5) - (1.7);

x_B, x_N - векторы базисных и небазисных переменных, причем $N = A \setminus B$.

Введем обозначения:

$b = B^{-1}b$; $A = B^{-1}N$; J_B, J_N - множества номеров базисных и небазисных переменных.

Опишем систему (1.9) в координатной форме:

$$x_i = \bar{b}_i - \sum_{j \in J_N} \bar{a}_{ij} x_j, \quad i \in J_B \tag{1.10}$$

Чтобы все x_i, $i \in J_B$ были целыми числами необходимо выполнение следующего условия:

$$\bar{b}_i - \sum_{j \in J_N} \bar{a}_{ij} x_j \equiv 0 \,(\mathrm{mod}\, 1), \quad i \in J_B,$$

что эквивалентно условию:

$$\{\bar{b}_i\} - \sum_{j \in J_N} \{\bar{a}_{ij}\} x_j \equiv 0 \,(\mathrm{mod}\, 1), \quad i \in J_B, \quad (1.11)$$

где {.} – дробная часть числа.

Выражение (1.11) является необходимым и достаточным условием целочисленности вектора x_B, которому трудно удовлетворить.

Для упрощения представим это выражение в виде неравенства, определяющего необходимое условие:

$$-\sum_{j \in J_N} \{\bar{a}_{ij}\} x_j \le -\{\bar{b}_i\}, \quad i \in J_B \qquad (1.12)$$

Если $\{\bar{b}_i\} > 0$, то оптимальное непрерывное решение $x^*_N = 0$, $x^*_B = b$ не удовлетворяет неравенству (1.12) и, следовательно, данное неравенство является правильным отсечением Гомори.

На рисунке 1.3 представлен пример правильного отсечения [16].

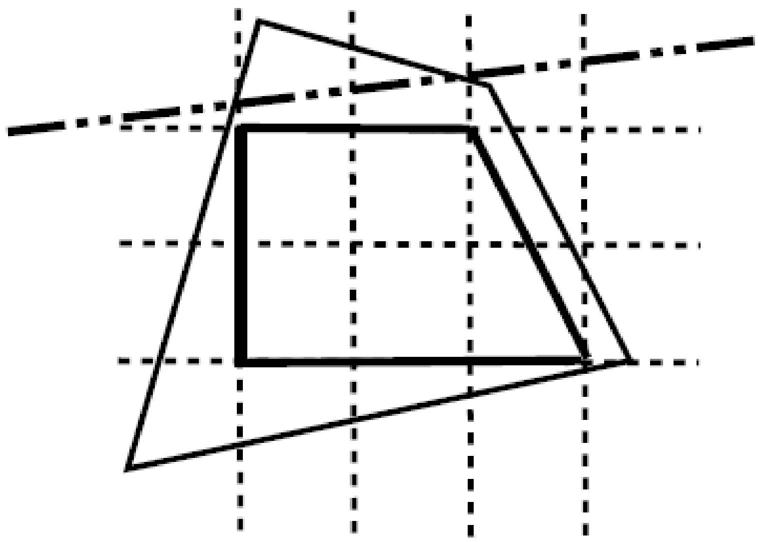

Рисунок 1.3 – Пример правильной отсекающей плоскости Гомори

Тонкие внешние линии изображают исходные ограничения.

Толстые внутренние линиями выделены сильные правильные отсечения. Они определяют минимальную область ЛП, которая включает в себя все возможные целочисленные решения.

Следует отметить, что сильные правильные отсечения всегда существуют. Если мы найдем их, то получим оптимальное целочисленное решение.

Проблема в том, что поиск сильных правильных отсечений довольно сложен.

Пунктирными линиями выделены правильные, но не сильные отсечения.

1.4 Эффективность отсечения метода Гомори

Эффективность отсечения определяется размерами области, котоая отсекается от многогранника допустимых решений [15].

Математически это выражается следующим образом.

Пусть даны следующие неравенства:

$$\sum_{j=1}^{n} a_{ij} x_j \geq a_{i0} \qquad (1.13)$$

$$\sum_{j=1}^{n} a_{kj} x_j \geq a_{k0} \qquad (1.14)$$

Предположим, что отсечение (1.13) более эффективно, чем отсечение (1.14), если выполняются условия $a_{i0} > a_{k0}$ и $a_{ij} \leq a_{kj}$ для всех j, причем хотя бы при одном j выполняется строгое неравенство.

Использование данного определения эффективности отсечения связано с существенными трудностями вычислительного характера.

В этой связи разработаны эмпирические правила, которые отражают смысл этого определения. Два таких правила предписывают строить отсечение на основе производящей строки, которой соответствует $\max_i \{a_{i0}\}$ или $\max_i (a_{i0} / \sum_{j=1}^{n} a_{ij})$.

Второе правило обладает определенными преимуществами перед первым, ввиду того, что оно теснее связано с представленным выше определением эффективности отсечения.

Основные недостатки метода отсечения:

1) ни один тип отсечений не обеспечивает высокой эффективности соответствующих вычислительных процедур;

2) метод отсечений не подходит для решения целочисленных задач больших размерностей.

Основной вывод о возможности использования метода отсечения Гомори заключается в констатации отсутствия эффективного решения общей задачи ЦЛП с помощью данного метода.

Глава 2 ПЕРВЫЙ АЛГОРИТМ ГОМОРИ

2.1 Описание первого алгоритма Гомори

Алгоритм Гомори, который в литературе называется первым алгоритмом Гомори или циклическим алгоритмом отсечения, состоит из следующих шагов [23].

Применяется только для решения полностью целочисленных задач ЛП.

Шаг 1. Инициализация.

Пусть значение наибольшего целого $\leq a$ (округленное значение).

Определяем дробную часть числа a как a - $[a]$.

Формулируем стандартную задачу ЦЛП. Если в уравнениях ограничений есть нецелые коэффициенты, преобразуем их в целые коэффициенты. Решаем его симплекс-методом, игнорируя целочисленное требование переменных.

Шаг 2. Проверка оптимальности.

а) Исследуем оптимальное решение. Если все основные переменные имеют целочисленные значения, то получается целочисленное оптимальное решение и процедура должна быть прекращена. Текущее оптимальное решение, полученное на шаге 1, является оптимальным базовым решением линейного ЦЛП.

б) Если одна или несколько базисных переменных имеют нецелые значения решения, перейдите к шагу 3.

Шаг 3. Генерация отсекающей плоскости.

Выбираем для нецелой переменной b_i^* дополнительное ограничение вида:

$$\sum_j a_{ij}^* x_j = b_i^*$$

Перепишем данное ограничение, используя дробные части $f_{ij} = a_{ij} - [a_{ij}]$, $f_i = b_i - [b_i]$:

$$\sum_j f_{ij}^* x_j - f_i^* = [b_i^*] - \sum_j [a_{ij}^*] x_j$$

Эмпирическим путем выбираем для шага 2 b_i^* с наибольшим значением f_i^*.

Шаг 4. Получаем новое решение.

Добавляем отсекающую плоскость, сгенерированную на шаге 3, в конец оптимальной симплекс-таблицы, полученной на шаге 3.

$$\sum_j f_{ij} x_j - f_i \ \geq 0$$

Находим новое оптимальное решение, используя двойной симплекс-метод.

Возвращаемся на шаг 3 и повторяем процесс, используя двойственный симплекс-метод, до тех пор, пока все значения b_i^* не станут неотрицательными.
и целочисленными.

На алгоритмическом языке данный метод будет иметь вид [14]:

Procedure

Begin

1. Решить непрерывную задачу ЛП с исходными ограничениями

While

Begin

2.1 Сформировать новое ограничение

2.2 Решить непрерывную задачу ЛП с новыми ограничениями

End

End.

Первый алгоритм Гомори обладает следующими свойствами [17]:

1) Дополнительные линейные ограничения никогда не отсекают ту часть исходного пространства допустимого решения, которая содержит возможное целочисленное решение исходной задачи.

2) Каждое новое дополнительное ограничение (или гиперплоскость) отсекает текущее нецелое оптимальное решение задачи линейного программирования.

3) Доказано, что первый алгоритм Гомори имеет конечное число итераций [2].

2.2 Численная реализация первого алгоритма Гомори

Блок-схема алгоритма Гомори для рассмотренного выше случая представлена на рисунке 2.1 [1].

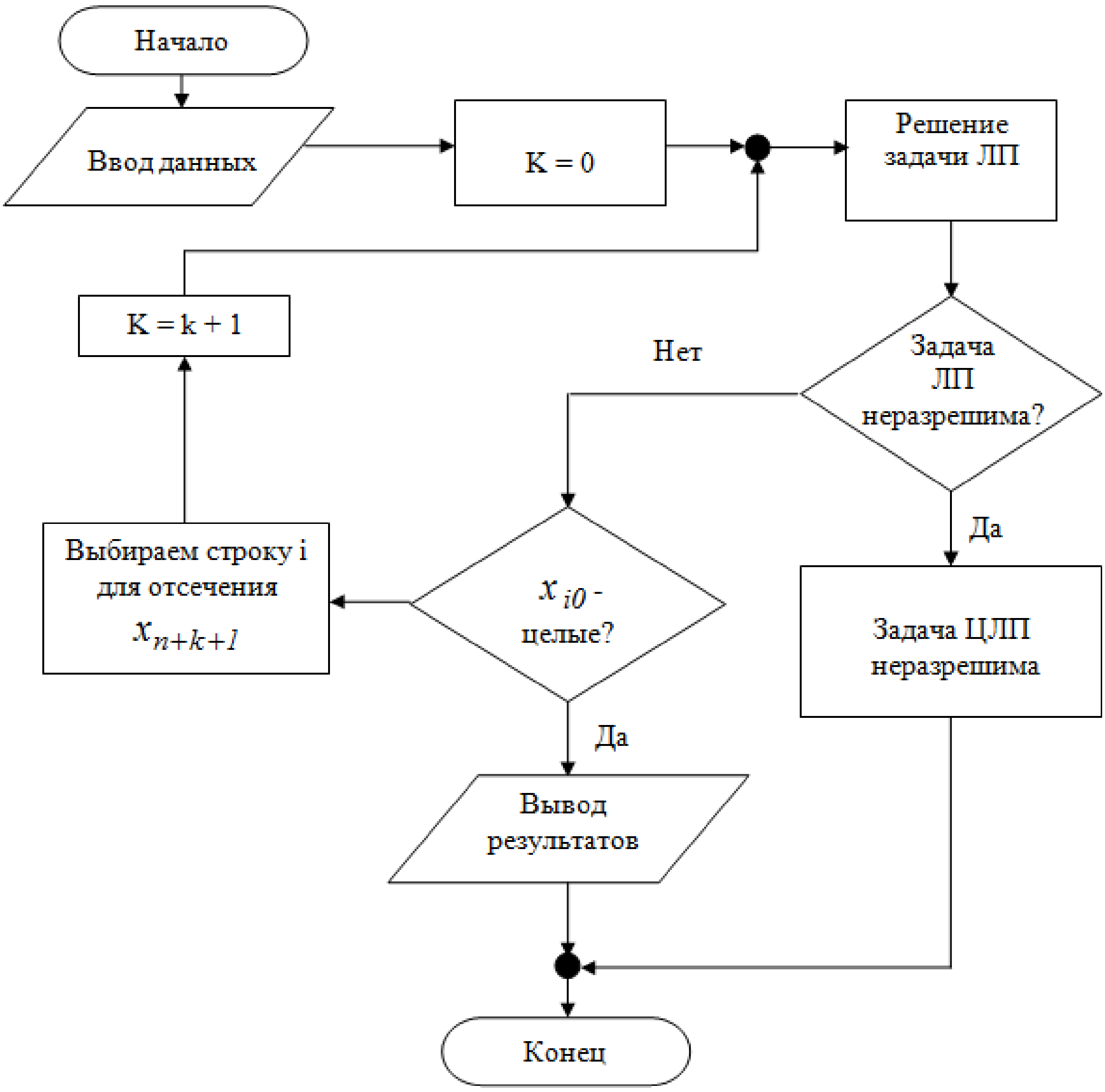

Рисунок 2.1 - Блок-схема первого алгоритма Гомори

Гомори показал, что при *K*-м возвращении к решению задачи линейного программирования, *K*-е дополнительное ограничение имеет вид [4]:

$$x_{n+k+1} = -(x_{i0} - [x_{i0}]) + \sum_{j \in I} (x_{ij} - [x_{ij}]) x_j, \quad k = 0, 1, 2, ..., x_{n+k+1} \geq 0,$$

где :

$[x_{i0}],[x_{ij}]$ - – целая часть соответствующей величины;

x_{i0} - нецелая координата оптимального плана задачи (1.9-1.10), у которой нецелая часть самая большая;

x_j - координаты разложения векторов A_j, не попавших в базис;

I_s – множество векторов, не попавших в базис.

При разработке компьютерной программы, реализующей первый алгоритм Гомори, можно использовать следующими рекомендациями [19]:

1) Для каждой итерации рассматриваем сходящуюся симплекс-таблицу на шаге 3 при *m* x *n* *A** (таблица 2.1).

Таблица 2.1 – Сходящаяся симплекс-таблица (**X** – вектор-матрица, *z* - решение**)**

	$\mathbf{x}^t$	
$\mathbf{X}^*_в$	A^*	b*
	c^{*t}	z^*_0

Для любой нецелой базовой переменной $x_{B(i)}$ имеем:

$$x_{B(i)} = b_i^* = \sum\nolimits_j a_{i,j}^* x_j \geq \sum\nolimits_j [a_{i,j}^*] x_j \geq \sum\nolimits_j [a_{i,j}^*][x_j]$$

2) Итак, для:

− целочисленной **X**:

– $\sum\nolimits_j [a_{i,j}^*] x_j \leq [x_{B(i)}]$ и слабой $x_{n+1} \geq 0$ имеем:

$$[b_i^*] = \sum\nolimits_j [a_{i,j}^*] x_j + x_{n+1}$$

Целочисленные решения исходной задачи должны удовлетворять новым ограничениям.

3) Вычитаем, чтобы получить дополнительное ограничение для таблицы:

$$f_i^* = \sum_j f_{i,j}^* x_j - x_{n+1} \quad \text{или} \quad -\sum_j f_{i,j}^* x_j + x_{n+1} = -f_i^*$$

4) Новое ограничение (плоскость) отсекает часть исходной допустимой области, но пересмотренная симплекс-таблица имеет такие же целочисленные решения, что и исходная задача.

Можно отметить следующие проблемы компьютерной реализации первого алгоритма Гомори:

1) целочисленное решение получается только в конце, что может существенно увеличить время решения задачи. Поэтому необходимо использовать компьютеры с мощными вычислительными ресурсами;

2) ошибки округления могут привести к нахождению неоптимального целочисленного решения.

Таким образом, первый алгоритм Гомори имеет некоторые ограничения для практического применения.

Глава 3 ВТОРОЙ АЛГОРИТМ ГОМОРИ

3.1 Описание второго алгоритма Гомори

Второй алгоритм Гомори предназначен для решения более широкого класса задач.

С его помощью в отличие от первого алгоритма Гомори решается частично целочисленная задача ЛП вида:

$$\begin{cases} F(x) = \sum_{j=1} c_j x_j \to \max & (3.1) \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,...,m, & (3.2) \\ x_j \geq 0, \qquad j = 1,...,n, & (3.3) \\ x_j - \text{целое}, \quad j = 1,...,n_1,\ n_1 \leq n & (3.4) \end{cases}$$

Второй алгоритм Гомори формулируется в виде следующей теоремы:

Теорема 3.1. Пусть

$X(D_k, F) = X^k -$ оптимальное решение задачи (D_k, F) и

$T_r = \left\| a_{ij}^k \right\|$ - соответствующая симлексная таблица

При a^k_{i0} ($1 \leq i \leq n_1$ или $0 \leq i \leq n_1$) - нецелом, следующее неравенство:

$$\sum_{j \in N_k} \gamma_j x_j \geq \gamma_0 \qquad (3.5)$$

или

$$z = -\gamma_0 + \sum_{j \in N_k} \gamma_j x_j, \qquad (3.6)$$

$z -$ целое ≥ 0 , где $\gamma_0 = \{a_{i0}^k\}$

$$\gamma_i = \begin{cases} \{a_{ij}^k\}, \quad j \leq n_1, \{a_{ij}^k\} \leq \{a_{i0}^k\}, \\ \dfrac{\{a_{i0}^k\}}{1-\{a_{i0}^k\}}(1-\{a_{ij}^k\}), \quad j \leq n_1, \{a_{ij}^k\} \leq \{a_{i0}^k\}, \\ a_{ij}^k, \qquad j \geq n_1 + 1, a_{ij}^k \geq 0, \\ \dfrac{\{a_{i0}^k\}}{1-\{a_{i0}^k\}}(-\{a_{ij}^k\}), \quad j \geq n_1 + 1, a_{ij}^k < 0, \end{cases} \qquad (3.7)$$

определяет правильное отсечение.

Доказательство приведено в [5].

Проверяем условие отсечения.

Для этого докажем, что оптимальное решение задачи (D_k, F), которое не является допустимым решением исходной задачи (3.1) – (3.4), не удовлетворяет условиям отсечения.

Допустим, что в оптимальном решении $X(D_k, F)$ величина $a_{i0}{}^k$ - нецелая и $1 \le i \le n_1$.

Ввиду того, что в оптимальном решении все небазисные переменные равны нулю (x_j =0, $j \in N_k$),

$$\sum_{j \in N_k} \gamma_j x_j = 0$$

В таком случае выражение (3.6) примет вид:

$$z = -\gamma_0 + \sum_{j \in N_k} \gamma_j x_j = -\gamma_0 + \sum_{j \in N_k} \gamma_j \cdot 0 = -\gamma_0 = -\{a_{i0}^k\}$$

Так как $0 < \{a_{i0}^k\} < 1,\ z < 0,$ то условие отсечения выполнено.

Сформулируем правило построения правильного сечения.

Допустим, что $X(D_k, F)$ не удовлетворяет условию (3.4) и $\left\| a_{ij}^k \right\|$ - симлексная таблица, соответствующая решению задачи (D_k, F).

Выбираем

$$i_0 = \min\{i \mid i \in (1, 2, ..., n); a_{i0}^k \text{ - не целое}\}$$

и строим правильное отсечение по формулам (3.5)-(3.7).

Известно доказательство конечности второго алгоритма [2].

Следует отметить, что второй алгоритм Гомори позволяет решать и полностью целочисленные задачи ЛП, но при этом не удастся достичь уровня эффективности первого алгоритма Гомори.

3.2 Численная реализация второго алгоритма Гомори

Блок-схема второго алгоритма аналогична блок-схеме первого алгоритма Р. Гомори и отличается лишь правилом построения коэффициентов правильного отсечения.

Следует напомнить, что в частично целочисленных задачах ЛП требование целочисленности накладывается не на все переменные, а на одну или некоторые из них.

Рассмотрим следующую задачу ЦЛП:

$$\max\left\{\begin{matrix} F\ \ x\ \ = \sum c_i x_i \mid \sum a_{ji} x_i\ \ \le = \ge\ \ b_j\,, \\ j\ =\ 1,m,\ x_i \ge 0\ \text{ для всех } i\ =\ 1,\ldots,n \end{matrix}\right\},\quad 3.8$$

При этом целочисленной должна быть только переменная x_k.

Симплекс-таблица будет иметь вид, представленный ниже.

Таблица 3.1 – Симплекс-таблица для решения задачи с помощью второго алгоритма Гомори

Базисные переменные	**Свободные члены**	**Небазисные переменные**			
		w_1	**w_2**	**…**	**w_n**
v_1	β_1	α_{11}	α_{12}	…	α_{1n}
v_2	β_2	α_{21}	α_{22}	…	α_{2n}
.	.	.	.	.	.
v_m	β_m	α_{m1}	α_{m2}	…	α_{mn}
F	β	c_1	c_2	…	c_n

В результате решения задачи с отброшенным условием целочисленности получена оптимальная симплексная таблица и переменной x_k соответствует строка базисной переменной v_k указанной таблицы.

В результате получим следующее уравнение:

$$\sum \alpha_{ki} w_i + v_k = \beta_k \qquad (3.9)$$

Выделим в β_k целую и дробную часть и преобразуем (3.9) к виду:

$$v_k - \beta_k = \beta_k - \sum \alpha_{ki} w_i \quad 3.10$$

Если значение $v_k = \beta_k$ дробное, то можно утверждать, об отсутствии допустимых целочисленных переменных решения в интервале $[\beta_k] < v_k < [\beta_k]+1$.

Тогда для целочисленной переменной v_k справедливо одно из следующих неравенств:

$$v_k \leq \beta_k \quad \text{или} \; v_k \geq \beta_k + 1$$

Следует учесть, левая часть уравнения (3.10) должна быть целой.

Необходимо рассмотреть условия, при которых правая часть уравнения (3.10) также будет целочисленной.

Если $v_k \leq [\beta_k]$, то из (3.10) следует для любого $\alpha_{ki} \geq 0$, что:

$$\beta_k - \sum \alpha_{ki} w_i \leq 0 \; \textit{или} \; \sum \alpha_{ki} w_i \geq \beta_k \qquad 3.11$$

Если $v_k \geq [\beta_k] + 1$, то v_k - $[\beta_k] \geq 1$ и из (3.10) следует для любого $\alpha_{ki} \leq 0$:

$$\beta_k - \sum \alpha_{ki} w_i \geq 1 \; \textit{или} \; \sum \alpha_{ki} w_i \leq \beta_k - 1 \quad 3.12$$

Умножив левую и правую части неравенства (3.12) на

$$\frac{\{\beta_k\}}{\{\beta_k\} - 1} < 0, \text{ получим}$$

$$\frac{\{\beta_k\}}{\{\beta_k\} - 1} \sum \alpha_{kj} w_i \geq \{\beta_k\} \qquad (3.13)$$

Ввиду того, что соотношения (3.11) и (3.13) не могут выполняться одновременно, их можно объединить в одно ограничение вида:

$$\sum_{i \in I^+} \alpha_{kj} w_i + \frac{\{\beta_k\}}{\{\beta_k\} - 1} \sum_{i \in I^-} \alpha_{kj} w_i \geq \{\beta_k\}, \qquad (3.14)$$

где:

I^+ - множество значений i, для которых выполняется условие $\alpha_{ki} > 0$;

I^- – множество значений i, для которых выполняется условие $\alpha_{ki} < 0$.

Описанное решение может быть положено в основу компьютерной реализации второго алгоритма Гомори.

3.3 Сравнительный анализ первого и второго алгоритмов Гомори

3.3.1 Сравнение характеристик первого и второго алгоритмов Гомори

Для сравнения первого и второго алгоритмов Гомори алгоритмов произведен анализ основных характеристик указанных алгоритмов [21].

Результаты сравнительного анализа первого и второго алгоритмов Гомори сведены в таблицу таблица 3.2.

Таблица 3.2 – Сравнительный анализ характеристик первого и второго алгоритмов Гомори

Алгоритм/ Характеристика	**1-й алгоритм Гомори**	**2-й алгоритм Гомори**
Метод, положенный в основу	метод отсечений	метод отсечений
Область решаемых задач	только полностью целочисленные задачи ЛП	частично целочисленные задачи ЛП, а также некоторые целочисленные задачи
Достоинства	более высокая эффективность при решении полностью целочисленных задач ЛП; относительная простота программной реализации	большая универсальность; используется качественный способ улучшения ограничений.

Продолжение таблицы 3.2

Недостатки	требование целочисленности для основных и дополнительных переменных; трудоемкость; ошибки округления, которые возникают в процессе вычислений, в некоторых случаях приводят к получению неправильного (неоптимального) целочисленного решения; решения, последовательно получаемые в процессе выполнения алгоритма, не являются допустимыми. Иными словами, алгоритм не позволяет получать какое-либо целочисленное решение, отличное от оптимального значения.	низкая эффективность при решении полностью целочисленных задач ЛП.

Общими недостатками первого и второго алгоритмов Гомори являются:

− трудоемкость;

− не обеспечивают высокую эффективность соответствующих вычислительных процедур;

− не подходят для решения целочисленных задач больших размерностей;

− не обеспечивают эффективное решение общей задачи ЦЛП.

К общими достоинствам первого и второго алгоритмов Гомори можно отнести неослабевающий интерес ученых к проблеме улучшения метода отсечений.

Таким образом, алгоритмы Гомори существенно расширяют возможности для решения задач ЦЛП, но не упрощают данный процесс.

3.3.2 Программная реализация первого и второго алгоритмов Гомори

В качестве средства для сравнительного анализа использована программа, разработанная на основе свободно-распространяемого онлайн-сервиса [9] и технологии Electron [13].

Electron - это библиотека с открытым исходным кодом, разработанная GitHub для создания кросс-платформенных настольных приложений с HTML, CSS и JavaScript. Electron выполняет это, объединяя Chromium и Node.js в единую среду выполнения, и приложения могут быть упакованы для Mac, Windows и Linux.

Компоненты технологии Electron:

JavaScript - высокоуровневый, интерпретируемый язык программирования. Это язык, который также характеризуется как динамический, слабо типизированный, прототипный и мультипарадигмный. JavaScript наряду с HTML и CSS является одной из трех основных технологий World Wide Web. JavaScript позволяет создавать интерактивные веб-страницы и, таким образом, является неотъемлемой частью веб-приложений.

Chromium - это проект веб-браузера с открытым исходным кодом, созданный Google, для предоставления исходного кода для собственного браузера Google Chrome. Два браузера разделяют большинство кодов и функций, хотя есть некоторые незначительные отличия в функциях и логотипах, и они имеют различное лицензирование.

Node.js - это среда с открытым исходным кодом, кросс-платформенная среда выполнения JavaScript, которая выполняет JavaScript-код на стороне сервера. Обычно JavaScript использовался прежде всего для сценариев на стороне клиенты, в которых скрипты языка встроены в HTML-страницу веб-страницы и запускаются клиентской стороной с помощью механизма JavaScript в веб-браузере пользователя [11].

Node.js позволяет разработчикам использовать JavaScript для серверных скриптовых сценариев на стороне сервера для создания динамического содержимого веб-страницы до того, как страница будет отправлена в веб-браузер пользователя.

Следовательно, Node.js представляет собой парадигму «JavaScript везде», унификацию разработки веб-приложений вокруг одного языка программирования, а не разные языки для сценариев на стороне сервера и на стороне клиента.

Процесс разработки программы состоит из следующих этапов:

1) используется ссылка на онлайн-сервис;

2) добавляется дополнительный код на JavaScript и Css для скрытия лишних элементов (рекламы, баннеров и т.п.), а также чтобы заставить работать сервис как десктоп-приложение (обязательно наличие Интернета для нормальной работы).

На рисунке 3.1 представлен жизненный цикл приложения разработанного в технологии Electron.

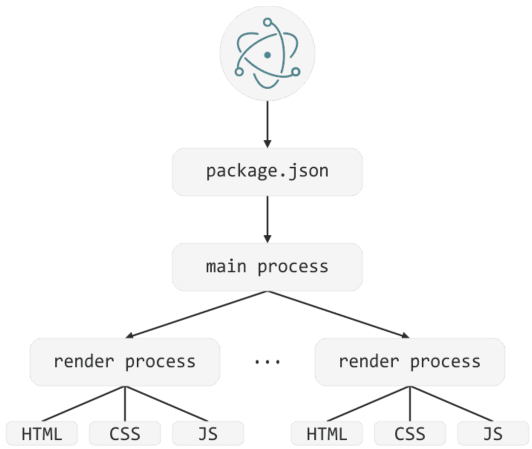

Рисунок 3.1 – Жизненный цикл Electron-приложения

На рисунке 3.2 представлена диаграмма вариантов использования программной реализации алгоритмов Гомори.

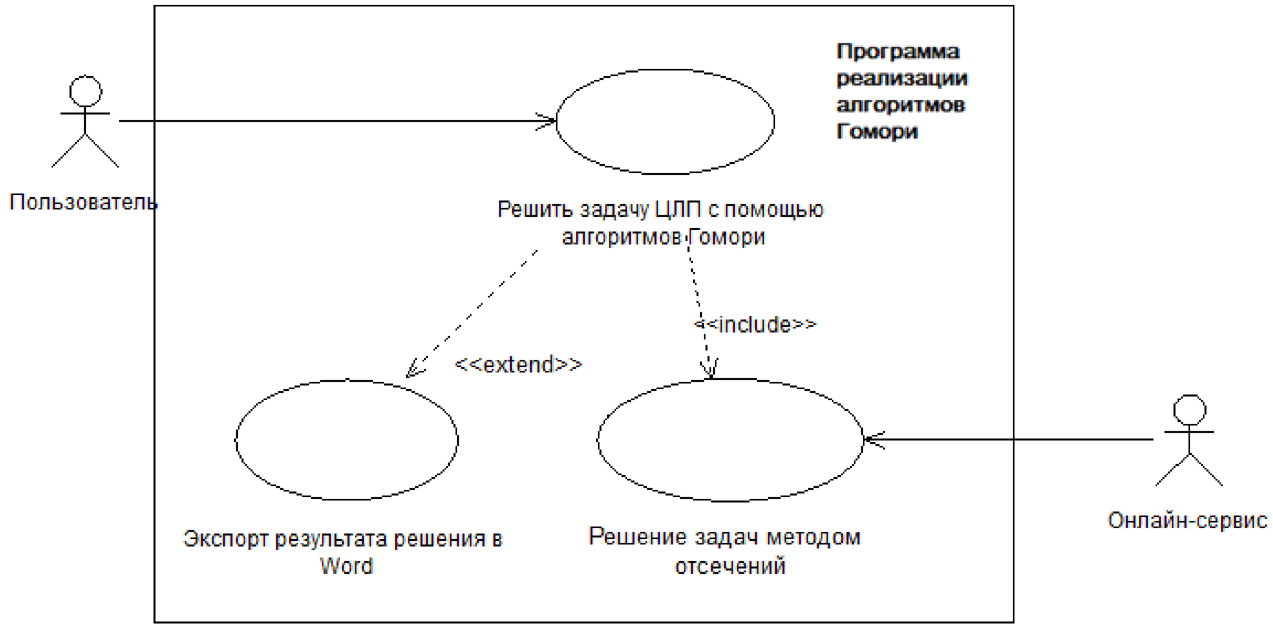

Рисунок 3.2 – Диаграмма вариантов использования программной реализации алгоритмов Гомори

На рисунке 3.3 представлена диаграмма компонентов приложения

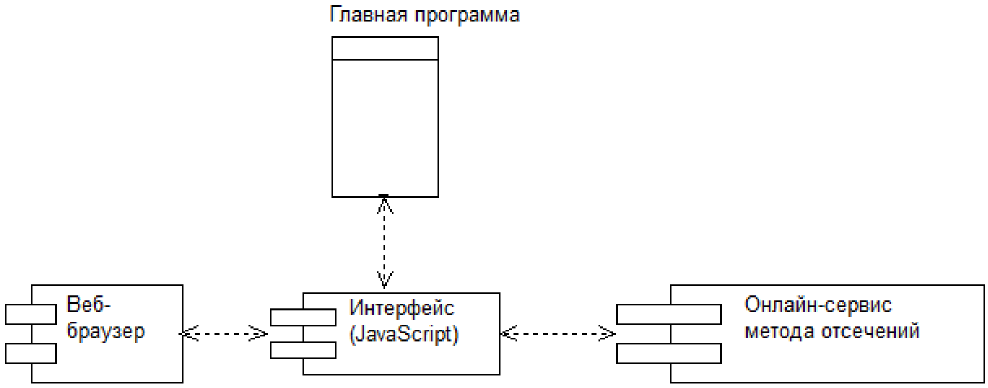

Рисунок 3.3 – Диаграмма компонентов программы реализации алгоритмов Гомори

Ниже приведен фрагмент кода программы на языке JavaScript приведен в Приложении А.

На рисунке 3.4 представлен интерфейс сервиса для решения задач ЦЛП методом отсечений.

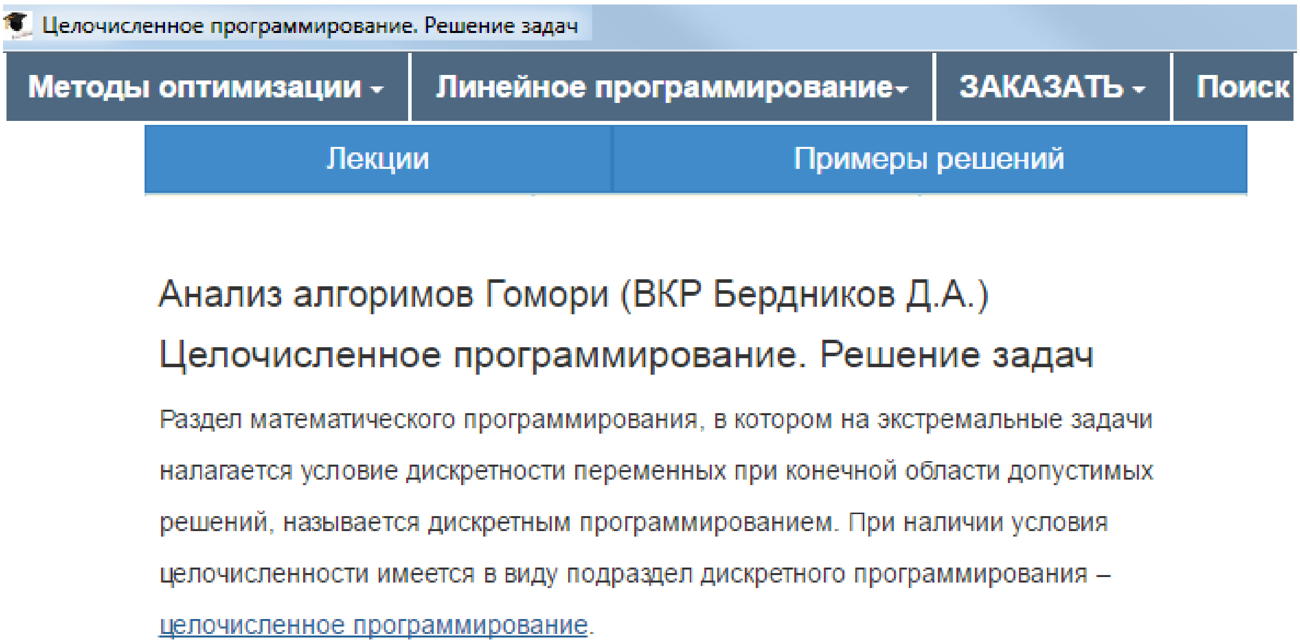

Рисунок 3.4 - Интерфейс сервиса для решения задач ЦЛП методом отсечений

На представленной программе выполнены тестовые решения нижеследующей задачи с помощью первого и второго алгоритмов Гомори:

Задача. Найти максимум целевой функции:

$F(X) = x_1+2x_2$ при ограничениях: $5x_1+7x_2 \le 21$ и $-x_1+3x_2 \le 8$.

Результат решения задачи с помощью первого алгоритма Гомори (таблицы 3.2-3.11).

Таблица 3.2 – Симлекс-таблица 1

Базис	B	x_1	x_2	x_3	x_4
x_3	21	5	7	1	0
x_4	8	-1	3	0	1
F(X0)	0	-1	-2	0	0

Итерация №0.

Таблица 3.3 – Симлекс-таблица 2

Базис	B	x_1	x_2	x_3	x_4	min
x_3	21	5	7	1	0	3
x_4	8	-1	**3**	0	1	**$2\,^{2}/_{3}$**
F(X1)	0	-1	**-2**	0	0	

Таблица 3.4 – Симлекс-таблица 3

Базис	B	x_1	x_2	x_3	x_4
x_3	$^{7}/_{3}$	$^{22}/_{3}$	0	1	$^{-7}/_{3}$

Продолжение таблицы 3.4

x_2	$^{8}/_{3}$	$^{-1}/_{3}$	1	0	$^{1}/_{3}$
F(X1)	$^{16}/_{3}$	$^{-5}/_{3}$	0	0	$^{-2}/_{3}$

Итерация №1.

Таблица 3.5 – Симлекс-таблица 4

Базис	B	x_1	x_2	x_3	x_4	min
x_3	$^{7}/_{3}$	$7^{1}/_{3}$	0	1	$^{-7}/_{3}$	$^{7}/_{22}$
x_2	$^{8}/_{3}$	$^{-1}/_{3}$	1	0	$^{1}/_{3}$	-
F(X2)	$^{16}/_{3}$	$-1^{2}/_{3}$	0	0	$^{2}/_{3}$	

Таблица 3.6 – Симлекс-таблица 5

Базис	B	x_1	x_2	x_3	x_4
x_1	$^{7}/_{22}$	1	0	$^{3}/_{22}$	$^{-7}/_{22}$
x_2	$^{61}/_{22}$	0	1	$^{1}/_{22}$	$^{5}/_{22}$
F(X2)	$^{129}/_{22}$	0	0	$^{5}/_{22}$	$^{3}/_{22}$

Таблица 3.7 – Симлекс-таблица 6

Базис	B	x_1	x_2	x_3	x_4
x_1	$^{7}/_{22}$	1	0	$^{3}/_{22}$	$^{-7}/_{22}$
x_2	$^{61}/_{22}$	0	1	$^{1}/_{22}$	$^{5}/_{22}$
F(X3)	$^{129}/_{22}$	0	0	$^{5}/_{22}$	$^{3}/_{22}$

Оптимальный план можно записать так:

$x_1 = {}^{7}/_{22}$, $x_2 = 2^{17}/_{22}$

$F(X) = 1 \bullet {}^{7}/_{22} + 2 \bullet 2^{17}/_{22} = 5^{19}/_{22}.$

Таблица 3.8 – Симлекс-таблица 7

Базис	B	x_1	x_2	x_3	x_4	x_5
x_1	$^{7}/_{22}$	1	0	$^{3}/_{22}$	$^{-7}/_{22}$	0
x_2	$^{61}/_{22}$	0	1	$^{1}/_{22}$	$^{5}/_{22}$	0
x_5	$^{-17}/_{22}$	0	0	$^{-1}/_{22}$	$^{-5}/_{22}$	1
F(X0)	$^{-129}/_{22}$	0	0	$^{-5}/_{22}$	$^{-3}/_{22}$	0

Таблица 3.9 – Симлекс-таблица 8

Базис	B	x_1	x_2	x_3	x_4	x_5
x_1	$^{7}/_{22}$	1	0	$^{3}/_{22}$	$^{-7}/_{22}$	0
x_2	$^{61}/_{22}$	0	1	$^{1}/_{22}$	$^{5}/_{22}$	0
x_5	$^{-17}/_{22}$	0	0	$^{-1}/_{22}$	$^{-5}/_{22}$	1
F(X0)	- $^{129}/_{22}$	0	0	$^{-5}/_{22}$	$^{-3}/_{22}$	0
θ		-	-	$^{-5}/_{22} : (^{-1}/_{22}) = 5$	$^{-3}/_{22} : (^{-5}/_{22}) = ^{3}/_{5}$	-

Выполняем преобразования симплексной таблицы методом Жордано-Гаусса:

Таблица 3.10 – Симлекс-таблица 9

Базис	B	x_1	x_2	x_3	x_4	x_5
x_1	$^{7}/_{5}$	1	0	$^{1}/_{5}$	0	$^{-7}/_{5}$
x_2	2	0	1	0	0	1
x_4	$^{17}/_{5}$	0	0	$^{1}/_{5}$	1	$^{-22}/_{5}$
F(X0)	$^{-27}/_{5}$	0	0	$^{-1}/_{5}$	0	$^{-3}/_{5}$

Таблица 3.11 – Симлекс-таблица 10

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_1	$^{7}/_{5}$	1	0	$^{1}/_{5}$	0	$^{-7}/_{5}$	0
x_2	2	0	1	0	0	1	0
x_4	$^{17}/_{5}$	0	0	$^{1}/_{5}$	1	$^{-22}/_{5}$	0
x_6	$^{-2}/_{5}$	0	0	$^{-1}/_{5}$	0	$^{-3}/_{5}$	1
F(X0)	$^{-27}/_{5}$	0	0	$^{-1}/_{5}$	0	$^{-3}/_{5}$	0

Таблица 3.12 – Симлекс-таблица 11

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_1	$^{7}/_{5}$	1	0	$^{1}/_{5}$	0	$^{-7}/_{5}$	0
x_2	2	0	1	0	0	1	0
x_4	$^{17}/_{5}$	0	0	$^{1}/_{5}$	1	$^{-22}/_{5}$	0
x_6	$^{-2}/_{5}$	0	0	$^{-1}/_{5}$	0	$^{-3}/_{5}$	1
F(X0)	$^{-27}/_{5}$	0	0	$^{-1}/_{5}$	0	$^{-3}/_{5}$	0
θ		-	-	$^{-1}/_{5} : (^{-1}/_{5}) = 1$	-	$^{-3}/_{5} : (^{-3}/_{5}) = 1$	-

Таблица 3.13 – Симлекс-таблица 12

Базис	B	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1	1	0	0	0	-2	1
x_2	2	0	1	0	0	1	0
x_4	3	0	0	0	1	-5	1
x_3	2	0	0	1	0	3	-5
F(X0)	-5	0	0	0	0	0	-1

Решение получилось целочисленным.

Оптимальный целочисленный план можно записать так:

$x_1 = 1$, $x_2 = 2$

F(X) = 1•1 + 2•2 = 5.

Задача решена.

2. Решение задачи с помощью второго алгоритма Гомори (таблицы 3.14-3.19).

Таблица 3.14 – Симлекс-таблица 1

Базис	B	x_1	x_2	x_3	x_4
x_3	21	5	7	1	0
x_4	8	-1	3	0	1

Продолжение таблицы 3.14

F(X0)	0	-1	-2	0	0

Итерация №0.

Таблица 3.15 – Симлекс-таблица 2

Базис	B	x_1	x_2	x_3	x_4	min
x_3	21	5	7	1	0	3
x_4	8	-1	3	0	1	$2\,^{2}/_{3}$
F(X1)	0	-1	-2	0	0	

Таблица 3.16 – Симлекс-таблица 3

Базис	B	x_1	x_2	x_3	x_4
x_3	$^{7}/_{3}$	$^{22}/_{3}$	0	1	$^{-7}/_{3}$
x_2	$^{8}/_{3}$	$^{-1}/_{3}$	1	0	$^{1}/_{3}$
F(X1)	$^{16}/_{3}$	$^{-5}/_{3}$	0	0	$^{2}/_{3}$

Итерация №1.

Таблица 3.17 – Симлекс-таблица 4

Базис	B	x_1	x_2	x_3	x_4	min
x_3	$^7/_3$	$7^1/_3$	0	1	$^{-7}/_3$	$^7/_{22}$
x_2	$^8/_3$	$^{-1}/_3$	1	0	$^1/_3$	-
F(X2)	$^{16}/_3$	$-1^2/_3$	0	0	$^2/_3$	

Таблица 3.18 – Симлекс-таблица 5

Базис	B	x_1	x_2	x_3	x_4
x_1	$^7/_{22}$	1	0	$^3/_{22}$	$^{-7}/_{22}$
x_2	$^{61}/_{22}$	0	1	$^1/_{22}$	$^5/_{22}$
F(X2)	$^{129}/_{22}$	0	0	$^5/_{22}$	$^3/_{22}$

Таблица 3.19 – Симлекс-таблица 6

Базис	B	x_1	x_2	x_3	x_4
x_1	$^7/_{22}$	1	0	$^3/_{22}$	$^{-7}/_{22}$
x_2	$^{61}/_{22}$	0	1	$^1/_{22}$	$^5/_{22}$

Продолжение таблицы 3.19

F(X3)	$^{129}/_{22}$	0	0	$^{5}/_{22}$	$^{3}/_{22}$

Оптимальный план можно записать так:

$x_1 = {}^{7}/_{22}$, $x_2 = 2^{17}/_{22}$

$F(X) = 1 \bullet {}^{7}/_{22} + 2 \bullet 2^{17}/_{22} = 5^{19}/_{22}$

Оптимальный целочисленный план можно записать так:

$x_1 = {}^{7}/_{22}$

$x_2 = 2^{17}/_{22}$

$F(X) = 5^{19}/_{22}$

Задача решена.

ЗАКЛЮЧЕНИЕ

Представленная бакалаврская работа посвящена актуальной проблеме анализа и реализации алгоритма Гомори.

В ходе выполнения бакалаврской работы достигнуты следующие результаты:

1) Проанализированы теоретические основы метода Гомори и описан метод отсечений для решения задач ЦЛП, положенный в основу алгоритмов Гомори.
2) Приведен анализ эффективности отсечения Гомори.
3) Даны описание и анализ первого и второго алгоритмов Гомори.
4) Представлены подходы к численной реализации первого и второго алгоритмов Гомори.
5) Произведено сравнение основных характеристик первого и второго алгоритмов Гомори.
6) На основе технологии Electron и бесплатно распространяемого веб-сервиса решения задач ЛП разработана программа реализации первого и второго алгоритмов Гомори.
7) Выполнено тестовое решение задачи ЛП на разработанной программе.

Результаты бакалаврской работы могут быть рекомендованы для решения задач ЦЛП, связанных с применением алгоритмов Гомори.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем.

Научная и методическая литература

2. Алексеева Е. В., Кутненко О. А., Плясунов А. В. Численные методы оптимизации: Учеб. пособие / Е. В. Алексеева, О. А. Кутненко, А. В. Плясунов/ - Новосибирск: Новосиб. ун-т, 2008. - 128 с.

3. Гераськин М.И. Линейное программирование: учеб. пособие / М.И. Гераськин, Л.С. Клентак; под общ. ред. Л.С. Клентак. – Самара: Изд-во СГАУ, 2014. – 104 с.

4. Землянухина Л.Н. Методические указания для студентов дневного и вечернего отделений механико-математического факультета по курсу «Методы оптимизации «Линейное программирование и смежные вопросы» / Л.Н. Землянухина, А.Б.Зинченко, Л.И. Сантылова. - Ростов-на-Дону, 1998. - 36 с.

5. Лемешко Б.Ю. Теория игр и исследование операций: Конспект лекций / Б.Ю. Лемешко. – Новосибирск: Изд-во НГТУ, 2013. – 167 с.

Электронные ресурсы

6. Кириллов Ю.В. Прикладные методы оптимизации. Часть 1. Методы решения задач линейного программирования [Электронный ресурс] : учебное пособие / Ю.В. Кириллов, С.О. Веселовская. - Новосибирск: Новосибирский осударственный технический университет, 2012. - 235 с. - Режим доступа: http://www.iprbookshop.ru/45430.html (дата обращения 30.04.2018).

7. Методы принятия оптимальных решений. Часть 1 [Электронный ресурс] : учебное пособие / Р.М. Безбородникова [и др.]. - Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2016. - 245 с. - Режим доступа: http://www.iprbookshop.ru/69912.html (дата обращения 30.04.2018).

8. Онлайн-сервис «Все калькуляторы онлайн» https://math.semestr.ru/simplex/partially-integer.php

9. Пантелеев А.В. Методы оптимизации [Электронный ресурс] : учебное пособие / А.В. Пантелеев, Т.А. Летова. - М. : Логос, 2011. - 424 с. - Режим доступа: http://www.iprbookshop.ru/9093.html (дата обращения 30.04.2018).

10. Платформа Node.js [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Node.js (дата обращения 30.04.2018).

11. Решение задач целочисленного программирования: методы и примеры [Электронный ресурс]. - Режим доступа: https://function-x.ru/zadacha_celochislennogo_programmirovanija.html (дата обращения 30.04.2018).

12. Технология Electron [Электронный ресурс]. - Режим доступа: https://electronjs.org (дата обращения 30.04.2018).

13. Хаггарти Р. Дискретная математика для программистов [Электронный ресурс] : учебное пособие / Р. Хаггарти. - М. : Техносфера, 2012. - 400 с. - Режим доступа: http://www.iprbookshop.ru/12723.html (дата обращения 30.04.2018).

14. Электронный учебник «Экономико-математические методы» [Электронный ресурс]. - Режим доступа: http://www.math.mrsu.ru/text/courses/method/index.htm (дата обращения 30.04.2018).

15. Branch-and-bound and Cutting Plane methods [Электронный ресурс]. - Режим доступа: http://www.cse.chalmers.se/~grohe/ashkanp/Session10.pdf (дата обращения 30.04.2018).

16. Integer programming problem and algorithms [Электронный ресурс]. - Режим доступа: http://shodhganga.inflibnet.ac.in/bitstream/10603/56771/8/08_chapter%202.pdf (дата обращения 30.04.2018).

17. Galli L. Algorithms for Integer Programming [Электронный ресурс]. - Режим доступа: http://www.di.unipi.it/optimize/Courses/RO2IG/aa1415/IP_algorithms.pdf (дата обращения 30.04.2018).

18. Gomory's cutting plane Algorithm [Электронный ресурс]. - Режим доступа: http://www.math.wsu.edu/faculty/genz/364/lessons/l603.pdf (дата обращения 30.04.2018).

19. Linear programming [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Linear_programming (дата обращения 30.04.2018).

20. MATH3220 Operations Research and Logistics Jan. 20, 2015 [Электронный ресурс]. - Режим доступа: https://www.math.cuhk.edu.hk/course_builder/1415/math3220/L5.pdf (дата обращения 30.04.2018).

Литература на иностранном языке

21. Gomory R.E. Outline of an Algorithm for Integer Solutions to Linear Programs, Bulletin of the American Mathematical Society 64, pp. 275-278, 1958.

22. Gomory R.E. An algorithm for integer solutions to linear programs, Recent Advances in Mathematical Programming. Robert L. Graves and Philip Wolfe, Editors. - N.Y.: McGraw-Hill, 1963, pp. 193−206.

23. Land A. H. and Doig A. G. An automatic method of solving discrete programming problems, 1960, pp. 497-520.

24. Qualizza A. Cutting planes for mixed integer programming. Dissertation, Tepper School of Business, Carnegie Mellon University, 2011.

25. Van den Broek P. Optimization of Product Instantiation using Integer. In Software Product Line Conference, volume 2, pp. 107–112, 2010.

ПРИЛОЖЕНИЕ А Код программы

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var methodOverride = require('method-override');
var calcResult = require('calc-result');
var jsonToString = require('json-to-string');
var db = {
    connection: null,
    start: function() {
        if (!db.connection) {
            var mysql      = require('mysql');
            var connection = mysql.createConnection({
              host     : 'localhost',
              user     : 'node_db',
              password : '28Unp1n30Cd'
            });
            connection.connect();
            db.connection = connection;
        }
    },
    end: function() {
        if (db.connection) {
            connection.end();
        }
    },
    query: function(query, data, callback) {
        if (data && !callback) {
            callback = data;
```

```
            data = null;
        }
        if (data) {
            connection.query(query, data, function(err, rows, fields) {
                callback(err, {
                    rows: rows,
                    fields: fields
                });
            });
        }
        else {
            connection.query(query, function(err, rows, fields) {
                callback(err, {
                    rows: rows,
                    fields: fields
                });
            });
        }
    }
}
var Users = {
    find: function(id, callback) {
        db.query('SELECT * FROM `user` WHERE `id`=\''+id+'\'', callback);
    }
}

var Steps = {
    get: function(stepInfo, callback) {
        db.query('SELECT * FROM `steps` WHERE `user_id`=\''+id+'\' AND
`step_id` = '+stepInfo.id+' ', callback);
```

```
    },
    save: function(stepInfo, callback) {
        db.query(
            'INSERT INTO `steps` (user_id, step_id, value, config) VALUES (?, ?, ?,
?)',
            [
                stepInfo.userId,
                stepInfo.id,
                stepInfo.value,
                jsonToString(stepInfo.config)
            ],
            callback
        );
    },
    calc: function(stepInfo, callback) {
        calcResult(stepInfo, function(calc){
            db.query(
                'INSERT INTO `last_calcs` (user_id, step_id, calc) VALUES (?, ?, ?, ?)',
                [
                    stepInfo.userId,
                    stepInfo.id,
                    jsonToString(calc)
                ],
                funciton(err, done) {
                    callback(err, calc);
                }
            );
        })
    }
}
```

```
function logErrors(err, req, res, next) {
  console.error(err.stack);
  next(err);
}
function clientErrorHandler(err,  req, res, next) {
  if (req.xhr) {
    res.status(500).send({ error: 'Server error' });
  } else {
    next(err);
  }
}
function errorHandler(err,  req, res, next) {
  res.status(500);
  res.render('error', { error: err });
}
app.use(bodyParser());
app.use(methodOverride());
app.use(logErrors);
app.use(clientErrorHandler);
app.use(errorHandler);
var myLogger = function (req, res, next) {
  console.log('LOGGED');
  next();
};
app.set('view engine', 'pug');
app.get('/', function (req, res) {
  res.render('index');
});
app.get('/auth', function auth(req, res, next) {
  Users.find(req.userId, function(err, user) {
```

```
    if(err) return next(err);
    res.json(user);
  });
});
app.get('/save-step', function getPaidContent(req, res, next) {
  Steps.save({
    userId: req.userId,
    step: req.stepInfo,
  }, function(err, step) {
    if(err) return next(err);
    res.json(step);
  });
});

app.get('/save-calc', function (req, res) {
  Steps.get({
    userId: req.userId,
    step: req.stepInfo,
  }, function(err, step) {
    if(err) return next(err);
    if (step.id == 4) {
        Steps.calc(step, function(err, calc) {
        if(err) return next(err);
        res.json(calc);
      });
    }
  });
});
import React, { Component } from 'react';
import './App.css';
```

```
import MatrixTable  from './MatrixTable.js';
import MatrixResult  from './MatrixResult.js';
import MatrixType  from './MatrixType.js';
const ExtraLink  = props => {
  const { kind, ...other } = props;
  const className = kind === "primary"  ? "ExtraLink"  : "ExtraLink2";
  return <a className={className}  {...other} />;
};
const StepInfo = props => {
  return <div className="StepInfo" {...props} />;
}
const StepForm = props => {
  return <form className="StepForm" {...props} />;
}
const NumberInput  = props => {
  return <input type="number"  {...props} />;
}
const StepNext = props => {
  const { handle, ...other } = props;
  return <button type="button" onClick={handle}>Далее</button>
}
class Step1 extends Component {
  constructor(props) {
    super(props);
    this.state = {
      cols: 3,
      rows: 3
    };
    this.nextStep = this.nextStep.bind(this);
    this.prevStep = this.prevStep.bind(this);
```

```
  }
  saveState() {
    this.props.saveState({
      cols: this.state.cols,
      rows: this.state.rows,
    });
  }
  nextStep(event) {
    this.saveState();
    this.props.nextStep();
  }
  prevStep(event) {
    this.saveState();
    this.props.prevStep();
  }
  render() {
    return (
      <div className="Step1">
        <StepInfo>
          <ExtraLink href="/instruction/">ИНСТРУКЦИЯ.</ExtraLink>
          «Выберите количество переменных и количество строк (количество
ограничений), нажмите Далее.
          Полученное решение сохраняется в файле Word (см. пример решения
методом Гомори).
          Дополнительно создается шаблон решения в формате Excel».
        </StepInfo>
        <StepForm>
          Количество переменных <NumberInput defaultValue="3"/>
          Количество строк (количество ограничений) <NumberInput
defaultValue="3" />
```

```
          <StepNext handle={this.nextStep}  />
          При этом ограничения типа xi ≥ 0 не учитывайте.
        </StepForm>
      </div>
    );
  }
}
class Step2 extends Component {
  constructor(props) {
    super(props);
    this.state = {
      rows: 3,
      cols: 3
    };
  }
```