Федеральное агентство по образованию

**Электростальский политехнический институт**

**(филиал) ФГОУ ВПО «Национальный исследовательский технологический университет «МИСиС»**

|  |  |
| --- | --- |
| *ФАКУЛЬТЕТ* | Экономики, менеджмента и информатика |
| *КАФЕДРА* | Прикладной информатики |
| *СПЕЦИАЛЬНОСТЬ* | 080801, Прикладная информатика (в экономике) |

**Дипломная**

**Работа**

на тему: «Система для решения задач математической экономики»

|  |  |  |
| --- | --- | --- |
| *Студент* |  | А. С. Прокопюк |
| *Руководитель работы* | (подпись, дата) | А. В. Жучин |
|  | (подпись, дата) |  |
| *Консультанты:* |  |  |
| *по экономике и управлению производством* |  | Н.Ф. Шеховцова |
| *по безопасности жизнедеятельности*  *и экологии* | (подпись, дата) | И.З. Вольшонок |
| *по стандартизации, метрологии*  *и нормоконтролю* | (подпись, дата) | Л. А. Тумарина |
|  | (подпись, дата) |  |
| *Рецензент* |  | В. В. Шемякин |
|  | (подпись, дата) |  |
| **Работа рассмотрена кафедрой и допущена к защите в ГАК** | | |
| (дата, № протокола) | | |
| *Заведующий кафедрой* |  | Г.Г. Чараев |
| *Декан факультета* | (подпись, дата) | О.М. Красильников |
|  | (подпись, дата) |  |

Электросталь 2010

# Ведомость дипломной работы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Формат | Обозначения | | Наименование | | | Кол-во листов\шт. | Примечания |
| А4 | 008.Д.2010.03.026.ПЗ | | Пояснительная записка | | | 77 |  |
| Диск | 008.Д.2010.03.026.СД | | Дипломный проект в электронном виде и презентация на CD | | | 1 |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  |  | |  | | |  |  |
|  | | | | | | | |
|  | | | Подпись | Дата | Ведомость дипломного проекта (работы) | | Литер У |
| Студент | | А. С. Прокопюк |  |  | Лист 1 |
| Рук. проекта (работы) | | А. В. Жучин |  |  | Листов 1 |
| Зав. кафедрой | | Г. Г. Чараев |  |  | Гр. ДИС-05-1 |

# Аннотация

Данная дипломная работа представляет собой систему для решения экономико-математических задач. Система имеет клиент-серверную архитектуру, для взаимодействия с пользователем используется веб-интерфейс. Все формы для ввода данных представляют собой шаблоны.

Для вычислений на сервере используются CGI-приложения, созданные с помощью совместной компиляции программного кода на языках программирования C++ и Fortran.

С помощью данной системы можно решить следующие задачи: задача линейного программирования, транспортная задача, задача о назначениях, игра двух лиц с нулевой суммой, игра с природой. К данным задачам сводятся многие более узкие экономико-математические задачи.

Система свободно доступна через сеть Интернет по адресу http://wemath.ru, для ее использования не нужны специальные знания. Благодаря использованию веб-интерфейса возможно использовать систему на различных аппаратных и программных платформах.

# Содержание

[Ведомость дипломной работы 3](#_Toc263968660)

[Аннотация 4](#_Toc263968661)

[Содержание 5](#_Toc263968662)

[Введение 7](#_Toc263968663)

[1 Аналитический обзор литературы 8](#_Toc263968664)

[1.1 Математическое обеспечение системы 8](#_Toc263968665)

[1.1.1 Математические методы в экономике 8](#_Toc263968666)

[1.1.2 Задача линейного программирования 9](#_Toc263968667)

[1.1.3 Транспортная задача 14](#_Toc263968668)

[1.1.4 Задача о назначениях 15](#_Toc263968669)

[1.1.5 Игра двух лиц с нулевой суммой 16](#_Toc263968670)

[1.1.6 Игра с природой 19](#_Toc263968671)

[1.2 Программное обеспечение и средства реализации 21](#_Toc263968672)

[1.2.1 Веб-технологии 21](#_Toc263968673)

[1.2.2 Язык программирования C++ 22](#_Toc263968674)

[1.2.3 Язык программирования Fortran 23](#_Toc263968675)

[1.2.4 Компиляция программ для ОС GNU/Linux, написанных на языках Fortran и C++ 24](#_Toc263968676)

[2 Описание системы для решения задач математической экономики 26](#_Toc263968677)

[2.1 Общее описание системы 26](#_Toc263968678)

[2.2 Конфигурация виртуальной среды и сервисов 28](#_Toc263968679)

[2.3 Общее описание интерфейса системы 31](#_Toc263968680)

[2.4 Интерфейс подсистемы решения задач 35](#_Toc263968681)

[2.4.1 Общее описание интерфейса подсистемы решения задач 35](#_Toc263968682)

[2.4.2 Задача линейного программирования 36](#_Toc263968683)

[2.4.3 Транспортная задача, задача о назначениях 39](#_Toc263968684)

[2.4.4 Задача решения игры двух лиц с нулевой суммой 43](#_Toc263968685)

[2.4.5 Решение задачи игры с природой 48](#_Toc263968686)

[2.5. Описание приложения для решения задачи линейного программирования 50](#_Toc263968687)

[2.5.1 Общее описание приложения 50](#_Toc263968688)

[2.5.2 Блок-схема приложения «Линейное программирование» 51](#_Toc263968689)

[2.5.3 XML для ввода и вывода данных 53](#_Toc263968690)

[2.5.6 Программный код приложения 55](#_Toc263968691)

[2.6 Решение экономико-математических задач с помощью разработанной системы 57](#_Toc263968692)

[2.6.1 Задача оптимального планирования производства 57](#_Toc263968693)

[2.6.2 Задача об оптимальном плане перевозок 62](#_Toc263968694)

[2.6.3 Задача о назначении рабочих на виды работ 64](#_Toc263968695)

[2.6.4 Задача об оптимальной стратегии поведения на фондовом рынке 65](#_Toc263968696)

[2.6.5 Задача о покупке акций 67](#_Toc263968697)

[3 Экономическая значимость системы 71](#_Toc263968698)

[4 Эргономическое обеспечение системы 74](#_Toc263968699)

[Заключение 76](#_Toc263968700)

[Список использованных источников 77](#_Toc263968701)

# Введение

Во время работы над КНИР были изучены варианты программного обеспечения, которое возможно использовать для решения экономико-математических задач. Большая часть существующих систем, пригодных для этого, являются платными решениями с высокой стоимостью. Также их недостатком можно считать то, что в связи с высокой универсальностью они имеют достаточно сложные правила ввода исходных данных – для их использования нужны не только знания экономико-математических задач, но и специальные знания, связанные с использованием конкретных систем. Дополнительным недостатком является привязанность каждой из систем к конкретной платформе – операционной системе и архитектуре вычислительной машины.

Указанные недостатки делают использование существующих систем затруднительным в небольших фирмах или учебных заведениях.

Таким образом, существует необходимость в системе, которая являлась бы бесплатной, легкой в использовании и независимой от платформы.

Для решения задачи разработки такой системы было решено использовать клиент-серверную архитектуру и веб-интерфейс, так как в большинстве современных операционных систем для большинства аппаратных платформ имеется программное обеспечение для просмотра веб-страниц. Это делает возможным использование системы вне зависимости от аппаратной или программной платформы.

Также было решено использовать шаблоны матриц, векторов и т. д. для ввода данных. Это делает возможным использование системы пользователями без специальных знаний.

Система должна решать следующие задачи: задача линейного программирования, транспортная задача, задача о назначениях, игра двух лиц с нулевой суммой, задача игры с природой.

# 1 Аналитический обзор литературы

## 1.1 Математическое обеспечение системы

### 1.1.1 Математические методы в экономике

Математические методы являются важнейшим инструментом анализа экономических явлений и процессов, построения теоретических моделей, позволяющих отобразить существующие связи в экономической жизни, прогнозировать поведение экономических субъектов и экономическую динамику. Математическое моделирование становится языком современной экономической теории, одинаково понятным для ученых всех стран мира [3].

Математическая экономика – сфера научной и практической деятельности, целью которой является математически формализованное описание экономических объектов, процессов и явлений.

Математика как основа теории принятия решений широко применяется для управления (планирования, прогнозирования, контроля) экономическими объектами и процессами [1].

В данной работе используются методы, относящиеся к таким разделам экономико-математического знания как математическое программирование и теория игр.

Математическое программирование – математическая дисциплина, изучающая теорию и методы решения задач о нахождении экстремумов функций на множествах конечномерного векторного пространства, определяемых линейными и нелинейными ограничениями (равенствами и неравенствами) [10].

Конкретно в данной работе рассматривается такой раздел математического программирования как линейное программирование – частный случай математического программирования, когда целевая функция и ограничения линейны [1][2][3][4][10].

Теория игр – математический метод изучения оптимальных стратегий в играх. Под игрой понимается процесс, в котором участвуют две и более сторон, ведущих борьбу за реализацию своих интересов. Каждая из сторон имеет свою цель и использует некоторую стратегию, которая может вести к выигрышу или проигрышу – в зависимости от поведения других игроков [1][2][3][10].

В данной работе рассматриваются такие виды игр как игра двух сторон с нулевой суммой и игра с природой.

### 1.1.2 Задача линейного программирования

Задача линейного программирования – это задача математического программирования, где целевая функция и ограничения линейны. Общая задача линейного программирования может быть сформулирована следующим образом [4]:

Здесь - множество индексов переменных, на которых наложено условие неотрицательности. Наиболее распространенным численным методом для решения задачи линейного программирования является симплекс-метод. Для решения задачи симплекс-методом ее необходимо преобразовать к канонической форме.

Задача линейного программирования в канонической форме выглядит следующим образом:

Так же есть следующие требования к задаче [3][4].

1. Ранг матрицы *A* должен быть равен количеству ее строк.
2. Количество переменных *n* должно превышать количество уравнений в системе ограничений *m*. В случае если *m = n*, система имеет единственное решение, которое может быть решением задачи только в случае, если выполняется условие неотрицательности. В случае, если *m > n*, и в системе уравнений нет линейно зависимых, которые могут быть удалены, система несовместна.

Если выполняются эти условия, множество решений системы ограничений представляет собой выпуклый многогранник, в угловых точках которого лежат решения. Алгоритм симплекс-метода позволяет, начав путь из произвольной угловой точки этого множества, найти такую угловую точку, где выполняется условие минимума или максимума целевой функции. Поиск начальной угловой точки является дополнительной задачей.

Возможные ситуации при решении задачи линейного программирования перечислены ниже [3][4].

1. Уравнения системы ограничений несовместны; в этом случае множество допустимых решений пусто, и задача не может быть решена.
2. Множество допустимых решений не ограничивает целевую функцию со стороны возрастания либо убывания. В данном случае, если происходит поиск максимума, и область не ограничивает функцию сверху, . В случае, если происходит поиск минимума, и область не ограничивает функцию снизу, . В данном случае решение задачи не имеет смысла; данный случай может быть распознан по специальному правилу в ходе решения задачи симплекс-методом.
3. Задача имеет единственное решение, либо решением задачи является любая линейная комбинация двух точек.

Так же специально нужно выделить случай, когда множество решений является вырожденным, то есть, среди угловых точек есть такие, где базисные переменные равны нулю. В таком случае при решении может произойти зацикливание симплекс-метода в связи с тем, что у вырожденной точки может быть больше одного базиса. Чтобы избежать зацикливания, необходимо при прохождении вырожденной точки выбирать разрешающий элемент по специальному лексикографическому правилу [4].

Наиболее распространенными методами поиска начальной угловой точки при решении задачи линейного программирования являются метод искусственного базиса и M-метод. В данной работе используется M-метод [4][5]. Он заключается в следующем: в задачу добавляется *m* искусственных переменных, которые должны выйти из базиса в ходе решения задачи. Они добавляются в целевую функцию с коэффициентом *M*, где *M* – достаточно большое число. Число *M* нужно выбирать так, чтобы оно одновременно служило штрафом за ввод искусственной переменной в базис и не инициировало излишнюю погрешность вычислений. К примеру, можно принять .

При использовании M-метода каноническая задача линейного программирования преобразуется к следующей задаче (в случае максимизации целевой функции) [5]:

В случае минимизации целевой функции, задача преобразуется к такой:

Здесь – единичная матрица размера , а .

В случае если после решения задачи среди базисных переменных окажутся искусственные, это будет означать, что ограничения несовместны.

С основной задачей линейного программирования тесно связана так называемая двойственная задача [3][4].

В случае если основная задача линейного программирования является задачей на максимизацию целевой функции, двойственная задача имеет следующий вид:

В случае минимизации двойственная задача имеет вид

Переменные *y* называются двойственными переменными или двойственными оценками.

Для взаимодвойственных задач возможны следующие случаи (первая теорема двойственности) [3].

1. В прямой и двойственной задачах имеются оптимальные решения, при этом значения целевых функций на оптимальных решениях совпадают: или .
2. В прямой задаче допустимое множество не пусто, но целевая функция на нем не ограничена. В этом случае у двойственной задачи будет пустое множество допустимых решений. Обратное так же верно.
3. Обе задачи имеют пустые множества допустимых решений.

Так же для этих задач выполняются условия дополняющей нежесткости (вторая теорема двойственности) [3]:

Условия дополняющей нежесткости позволяют сократить двойственную задачу, если известно решение прямой. После сокращения решение двойственной задачи сводится к решению системы линейных уравнений.

Экономический смысл двойственных оценок обусловлен следующим их свойством (теорема об оценках) [3]:

Экономический смысл двойственных оценок может быть следующим [3]:

1. мера дефицитности ресурсов (вектор правых частей);
2. мера влияния ограничений на целевую функцию;
3. инструмент определения эффективности отдельных вариантов;
4. инструмент балансирования суммарных затрат и результатов.

Если известны решения прямой и двойственной задач, можно провести анализ задачи на чувствительность к изменению элементов вектора правых частей ограничений или элементов вектора коэффициентов целевой функции [3].

При изменении элементов вектора правых частей нужно оценить верхнюю и нижнюю границы интервалов, в пределах которых можно изменять значения этих элементов так, чтобы в целом решение оставалось допустимым. Для этого нужно использовать следующие формулы:

Здесь , где – матрица, составленная из столбцов матрицы *A*, которые соответствуют коэффициентам при базисным переменным оптимального решения задачи.

Новое значение оптимального решения можно найти так:

Изменение целевой функции при изменении элементов вектора правых частей можно оценить с помощью теоремы об оценках.

Наиболее распространенными экономико-математическими задачами, решение которых производится с помощью линейного программирования, являются следующие задачи.

*Задача оптимального планирования производства [2][3][4][10].*

Пусть на некотором предприятии выпускают *n* видов продукции, при этом при их производстве используется *m* видов ресурсов, запасы которых составляют . Известна прибыль, которую получит предприятии от продажи каждого из видов продукции, она составляет . Так же известны нормативные затраты каждого из видов ресурсов на производство единицы каждого из видов продукции. Они составляют матрицу *A*. Необходимо найти такой выпуск продукции, чтобы прибыль была максимальна, и были выполнены ограничения по ресурсам.

Здесь за переменные примем объем выпуска каждого из видов продукции: .

Целевая функция, выражающая суммарную прибыль, выглядит следующим образом: , она должна быть максимизирована.

Ограничения по ресурсам выглядят следующим образом: .

*Задача об оптимальном использовании посевной площади [4].*

Имеется *m* земных угодий , предназначенных для засева той или иной сельскохозяйственной культурой. Эти площади отличаются либо положением, либо характером почвы. На каждом из угодий могут быть размещены одна или несколько из *n* сельскохозяйственных культур (пшеница, рожь, кукуруза, картофель и т. д.).

Пусть известна урожайность культуры *Qj* на поле *Si*; она равна *aij* центнеров с гектара. Будем обозначать площадь поля *Si* в гектарах через *di*. Ограничения в этой задаче таковы: задан план производства *bj* каждой сельскохозяйственной культуры *Qj*. Известны закупочные цены *cj* на каждый вид *Qj* продукции. Требуется определить план засева посевных площадей с целью максимизации дохода от продажи сельскохозяйственной продукции.

Пусть *xij*- площадь (в гектарах) на i-м поле, занятая культурой *Qj*.

Математическая модель задачи выглядит следующим образом.

Так же частными случаями задачи линейного программирования являются *транспортная задача* и *задача о назначениях*.

### 1.1.3 Транспортная задача

Транспортная задача – задача об оптимальном плане перевозок продукта из пунктов отправления в пункты потребления. Разработка и применение оптимальных схем грузовых потоков позволяют снизить затраты на перевозки [2][3][4][5].

Классическая транспортная задача – задача об оптимальном плане перевозок однородного продукта из однородных пунктов наличия в однородные пункты потребления на однородных транспортных средствах (предопределённом количестве) со статичными данными и линеарном подходе (это основные условия задачи).

Для классической транспортной задачи выделяют два типа задач: критерий стоимости (достижение минимума затрат на перевозку) или расстояний и критерий времени (затрачивается минимум времени на перевозку).

В математической форме транспортная задача формулируется следующим образом: обозначим через количество продукта, имеющееся у каждого из поставщиков, а через - потребность в продукте у каждого из потребителей. Через обозначим затраты денежных средств или времени на перевозку единицы продукции из пункта *i* в пункт *j*. Через обозначим количество продукции, которое следует перевезти из пункта *i* в пункт *j*. Тогда математическая модель будет следующей [5]:

Для решения этой задачи разработан специальный метод, называемый методом потенциалов. Но она также может быть решена и симплекс-методом.

### 1.1.4 Задача о назначениях

Классической постановкой задачи о назначениях является такая. Имеется *n* исполнителей работ, которых нужно распределить по *n* видам работ. При этом известны затраты на выполнение исполнителем *i* вида работ *j*: . Необходимо так распределить исполнителей по видам работ, чтобы суммарные затраты были минимальны [3].

Математическая модель такой задачи выглядит следующим образом:

Задача о назначениях является частным случаем транспортной задачи и, соответственно, частным случаем задачи линейного программирования. Для ее решения разработан специальный метод, называемый венгерским методом. Но она также может быть решена методом потенциалов и симплекс-методом.

### 1.1.5 Игра двух лиц с нулевой суммой

Постановка данной задачи следующая. В системе имеется два лица, принимающих решение – *A* и *B*, которые имеют *m* и *n* чистых стратегий соответственно. Известна платежная матрица *H*, в которой отражены выигрыши игрока *A*, которые так же являются «проигрышами» игрока *B* [1].

Решение задачи заключается в принятии решения о выборе игроками стратегий, гарантирующих равновесную ситуацию.

В случае если в платежной матрице есть седловая точка, решением является пара чистых стратегий. В случае если седловой точки нет, необходимо найти смешанные стратегии – поставить в соответствие каждой стратегии вероятность ее выбора.

Решение игры в смешанных стратегиях можно найти, используя линейное программирование, а именно следующим образом.

Необходимо найти решение двух взаимодвойственных задач:

После этого смешанные стратегии и ожидаемый выигрыш могут быть найдены так:

В виде игры двух лиц с нулевой суммой можно представить, например, следующие экономические задачи.

*Задача выбора оптимального момента времени для выставления товара на продажу в условиях конкуренции.*

Пусть на рынке какой-то однотипной продукции есть два продавца (*A* и *B*). При этом, товар, поступивший на этом рынке в продажу позже, имеет абсолютный приоритет у потребителей как более «свежий» - спрос на него становится абсолютным, в то время как спрос на более старый товар падает до нуля. Если же товары поступают в продажу одновременно, они пользуются одинаковым спросом.

Товары могут быть выставлены на продажу в моменты времени . Обозначим через *i* момент времени выставления на продажу товара продавцом *A*, а через *j –* продавцом *B.* Считая прибыль от продажи товара в единицу времени одинаковой для обоих продавцов, построим зависимости полученной прибыли от ситуации - пары «время продавца *A* – время продавца *B*» (прибыль от продажи товара в единицу времени обозначим через ).

1. Если , то продавец *A* один продает товар в течение времени и получает прибыль . Начиная с момента *j*, продажи продавца *A* упадут до нуля, а продавец *B* за оставшееся время получит прибыль .
2. Если , то продажи разделятся между продавцами поровну, и прибыль составит .
3. Если , то в течение времени продавец *B* получает прибыль . Начиная с момента *i*, продажи продавца *B* упадут до нуля, а продавец *A* за оставшееся время получит прибыль .

В качестве чистых стратегий «игроков» *A* и *B* выступают моменты времени *i* и *j,* платежная матрица *D* составляется с помощью показанных выше зависимостей. Постоянный множитель при этом можно отбросить. Игру рассмотрим со стороны продавца *A*:



В результате решения игры мы получим оптимальные смешанные стратегии *P\** и *Q\** для «игроков» *A* и *B* соответственно либо оптимальные чистые стратегии *A\** и *B\**, если в игре есть седловая точка. Так же мы получим величину ожидаемой прибыли *V*, которая достанется обоим игрокам.

*Определение объемов продаж, гарантирующих оптимальную прибыль в условиях игровой неопределенности потребительского спроса.*

Торговое предприятие может продавать *m* видов товаров, получая при этом прибыль, зависящую от спроса потребителей, который может быть в одном из *n* состояний. Известна матрица

где означает прибыль, которую получит предприятие при продаже единицы *i*-го товара с *j*-м состоянием спроса.

Требуется определить оптимальные объемы продаж для каждого из товаров при заданном общем объеме продаж *N*.

В конфликте «предложение продавца – потребительской спрос» с одной стороны выступает предприятие (игрок A), а с другой – потребительский спрос (игрок B).

За выигрыш игрока A при выборе им *i*-ой чистой стратегии (продаже только *i*-го вида товара в количестве *N*) при *j*-ом состоянии спроса примем прибыль . Таким образом, платежной матрицей рассматриваемой игры будет

Если продавать каждый из товаров в объеме , то, учитывая общий объем продаж, получим

Значит, доля *i*-го вида товара есть

Требуется определить эти доли , так как величины однозначно определяются через них.

можно трактовать как смешанную стратегию игрока A, причем

Смешанную стратегию игрока B (спрос) зададим вектором , где – вероятность выбора *j*-ой стратегии (состояния спроса).

Поскольку объем продаж *N* известен, то в качестве платежной матрицы будем рассматривать матрицу A.

### 1.1.6 Игра с природой

Ситуации, при которой либо ничего не известно об интересах второй стороны (или сторон), либо эти инте­ресы действительно отсутствуют (второй игрок - «природа»), харак­теризуются как ситуации принятия решений в условиях полной неоп­ределенности (или игры с «природой»). Естественно, что термин «природа» употребляется здесь в некотором символическом смысле как обозначение некой действительности, мотивы проявления которой неизвестны [1].

Принятие решения в игре с природой осуществляется с помощью специальных критериев. Наиболее известными критериями являются нижеперечисленные критерии.

1. *Максиминный критерий Валь­да.* Он отражает «принцип га­рантированного результата», то есть мы откладываемся на самый неблагоприятный для нас случай и пытаемся выбрать такую стратегию, которая мак­симизировала бы наш выигрыш в самой неблагоприятной для нас ситуации. Иногда этот критерий называют критерием «крайнего пессимизма».

2. *Критерий максимакса.* Он предполагает рассмотрение наиболее благоприятного случая. Выбирается в качестве оптимальной такая стратегия, для которой этот самый благоприятный случай дает самый большой выигрыш. Иногда этот критерий называют критерием «крайнего оптимизма».

3. *Критерий Гурвица.* Он представляет из себя целое семейст­во критериев, зависящих от некоторого параметра *а*, смысл которо­го—в определении баланса между подходами «крайнего пессимиз­ма» и «крайнего оптимизма». В качестве оптимальной стратегии выбирается стратегия, на которой дости­гается значение max. Значение параметра выбирается из интервала 0 < *а* < 1. Выбор конкретного значения параметра определя­ется скорее субъективными факторами, например склонностью к риску ЛПР (лица принимающего решение). При отсутствии каких-либо явных предпочтений вполне логично, например, выбрать значение *а =* 0,5.

4. *Критерий Сэвиджа (критерий минимаксного риска).* Применение данного критерия предполагает рассмотрение некоторой производной матрицы, смысл которой состоит в том, что для каждой стратегии второго игрока определяется выигрыш в наиболее благоприятном случае (при наиболее правильном выборе стратегии первым игроком для данной ситуации), а далее вычисляются величины «недополучен­ных» выигрышей для всех остальных стратегий первого игрока при рассматриваемой стратегии второго игрока. Далее к матрице рисков применяется минимаксный подход.

В качестве оптимальной стратегии выбирается стратегия, на которой достигается min. Тем самым мы выбираем такую стратегию, для которой наибольшее значение «недополучения» будет иметь наименьшее значение.

5. *Критерий Лапласа.* Этот критерий исходит из следующего соображе­ния. Поскольку нам ничего не известно о принципах или вероятно­стях применения вторым игроком своих стратегий, то мы предполагаем эти вероятности все равными. Таким образом, смысл данного критерия — максимизация ожидаемо­го выигрыша в предположении о равновероятности применения вторым игроком своих стратегий.

6. *Критерий Байеса.* В отличие от ситуации полной не­определенности весьма частой являет­ся ситуация, когда в распоряжении первого игрока есть информация о ве­роятностях применения стратегий вто­рой стороной. Эти вероятности назы­ваются априорными вероятностями. В качестве оптимальной выбирается стратегия, которая гарантирует наибольшее математическое ожидание выигрыша.

## 1.2 Программное обеспечение и средства реализации

### 1.2.1 Веб-технологии

Веб-сервером называют как компьютер, предоставляющий доступ к данным по протоколу HTTP, так и программное обеспечение, с помощью которого организуется доступ к этим данным.

Клиент, которым обычно является веб-браузер, передает веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы – это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP [6].

На ноябрь 2009 года по данным компании Netcraft двумя наиболее распространёнными веб-серверами, вместе занимающими около 68% рынка, являются:

* Apache — свободный веб-сервер, наиболее часто используемый в Unix-подобных операционных системах;
* IIS от компании Microsoft, распространяемый с ОС семейства Windows NT.

Как правило, веб-сервер как компьютер так же требует установки дополнительного программного обеспечения для администрирования. Это SSH-сервер для удаленного входа в систему, FTP-сервер для обмена файлами с сервером и специальное программное обеспечение, связанное со сбором статистики и другими операциями.

Веб-программирование – это раздел программирования, ориентированный на разработку динамических интернет-приложений. Языки веб-программирования – это соответственно языки, которые в основном предназначены для работы с интернет-технологиями. Языки веб-программирования делятся на две группы: клиентские и серверные.

Клиентские языки рассчитаны на написание сценариев, которые должны выполняться браузером на стороне клиента. Серверные же сценарии выполняются на сервере, а клиенты получают уже результат их выполнения.

Наиболее распространенным клиентским языком программирования является JavaScript. Наиболее распространенным серверным – PHP [7].

### 1.2.2 Язык программирования C++

C++ (Си++) – компилируемый статически типизированный язык программирования общего назначения. Поддерживает разные парадигмы программирования, но, в сравнении с его предшественником — языком Си, - наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования [9].

Название «C++» происходит от Си (C), в котором унарный оператор ++ обозначает инкремент переменной.

В 1990-х годах язык стал одним из наиболее широко применяемых языков программирования общего назначения.

При создании C++ стремились сохранить совместимость с языком Си. Большинство программ на Си будут исправно работать и с компилятором C++.

В книге «Дизайн и эволюция C++» Бьёрн Страуструп описывает принципы, которых он придерживался при проектировании C++. Эти принципы объясняют, почему C++ именно такой, какой он есть. Некоторые из них перечислены ниже.

* Получить универсальный язык со статическими типами данных, эффективностью и переносимостью языка Си.
* Непосредственно и всесторонне поддерживать множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование.
* Дать программисту свободу выбора, даже если это даст ему возможность выбирать неправильно.
* Максимально сохранить совместимость с Си, тем самым делая возможным лёгкий переход от программирования на Си.
* Избежать разночтений между Си и C++: любая конструкция, которая допустима в обоих этих языках, должна в каждом из них обозначать одно и то же и приводить к одному и тому же поведению программы.
* Избегать особенностей, которые зависят от платформы или не являются универсальными.
* Никакое языковое средство не должно приводить к снижению производительности программ, не использующих его.
* Не требовать слишком усложнённой среды программирования.

### 1.2.3 Язык программирования Fortran

Fortran – первый язык программирования высокого уровня, имеющий транслятор. Создан в период с 1954 по 1957 год группой программистов под руководством Джона Бэкуса в корпорации IBM. Название Fortran является аббревиатурой от FORmula TRANslator, то есть, переводчик формул. Фортран широко используется в первую очередь для научных и инженерных вычислений [8].

Фортран — жёстко стандартизированный язык, именно поэтому он легко переносится на различные платформы. Существует несколько международных стандартов языка.

FORTRAN 66 (1966). На базе стандарта фирмы IBM FORTRAN IV

FORTRAN 77 (1978). Множество улучшений: строковый тип данных и функции для его обработки, блочные операторы IF, ELSE IF, ELSE, END IF, оператор включения фрагмента программы INCLUDE и т. д.

Fortran 90 (1991). Значительно переработан стандарт языка. Введён свободный формат написания кода. Появились дополнительные описания IMPLICIT NONE, TYPE, ALLOCATABLE, POINTER, TARGET, NAMELIST; управляющие конструкции DO … END DO, DO WHILE, CYCLE, SELECT CASE, WHERE; работа с динамической памятью (ALLOCATE, DEALLOCATE, NULLIFY); программные компоненты MODULE, PRIVATE, PUBLIC, CONTAINS, INTERFACE, USE, INTENT. Появились новые встроенные функции, в первую очередь, для работы с массивами. В языке появились элементы ООП. Отдельно объявлен список устаревших черт языка, предназначенных для удаления в будущем.

Fortran 95 (1997). Коррекция предыдущего стандарта.

Fortran 2003 (2004). Дальнейшее развитие поддержки ООП в языке. Взаимодействие с операционной системой.

### 1.2.4 Компиляция программ для ОС GNU/Linux, написанных на языках Fortran и C++

Для достижения совместного использования этих языков программирования необходимо провести компиляцию программы компилятором “gfortran” с подключением библиотеки *stdc++*, при этом код C++ должен быть предварительно скомпилирован в объектный файл.

К примеру, для компиляции программы, реализующей решение задачи линейного программирования, код которой содержится в файлах cpp\_functions.cpp, linprog.for и linprog\_cgi.for, необходимо выполнить следующий сценарий:

*#!/bin/bash*

*#Создать объектный файл для подпрограмм на C++*

g++ cpp\_functions.cpp -c

*#Произвести компиляцию основной программы*

*#с использованием библиотек stdc++ и lapack*

gfortran cpp\_functions.o linprog.for linprog\_cgi.for -o linprog.cgi -ffree-form -ffree-line-length-none -lstdc++ -llapack

При совместном использовании разных языков программирования необходимо следить за совместимостью типов данных и способов передачи данных от основной программы подпрограмме. Так в языке C++ по умолчанию параметры передаются по значению, в Fortran – по ссылке. Типу *double* языка C++ в Fortran 95 соответствует тип *real(8)*, типу *bool* – *logical(1)*, а структуры, соответствующей многомерным массивам C++ в языке Fortran нет, что приводит к необходимости передавать данные через одномерный массив, с последующей перегруппировкой этих данных.

Совместное использование языков программирования – мощный инструмент, который в случаях, когда действительно требуются возможности каждого из языков, ведет к выигрышу в снижении трудоемкости написания кода, а в некоторых случаях и к снижению требований программы к ресурсам компьютера.

# 2 Описание системы для решения задач математической экономики

## 2.1 Общее описание системы

Разработанная система для решения задач математической экономики предназначена для решения задач на сервере с предоставлением пользователям клиентского интерфейса. Она функционирует по следующему принципу: ввод данных и вывод результатов осуществляется через web-интерфейс, обмен данными с сервером осуществляется в фоновом режиме (концепция AJAX), при этом данные для передачи структурируются с использованием формата XML. Расчеты проводятся на сервере с помощью CGI-приложений, созданных посредством совместной компиляции программного кода на языках Fortran 95 и C++.

Процесс решения задачи в данной системе можно представить в виде схемы, изображенной на рисунке 1.

Ввод исходных данных через web-интерфейс

Структурирование данных с помощью XML на стороне клиента

Отправка данных в CGI-приложение

Разбор XML с помощью функций C++ на стороне сервера

Расчет с помощью функций Fortran на стороне сервера

Формирование XML для отправки клиенту на стороне сервера

Разбор XML и вывод данных на стороне клиента

Программа, созданная с помощью совместной компиляции

Рисунок 1. Этапы решения задачи в разработанной системы.

Таким образом, разработанная система имеет клиент-серверную архитектуру с использованием «тонкого клиента», в роли которого выступает браузер.

В данный момент система доступна через сеть Интернет по доменному имени *wemath.ru*.

Для того чтобы система была доступна через сеть Интернет, было зарегистрировано доменное имя второго уровня «wemath» в домене первого уровня «ru», а так же был заключен договор с ООО «Агава-хостинг» на предоставление виртуального выделенного сервера (VPS). Законность этих сделок подтверждается соответствующими договорами. При оплате услуг в качестве платежного средства использовались электронные деньги, а именно «Яндекс.Деньги».

Выделенный виртуальный сервер основан на операционной системе Debian GNU/Linux и OpenVZ – программном средстве разделения единой среды ОС на множество виртуальных сущностей. OpenVZ позволяет на одном физическом сервере запускать множество изолированных копий операционной системы, называемых «виртуальные выделенные серверы» (Virtual Private Servers, VPS) или «виртуальные среды» (Virtual Environments, VE). Виртуальная среда реализует почти полную функциональность операционной системы.

Так же для функционирования системы необходимо дополнительное программное обеспечение: web-сервер с поддержкой интерпретации языка программирования PHP для обеспечения функционирования web-интерфейса, набор компиляторов GCC (Gnu Compiler Collection) для компиляции кода CGI-приложений, FTP-сервер для поддержки загрузки файлов на сервер. Для обеспечения расчетов так же необходимы библиотеки численных методов Atlas и Lapack, содержащие функции для решения задач линейной алгебры: решение систем линейных уравнений, LU-разложение, нахождение собственных чисел и т. д.

В качестве web-сервера был выбран Apache – наиболее распространенный web-сервер, к тому же он является свободным программным продуктом с открытым кодом. PHP подключается к Apache как модуль. В качестве FTP-сервера был выбран VSFTP – быстрый FTP-сервер, не отягощенный функциями, не нужными для решения текущей задачи.

Удаленное администрирование VPS осуществляется по протоколу SSH. На VPS установлен сервер OpenSSH, на удаленной машине используется программа PuTTY – клиент для SSH, Telnet и некоторых других протоколов удаленного доступа.

## 2.2 Конфигурация виртуальной среды и сервисов

Сервер VPS изначально был сконфигурирован достаточно для функционирования в сети и удаленного администрирования. Таким образом, его конфигурирование необходимо было начать с установки специального программного обеспечения:

* Aptitude – оболочка для APT, увеличивающая удобство установки и удаления программ.
* Chkconfig – утилита для управления автоматическим запуском сервисов при старте системы.
* VsFTP – FTP-сервер.
* Apache – web-сервер.
* PHP – интерпретатор языка программирования PHP.
* G++ - компилятор C++ из набора GCC.
* Gfortran – компилятор языка программирования Fortran 95 из набора GCC.

Далее необходимо сконфигурировать сервисы для правильного их функционирования.

**VsFTP** не требует специального конфигурирования за исключением настройки авторизации пользователей FTP. Для этого необходимо указать сервис авторизации. Есть два варианта: для авторизации используются данные о системных пользователях операционной системы либо для авторизации используются данные о виртуальных пользователях, данных о которых хранятся в специальной базе данных. По умолчанию используется первый метод, который и был выбран для разработанной системы. Помимо легкости реализации он так же удобен тем, что от имени системных пользователей может выполняться так же и работа с файлами модулем PHP веб-сервера, и в таком случае не возникнет трудностей с доступом веб-сервера на запись к файлам, загруженным через FTP.

Дополнительно, из соображений безопасности, необходимо в файле /etc/shells добавить строчку /bin/false – это программа, которая ничего не делает и возвращает код неудачного завершения. Эту программу необходимо указывать в качестве интерпретатора командной строки для пользователей, которых предполагается использовать исключительно для входа через ftp и использования для веб-сервера.

Конфигурирование **Apache** необходимо начать с установки дополнительного модуля «apache2 mpm-itk», который позволяет работать с файлами от имени разных пользователей для разных виртуальных хостов. Это нужно для того, чтобы отделить виртуальный хост, на котором выполняется веб-интерфейс системы, был отделен от других виртуальных хостов сервера (помимо системы для решения задач на сервере было размещено два сайта).

Далее необходимо создать в каталоге /etc/apache2/sites-avaible файл виртуального хоста и создать в каталоге /etc/apache2/sites-enabled символьную ссылку на него. В нашем случае это файл wemath.ru.vhost. Рассмотрим его содержание:

<VirtualHost \*:80>

**#Администратор сервера, доменное имя и псевдоним**

ServerAdmin mirror-xiii@rambler.ru

ServerName wemath.ru

ServerAlias www.wemath.ru

**#Параметры каталогов**

DocumentRoot /var/www/wemath.ru/web/

<Directory />

Options FollowSymLinks

AllowOverride None

</Directory>

<Directory /var/www/wemath.ru/web/>

Options Indexes FollowSymLinks MultiViews

AllowOverride All

Order allow,deny

allow from all

</Directory>

ScriptAlias /cgi-bin/ /var/www/wemath.ru/cgi-bin/

<Directory "/var/www/wemath.ru/cgi-bin/">

AllowOverride None

Order allow,deny

Allow from all

</Directory>

AddHandler cgi-script .cgi

AddHandler cgi-script .pl

**#Логи сервера**

ErrorLog /var/www/wemath.ru/log/apache\_error.log

# Possible values include: debug, info, notice, warn, error, crit,

# alert, emerg.

LogLevel warn

CustomLog /var/www/wemath.ru/log/access.log combined

**#Настройка пользователя и группы для работы с файлами**

<IfModule mpm\_itk\_module>

AssignUserId wemath-ru www-users

</IfModule>

</VirtualHost>

При настройке **PHP** для целей отладки в файле /etc/php5/apache2/php.ini необходимо назначить опции error\_reporting значение E\_ALL, что значит «выводить все сообщения об ошибках», это необходимо для целей отладки web-интерфейса.

Так же для целей безопасности сервера необходимо настроить **фильтрацию трафика**. Правила для фильтрации должны реализовывать следующую логику:

Возможно использование служб ssh, ftp и www. Так же возможно получение icmp-сообщений об ошибках и сообщения «echo-request». Возможен любой входящий трафик по соединениям, которые считаются уже установленными либо дочерними, а так же любой исходящий трафик и любой трафик по локальному интерфейсу (lo). Весь остальной трафик считается запрещенным.

Данную логику реализуют следующие правила для iptables:

**#Сброс**

iptables -F

**#Политики по умолчанию**

iptables -P INPUT DROP

iptables -P OUTPUT ACCEPT

iptables -P FORWARD DROP

**#Разрешить трафик по соединениям, считающимся уже установленными или дочерними**

iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

**#Разрешить ssh, ftp, www**

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A INPUT -p tcp --dport 21 -j ACCEPT

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

iptables -A INPUT -p tcp --dport 49152: -j ACCEPT

**#Разрешить трафик по локальному интерфейсу**

iptables -A INPUT -i lo -p ALL -j ACCEPT

**#Разрешить icmp**

iptables -I INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT

iptables -I INPUT -p icmp --icmp-type source-quench -j ACCEPT

iptables -I INPUT -p icmp --icmp-type time-exceeded -j ACCEPT

iptables -I INPUT -p icmp --icmp-type echo-request -j ACCEPT

Данные правила размещены в сценарии /etc/init.d/firewall.sh, что делает возможным их автоматическое добавление при загрузке системы.

## 2.3 Общее описание интерфейса системы

Интерфейс системы создан с помощью веб-технологий. Для разметки в нем используется язык XHTML 1.0 Strict, выполнение стандартов которого подтверждается валидатором W3C. Для обеспечения динамического изменения содержимого страниц и обмена данными с сервером используется язык программирования JavaScript.

Код интерфейса содержится в следующих каталогах и файлах:

/

/css/

- position.css *(css-код для создания структуры страниц интерфейса)*

- style.css *(css-код для стилизации элементов интерфейса)*

- lte\_ie7\_stylesheet.css *(css-код для старых версий браузера Internet Explorer)*

/images/

- *изображения для интерфейса*

/inc/

- top.php *(php- и html-код для верхней части страниц интерфейса)*

- bot.php *(php- и html-код для нижней части страниц интерфейса)*

/system/

- config.php *(параметры интерфейса системы)*

/jscripts/

- mirror-xiii\_ajax.js *(JavaScript-функции для фонового обмена данными с сервером)*

/missions/

- *файлы интерфейсов для каждой из задач, их необходимо описать отдельно*

Рисунок 2. – Каталоги и файлы веб-интерфейса.

Верхняя часть страниц интерфейса хранится в файле top.php, который посредством функции языка PHP include включается в начало каждого из файлов интерфейса. Рассмотрим код данного файла:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title><?=$page\_title?></title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=8" />

<link rel="stylesheet" type="text/css" href="<?=$project\_dir?>/css/position.css" />

<link rel="stylesheet" type="text/css" href="<?=$project\_dir?>/css/style.css" />

<!--[if lte IE 7]>

<link rel="stylesheet" type="text/css" href="/css/lte\_ie7\_stylesheet.css" />

<![endif]-->

<script type="text/javascript" src="/system/jscripts/mirror-xiii\_ajax.js"></script>

</head>

<body>

<table id="page">

<tr>

<td id="page\_td">

<table cellpadding="0" cellspacing="0" id="main\_table" border="0">

<tr>

<td id="page\_nav">

<p><b>Разделы:</b></p>

<p><a href="/">Начало</a></p>

<p><a href="/missions/linprog/">Линейное программирование</a></p>

<p><a href="/missions/transport/">Транспортная задача</a></p>

<p><a href="/missions/naznach/">Задача о назначениях</a></p>

<p><a href="/missions/game\_of\_two/">Игра двух лиц с нулевой суммой</a></p>

<p><a href="/missions/game\_with\_nat/">Игра с природой</a></p>

</td>

<td id="page\_main">

<div id="page\_content">

<div class="page\_content\_push"></div>

Здесь первая строчка <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> указывает на тип документа. В данном случае код необходимо интерпретировать в режиме соответствия стандарту XHTML 1.0 Strict. Атрибут xmlns тэга html указывает пространство имен XML, так как спецификация XHTML полностью соответствует стандартам XML.

Тэги страницы снабжены атрибутами id и class, которые позволяют поставить им в соответствие необходимые стилевые свойства.

Html- и css-код разрабатывался в соответствии со стандартами W3C, и был ориентирован на браузеры, поддерживающие эти стандарты. Такими являются все популярные современные браузеры, за исключением браузера Internet Explorer. Такими являются браузеры Mozilla Firefox 2 и выше, Opera 9 и выше, Google Chrome, Safari.

Браузер Internet Explorer до версии 8 включительно не поддерживает общепринятых стандартов, поэтому для поддержки интерфейсом этого браузера необходимо принять дополнительные меры. Поскольку с каждой новой версией разработчики Internet Explorer приближаются к поддержке стандартов, необходимо указать, что при интерпретации кода страницы необходимо использовать режим IE8. Это делается с помощью специального meta-тэга: <meta http-equiv="X-UA-Compatible" content="IE=8" />. Следующим шагом является включение в код стилей, рассчитанных на ранние версии браузера IE. Это делается с помощью условного комментария «if lte IE 7»:

<!--[if lte IE 7]>

<link rel="stylesheet" type="text/css" href="/css/lte\_ie7\_stylesheet.css" />

<![endif]-->

Завершается код шаблона в файле bot.php:

<div class="page\_content\_push\_bot"></div>

</div>

</td>

</tr>

</table>

</td>

</tr>

</table>

</body>

</html>

Данный код описывает верхнюю и нижнюю части страницы. В центре при этом находится рабочая область.

При создании интерфейса использовались как табличная, так и блочная техники верстки. Стилевые свойства, отвечающие за создание структуры страниц, находятся в файле positions.css. Рассмотрим его содержимое:

html, body

{

height: 100%;

margin: 0;

padding: 0;

}

#page

{

width: 100%;

height: 100%;

min-width: 900px;

max-width: 1300px;

margin: 0 auto;

border-collapse: collapse;

}

#page\_td

{

vertical-align: top;

height: 100%;

padding: 0;

}

#main\_table

{

width: 100%;

height: 100%;

margin: 0;

border-collapse: collapse;

}

#page\_nav

{

width: 100px;

height: 100%;

vertical-align: top;

}

#page\_main

{

vertical-align: top;

height: 100%;

padding: 0;

}

Интерфейс системы можно увидеть на рисунке 3.

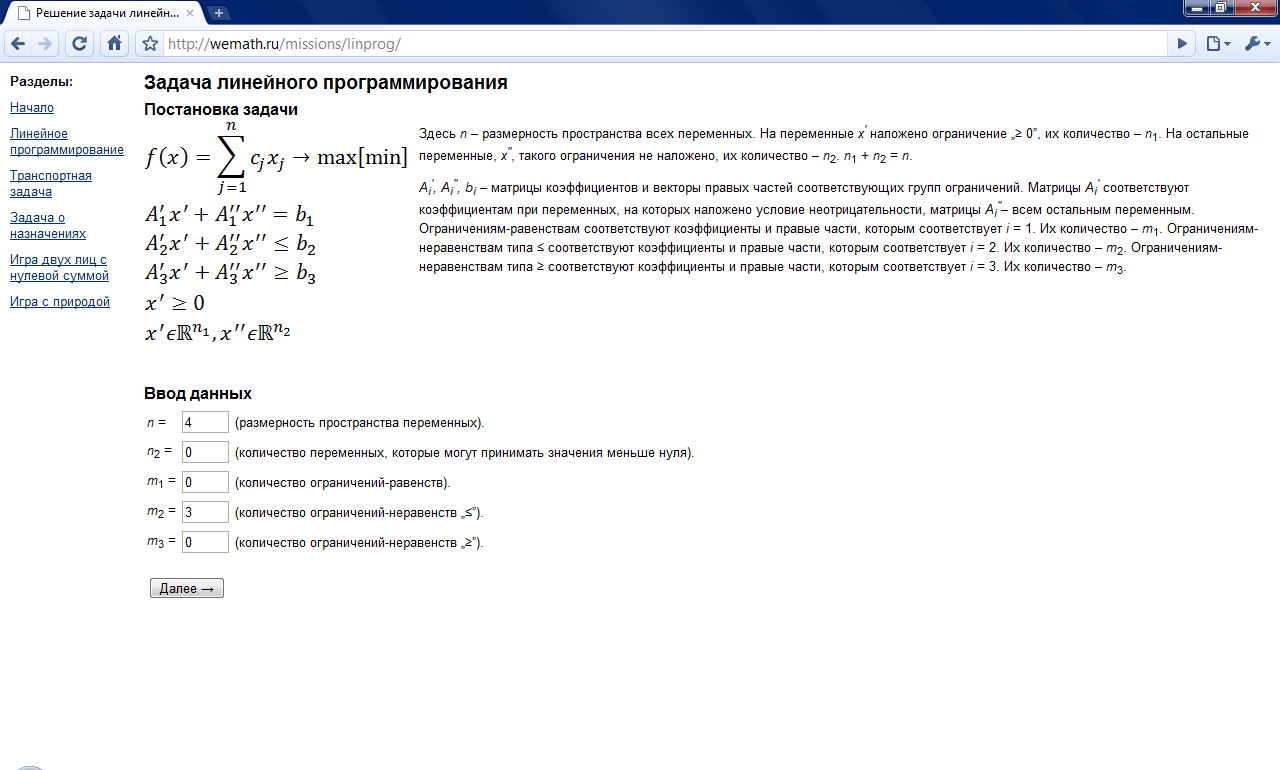


Рисунок 3. – Интерфейс системы.

Рассмотрим более подробно интерфейс подсистемы непосредственно решения задач.

## 2.4 Интерфейс подсистемы решения задач

### 2.4.1 Общее описание интерфейса подсистемы решения задач

Файлы, отвечающие за интерфейс решения задач, находятся в подкаталогах каталога missions.

В данный момент там находятся подкаталоги:

* linprog – задача линейного программирования.
* transport – транспортная задача.
* naznach – задача о назначениях.
* game\_of\_two – задача решения игры двух лиц с нулевой суммой.
* game\_with\_nat – задача решения игры с природой.

В каждом из этих подкаталогов находится файл index.html, который является основой интерфейсной части.

Рассмотрим по порядку интерфейсы решения каждой из задач.

### 2.4.2 Задача линейного программирования

Интерфейс решения задачи линейного программирования имеет три состояния: ввод размерностей, ввод численных данных и параметров (показать задачу в канонической форме, решить двойственную задачу, провести анализ задачи на чувствительность), вывод результатов расчетов и другой информации. Так же в третьем состоянии является возможным провести исследование задачи на чувствительность к изменению вектора правых частей системы ограничений. Рассмотрим каждое из этих состояний.

1. Ввод размерностей.

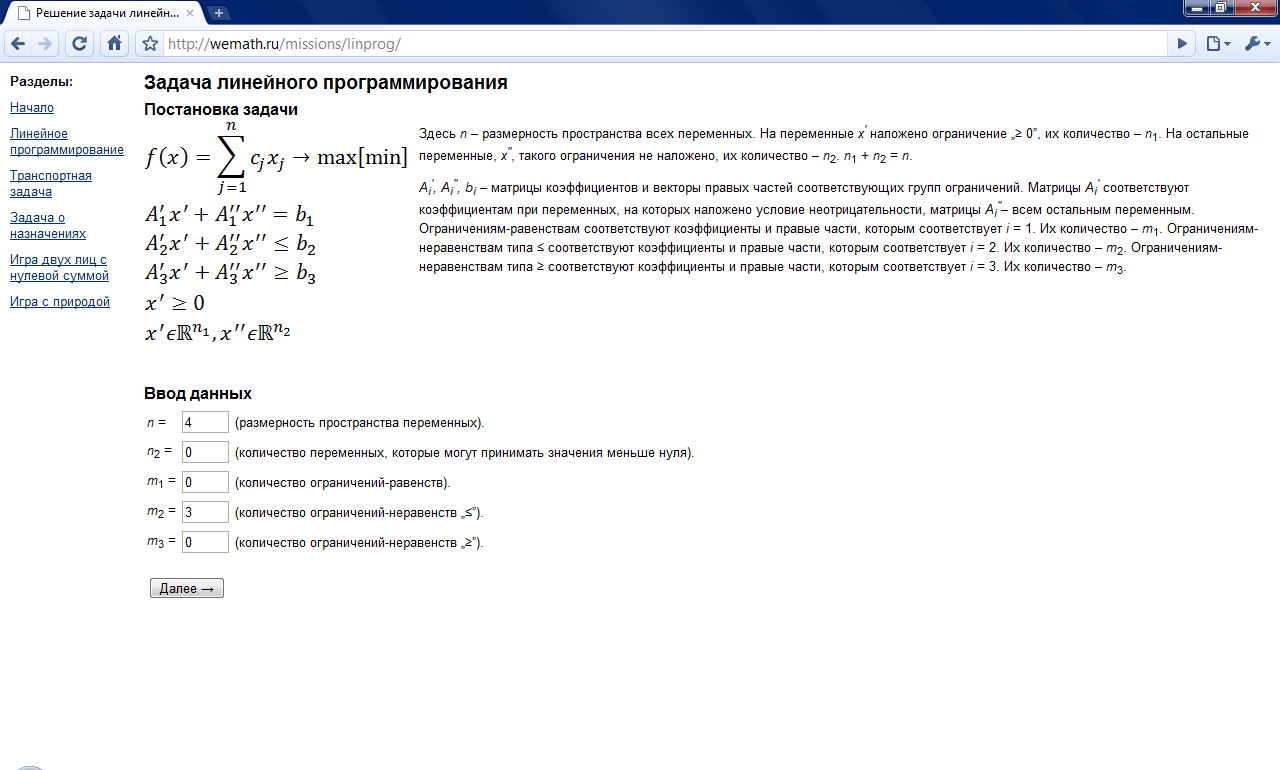


Рисунок 4. – Линейное программирование, ввод размерностей.

2. Ввод численных данных и параметров.

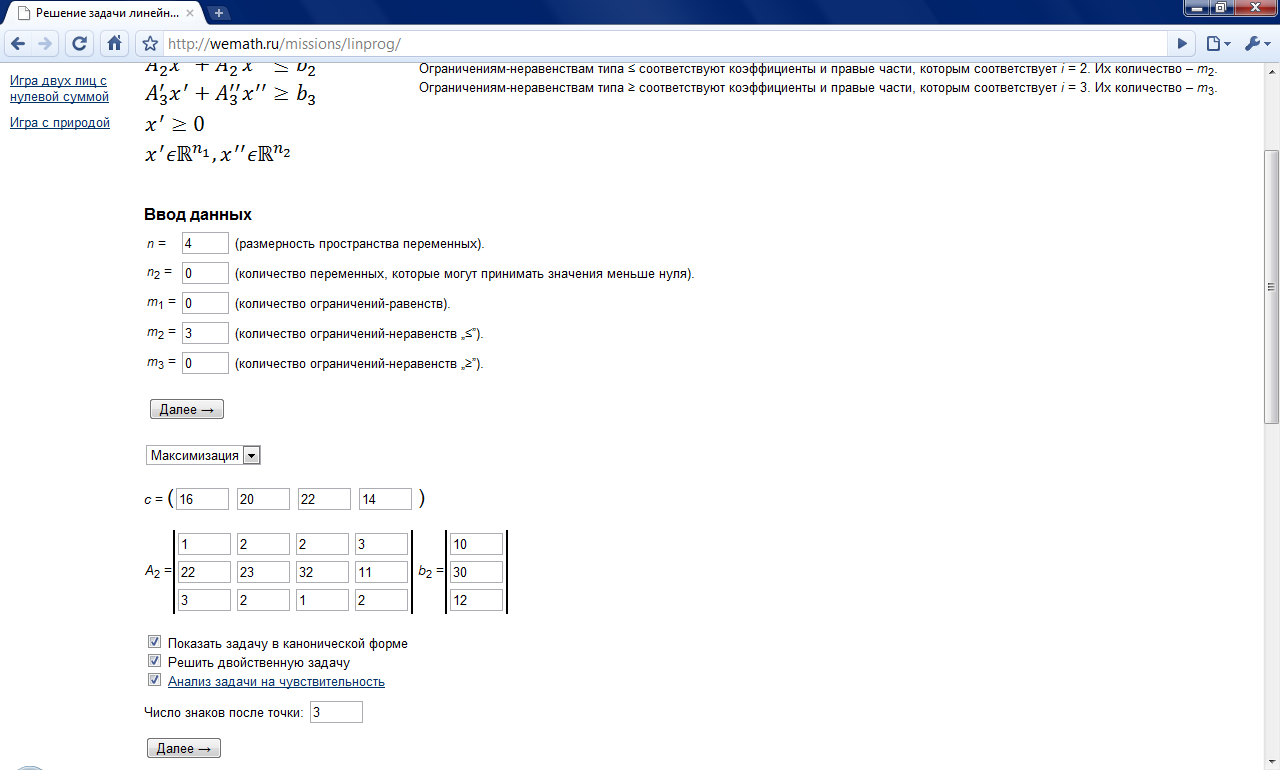


Рисунок 5. – Линейное программирование, ввод численных данных и параметров.

3. Вывод результатов расчетов и другой информации. Так же в данном состоянии возможно исследование задачи на чувствительность к изменению вектора правых частей.

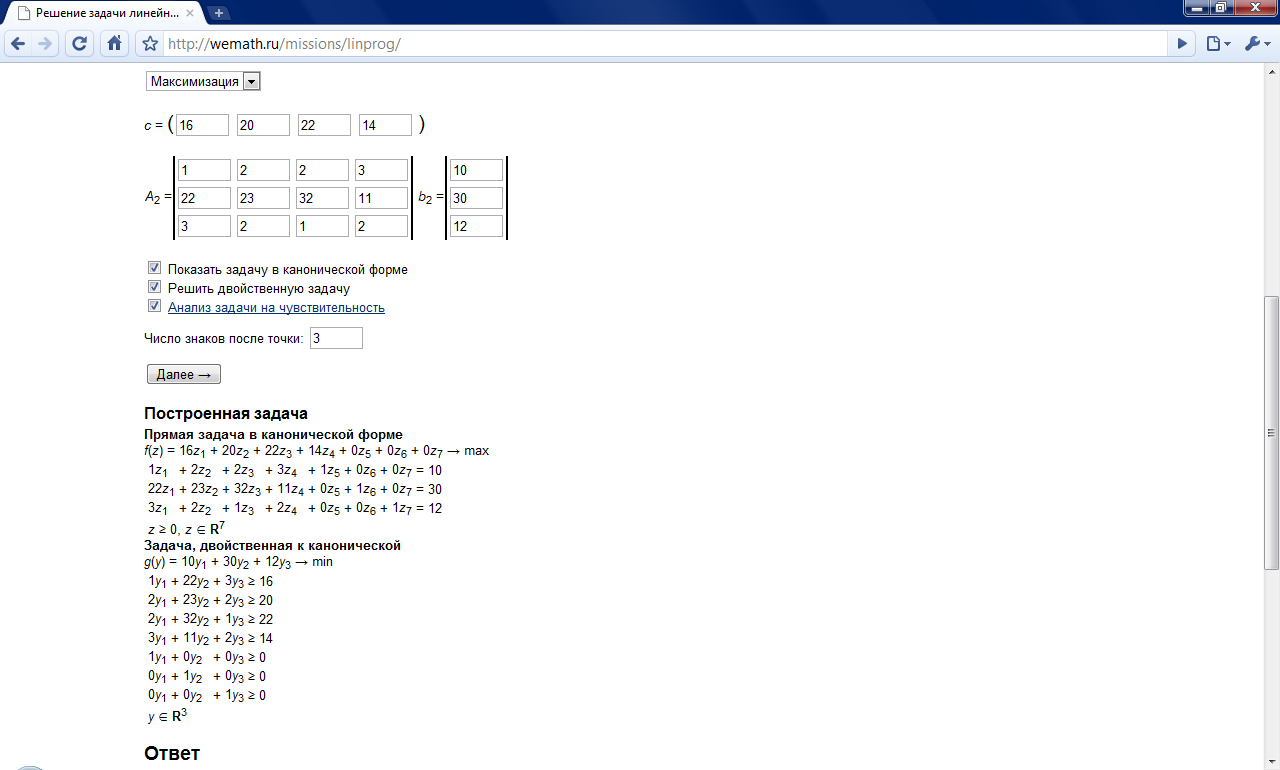


Рисунок 6. – Линейное программирование, вывод данных.

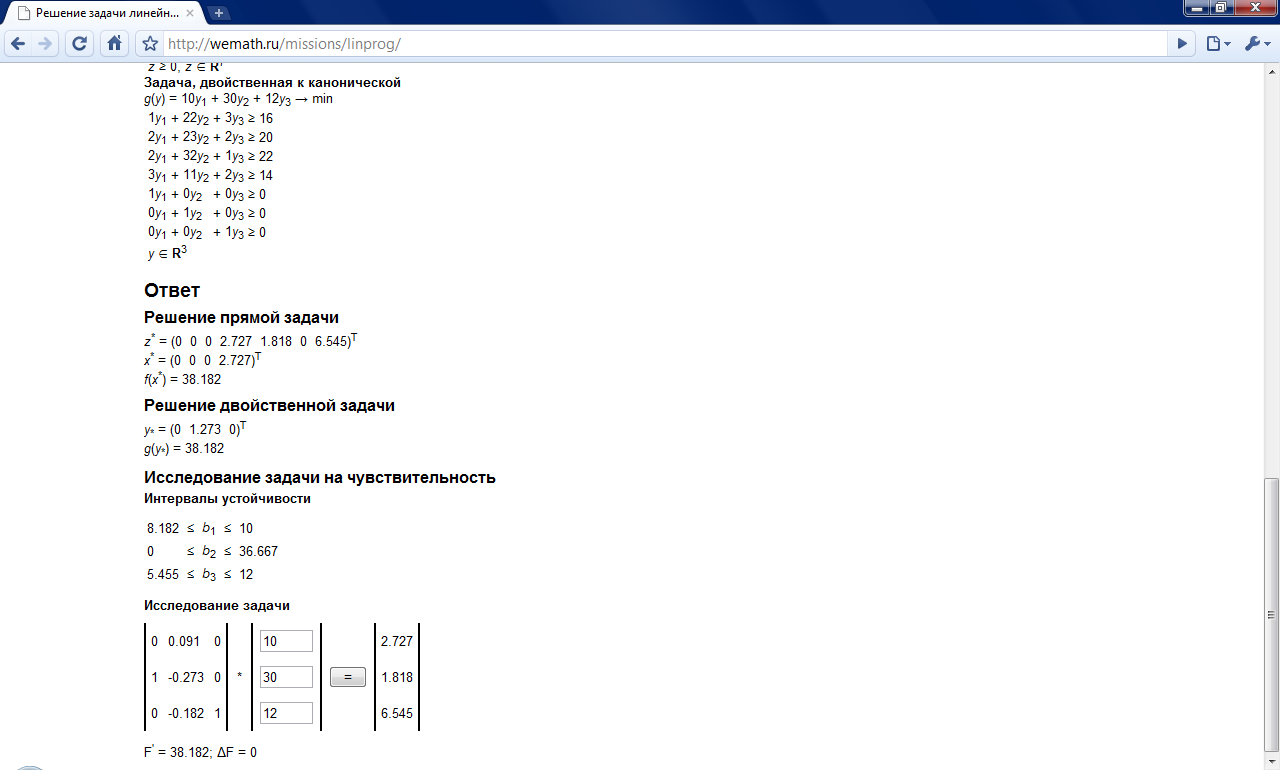


Рисунок 7. – Линейное программирование, вывод данных, продолжение.

Добавление на страницу новых форм производится с помощью сценариев на языке JavaScript. Решение задачи производится с помощью CGI-приложения на сервере, запрос к которому и получение результата от которого осуществляется в фоновом режиме, без перезагрузки страницы, с помощью сценария на языке JavaScript.

Рассмотрим участок кода, который реализует составление XML для запроса на сервер и сам запрос.

**//Создание XML для отправки на сервер**

XML = "<"+"?xml version='1' encoding='utf-8'?"+">\n";

XML += "<linprog>\n";

XML += " <m>"+m+"</m>\n";

XML += " <n>"+nn+"</n>\n";

XML += " <eps>0.00001</eps>\n";

XML += " <findmin>"+(findmin ? '1' : '0')+"</findmin>\n";

XML += " <need\_dual>"+(need\_dual ? '1' : '0')+"</need\_dual>\n";

XML += " <need\_intervals>"+(need\_interv ? '1' : '0')+"</need\_intervals>\n";

XML += " <vect\_c>\n";

for(j=0; j<nn; j++)

{

XML += " <c j='"+(j+1)+"'>"+c[j]+"</c>\n";

}

XML += " </vect\_c>\n";

XML += " <mat\_A>\n";

for(i=0; i<m; i++)

{

for(j=0; j<nn; j++)

{

XML += " <a i='"+(i+1)+"' j='"+(j+1)+"'>"+A[i][j]+"</a>\n";

}

}

XML += " </mat\_A>\n";

XML += " <vect\_b>\n";

for(i=0; i<m; i++)

{

XML += " <b i='"+(i+1)+"'>"+b[i]+"</b>\n";

}

XML += " </vect\_b>\n";

XML += "</linprog>\n";

**//Отправка запроса**

POST = encodeURIComponent(XML);

document.getElementById('output').innerHTML = "<img src='/images/ajax-loader.gif' alt='Loading' width='66' height='66' style='display: block; margin: 0 auto;' />";

AjaxPOST('/cgi-bin/linprog.cgi', POST, 'Answer(ajax\_request.responseText)');

Функция AjaxPOST(action, post, code) реализует запрос к адресу «action» с параметрами «post», а при получении ответа от сервера выполняется код, переданный в виде текста через переменную «code». В данном случае выполняется функция Answer(responceText), которая обрабатывает ответ сервера, который, как и запрос, структурирован в XML и выводит данные на страницу.

### 2.4.3 Транспортная задача, задача о назначениях

Интерфейс решения транспортной задачи имеет три состояния: 1. ввод размерностей (количество поставщиков, количество потребителей), 2. ввод векторов запасов и потребностей, а так же матрицы стоимостей перевозок, 3. вывод оптимального плана перевозок.

Рассмотрим каждое из них.

1. Ввод размерностей.

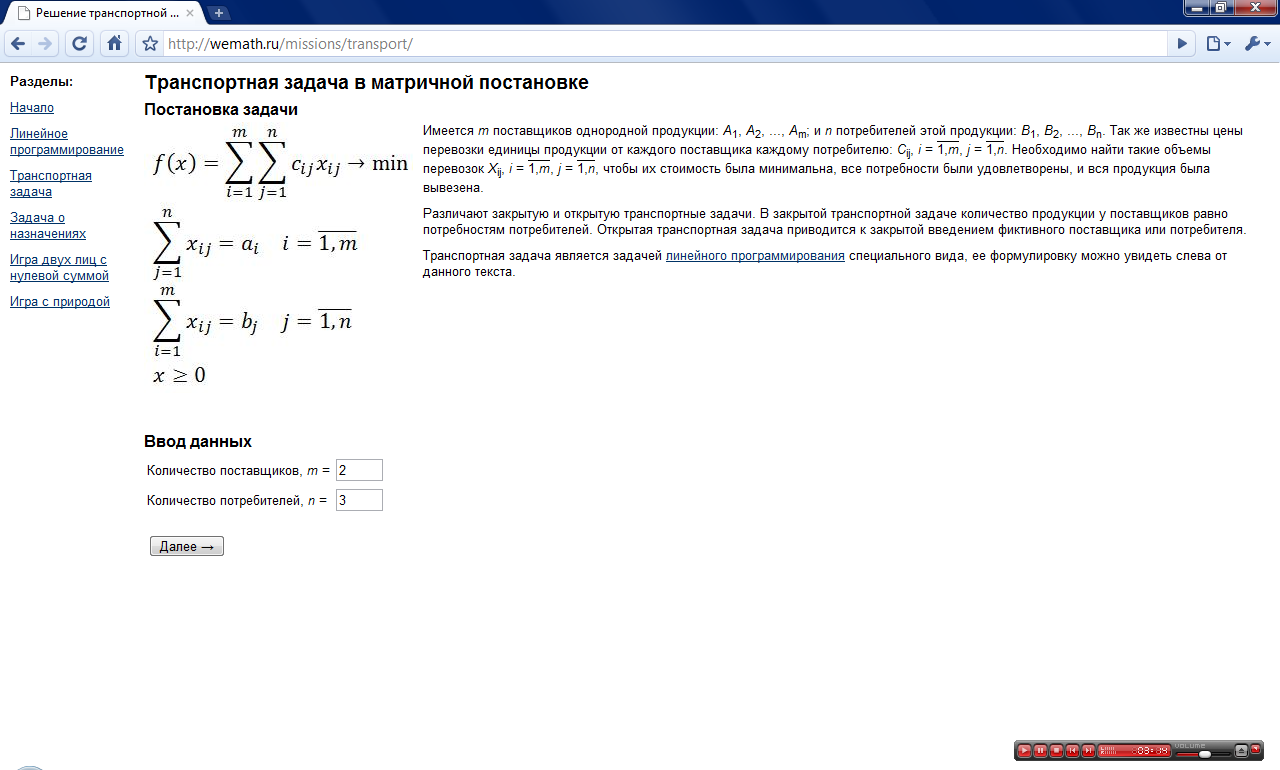


Рисунок 8. – Транспортная задача, ввод размерностей.

2. Ввод векторов запасов и потребностей, а так же матрицы стоимостей перевозок.

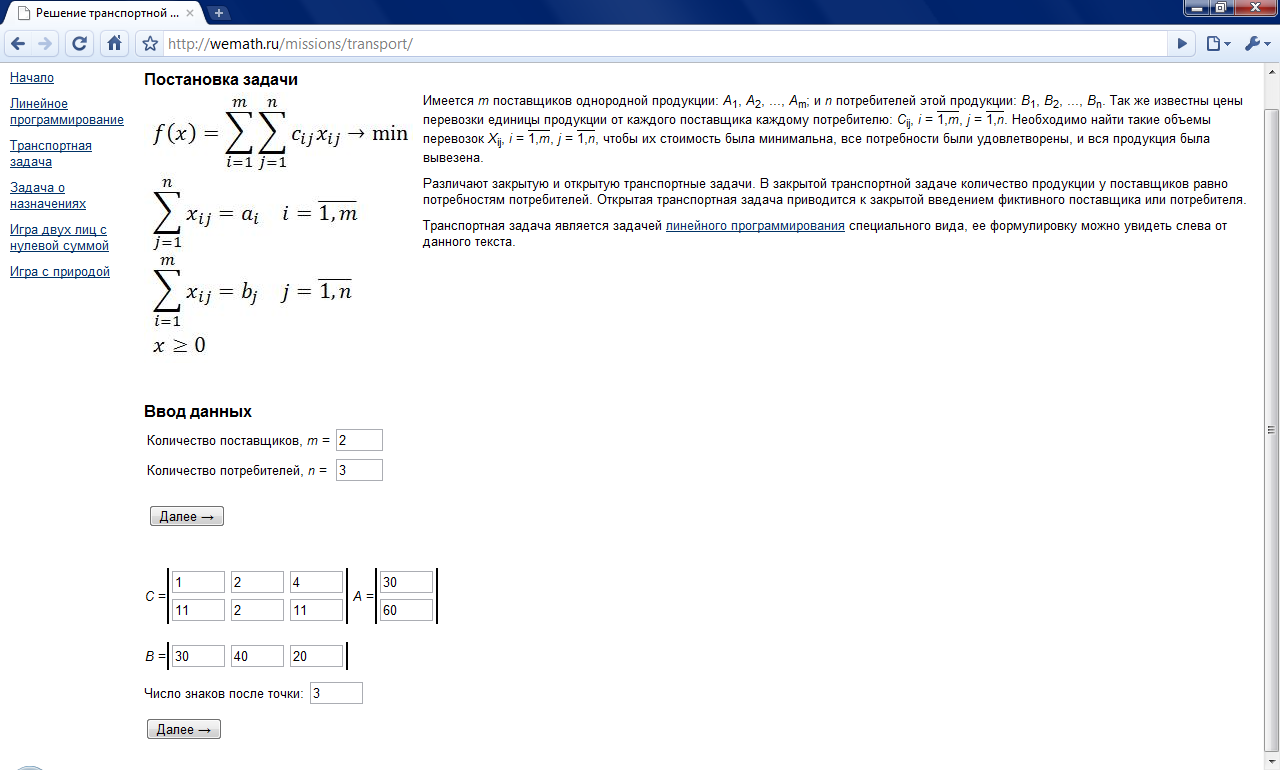


Рисунок 9. – Транспортная задача, ввод исходных данных.

3. Вывод оптимального плана перевозок.

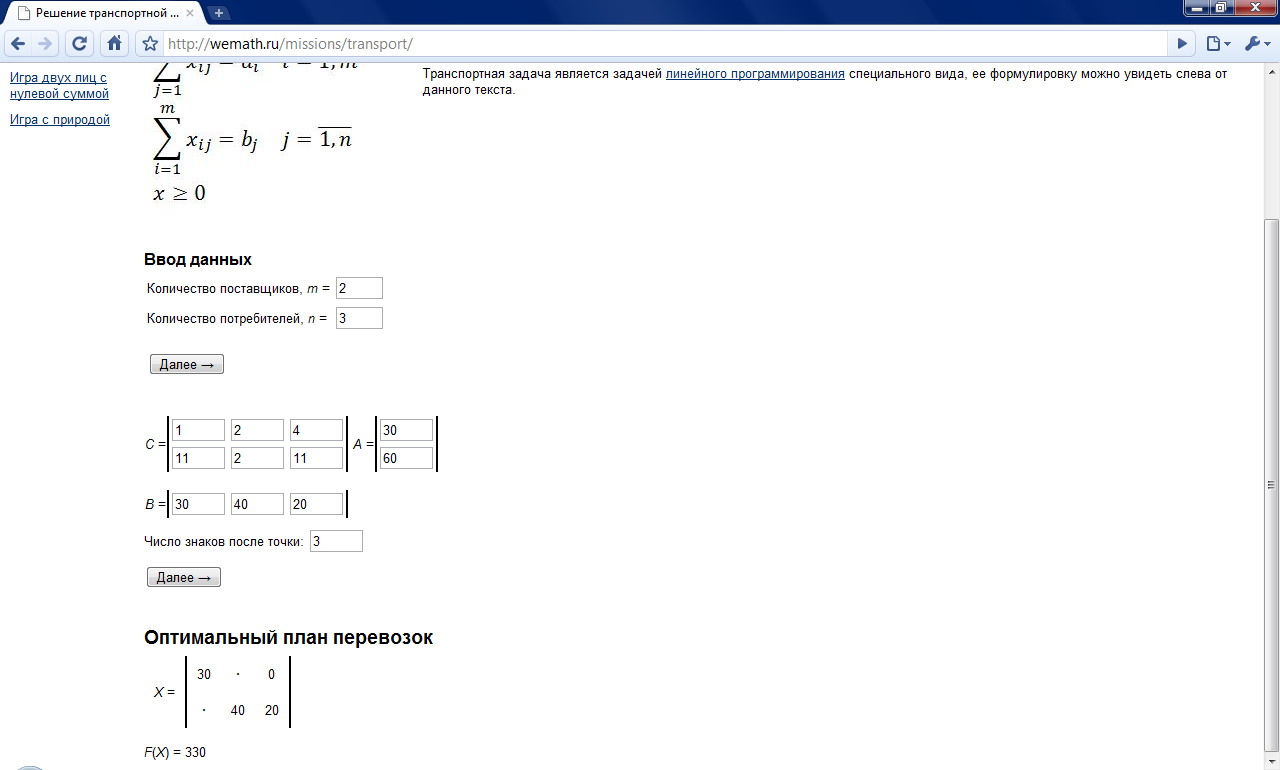


Рисунок 10. – Транспортная задача, вывод результатов.

Добавление новых форм на страницу производится с помощью сценариев на языке JavaScript, запрос на сервер отправляется аналогично случаю предыдущей задачи. Поскольку транспортная задача является частным случаем задачи линейного программирования, запрос производится к тому же CGI-приложению, что и в случае общей задачи линейного программирования, запрос и ответ сервера при этом структурируются аналогичным образом с помощью XML.

Интерфейс задачи о назначениях построен по тому же принципу, что и интерфейс транспортной задачи. Это объясняется тем, что задача о назначениях является частным случаем транспортной. Запрос на сервер организован так же аналогично транспортной задаче.

Интерфейс имеет три состояния: 1. ввод количества работ, 2. ввод матрицы затрат, 3. вывод оптимального плана назначений. Рассмотрим каждое из состояний.

1. Ввод количества работ.

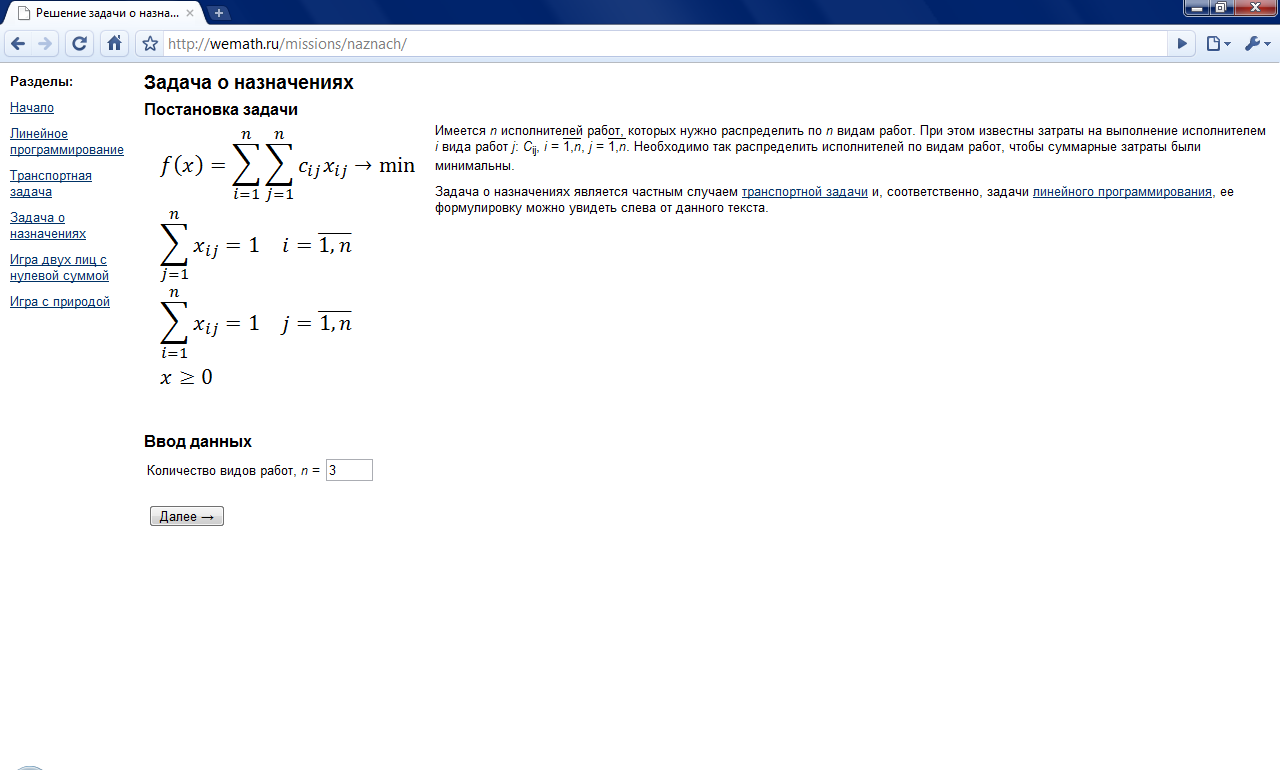


Рисунок 11. – Задача о назначениях, ввод количества работ.

2. Ввод матрицы затрат.

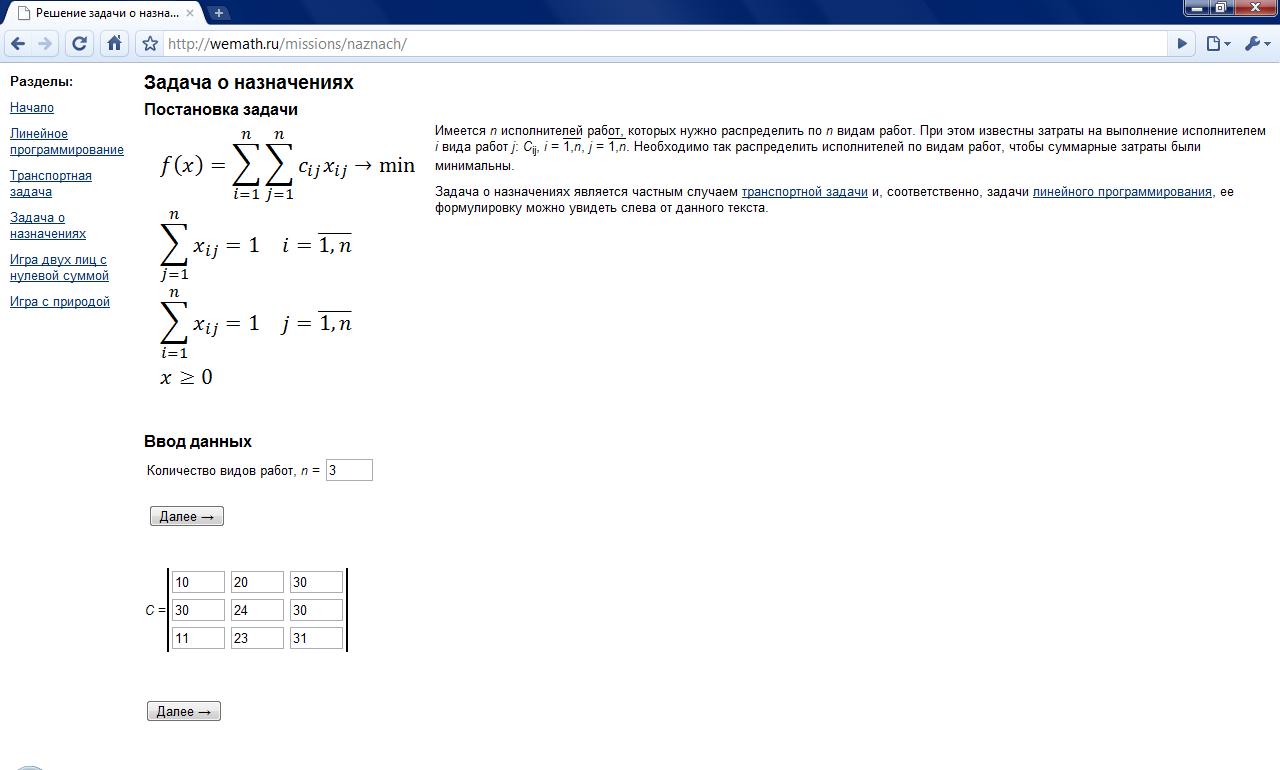


Рисунок 12. – Задача о назначениях, ввод матрицы затрат.

3. Вывод оптимального плана назначений.

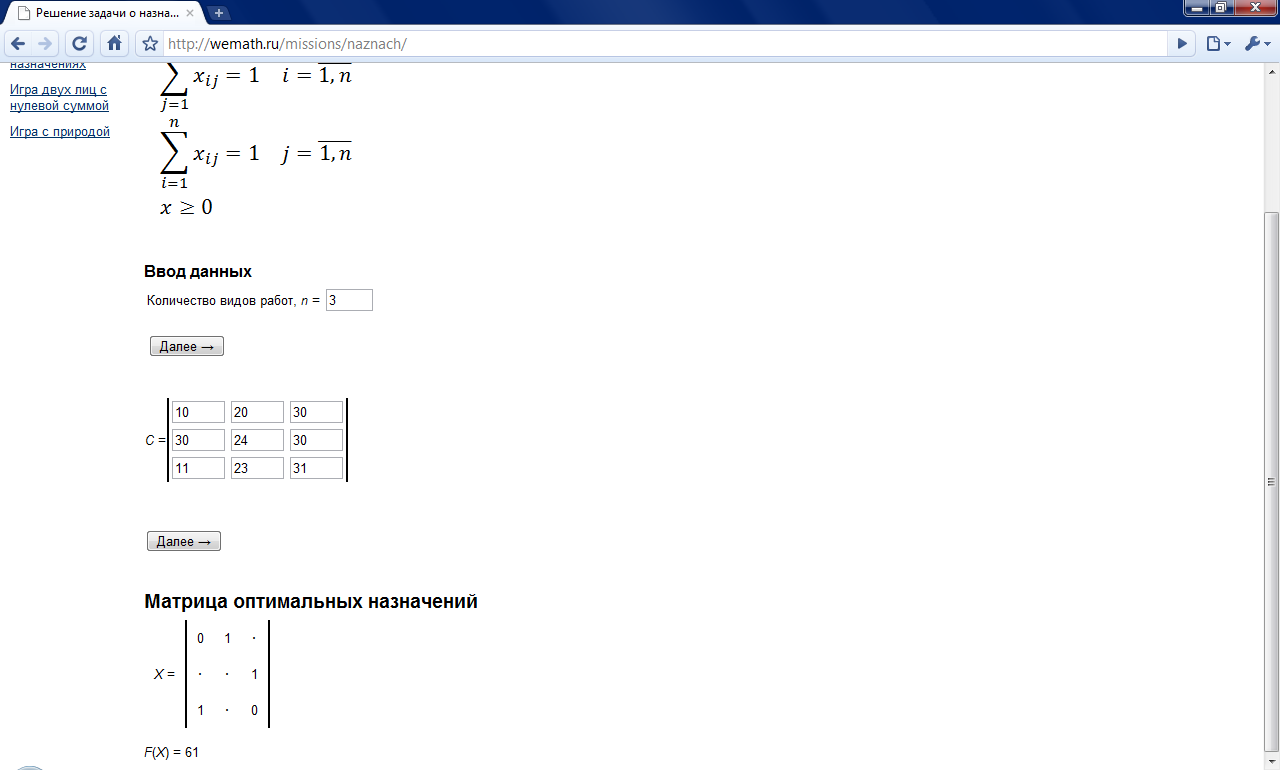


Рисунок 13. – Задача о назначениях, вывод оптимального плана назначений.

### 2.4.4 Задача решения игры двух лиц с нулевой суммой

Интерфейс решения задачи игры двух лиц с нулевой суммой имеет три состояния: ввод размерностей, ввод платежной матрицы, вывод результатов расчетов. Рассмотрим каждое из них.

1. Ввод размерностей. Здесь указываются количества стратегий для каждого из игроков.

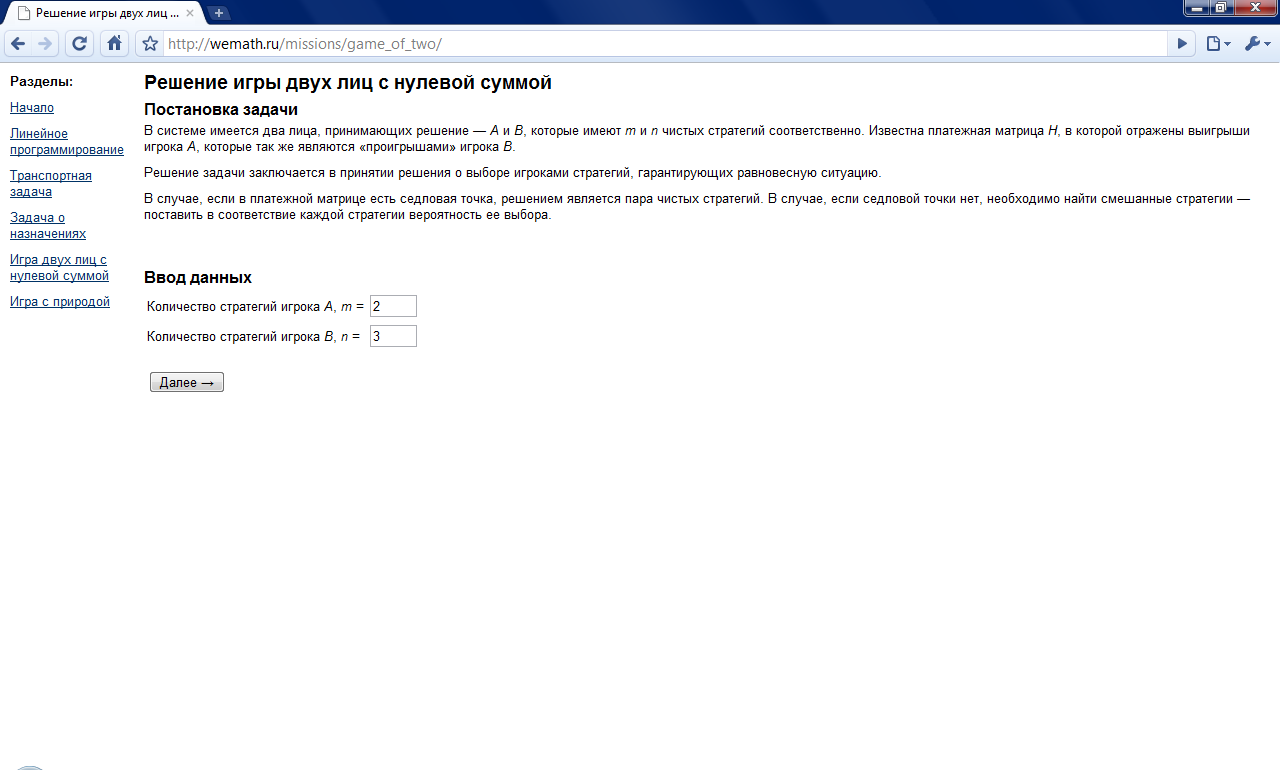


Рисунок 14. – Игра двух лиц, ввод размерностей.

2. Ввод платежной матрицы.

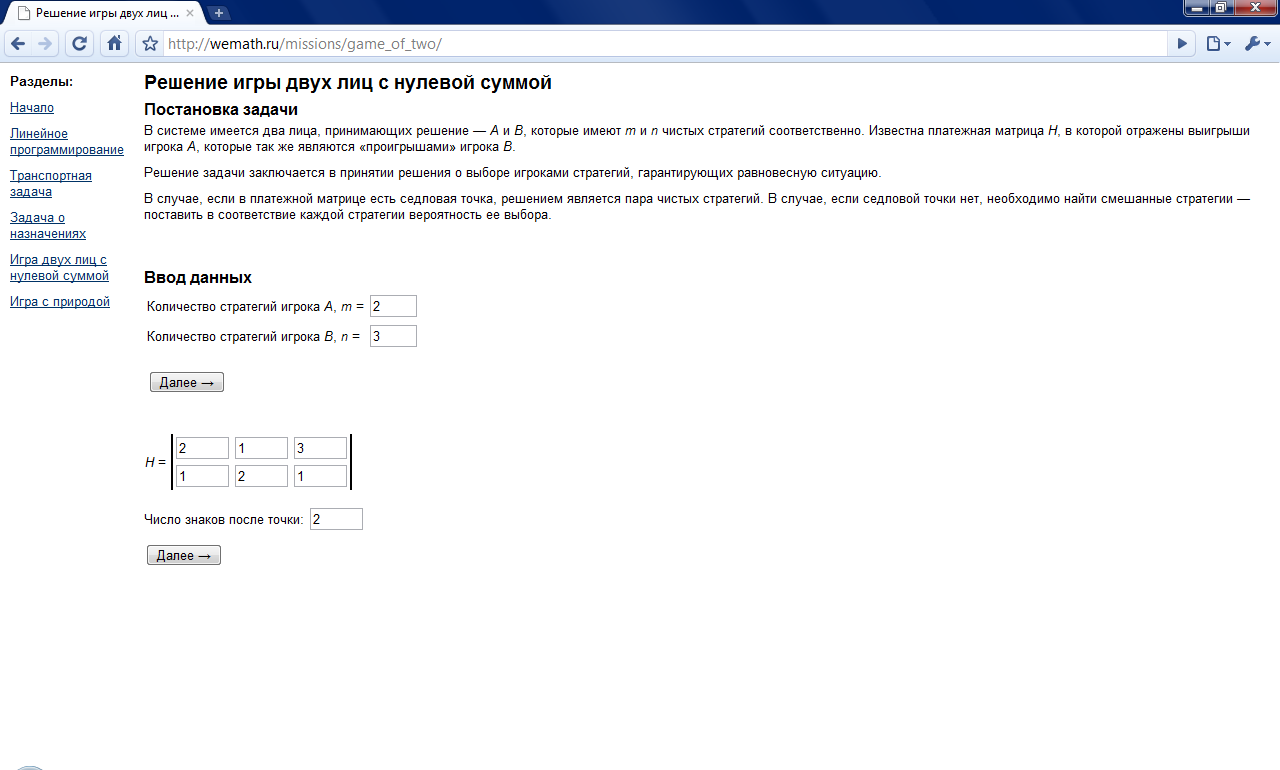


Рисунок 15. – Игра двух лиц, ввод платежной матрицы.

3. Вывод результатов расчетов.

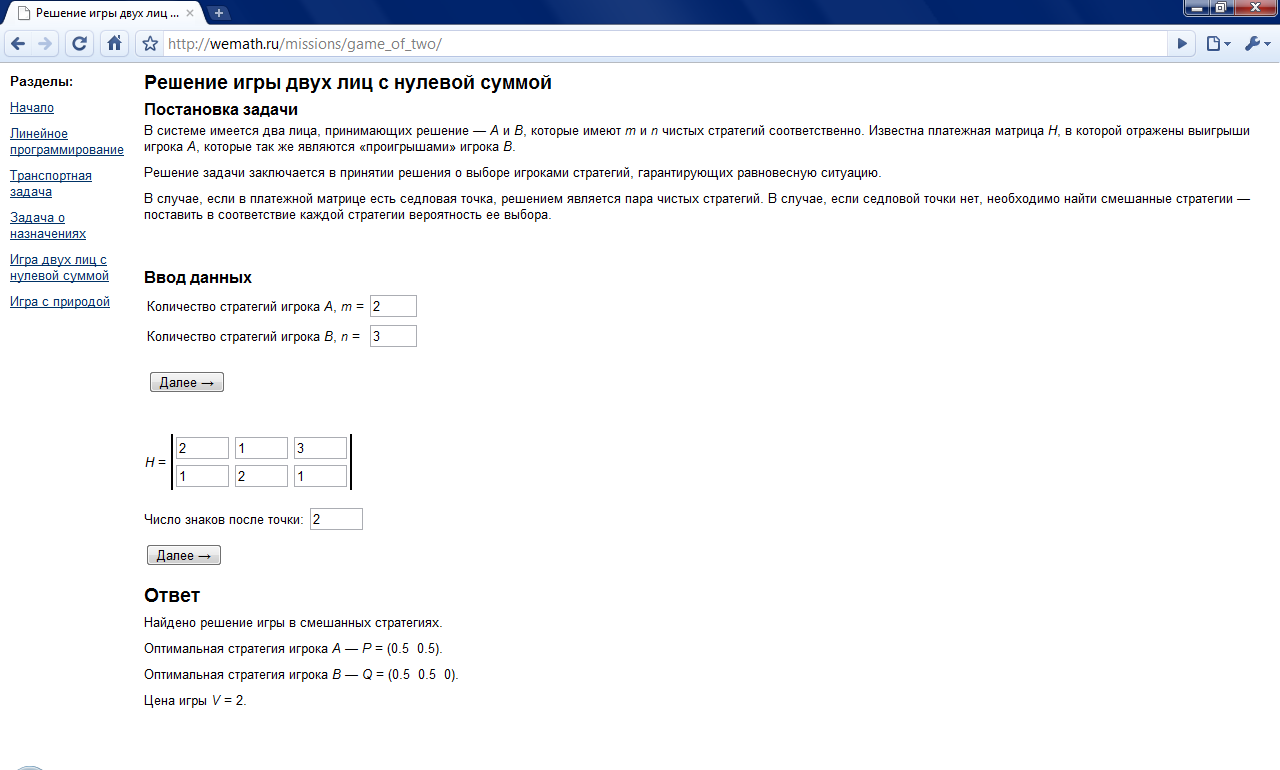


Рисунок 16. – Игра двух лиц, вывод результатов.

Видно, что содержимое web-страницы изменяется при переходах между состояниями. Это достигается посредством сценариев на языке JavaScript.

Для примера рассмотрим код формы ввода размерностей:

<div id="input1">

<table>

<tr>

<td>

Количество&nbsp;стратегий&nbsp;игрока&nbsp;<i>A</i>,&nbsp;<i>m</i>&nbsp;=

</td>

<td>

<input type="text" id="inp\_m" size="2" maxlength="2" value="2" />

</td>

</tr>

<tr>

<td>

Количество&nbsp;стратегий&nbsp;игрока&nbsp;<i>B</i>,&nbsp;<i>n</i>&nbsp;=

</td>

<td>

<input type="text" id="inp\_n" size="2" maxlength="2" value="3" />

</td>

</tr>

<tr>

<td colspan="3">

<div>&nbsp;</div>

<input type="button" value="Далее &rarr;" onclick="SetInp2();" />

</td>

</tr>

</table>

</div>

Видно, что при нажатии на кнопку «Далее» вызывается функция «SetInp2», которая создает форму для ввода платежной матрицы. Рассмотрим код этой функции:

//Создать формы для ввода данных

function SetInp2()

{

//Ввод n

var inp\_n = document.getElementById('inp\_n');

n = parseInt(inp\_n.value.replace(/(^\s+)|(\s+$)/g, ""));

if(isNaN(n) || n < 1)

{

alert('n должно быть целым положительным числом, n > 0!');

n = 3;

inp\_n.value = 3

return;

}

else if (n > MaxS)

{

alert('Максимальное количество стратегий = '+MaxS+'!');

n = 3;

inp\_n.value = 3;

return;

}

//Ввод m

var inp\_m = document.getElementById('inp\_m');

m = parseInt(inp\_m.value.replace(/(^\s+)|(\s+$)/g, ""));

if(isNaN(m) || m < 0)

{

alert('m должно быть целым неотрицательным числом!');

m = 2;

inp\_m.value = 2;

return;

}

//Создание формы ввода данных

var input2 = document.getElementById('input2');

input2.innerHTML = '';

//форма ввода матрицы H

input2.innerHTML += "<div style='height: 0.5cm;'>&nbsp;</div>";

var h\_table = document.createElement('table');

h\_table.cellSpacing = '0';

//h\_table.border = '1';

var h\_tbody = document.createElement('tbody');

var h\_tr, h\_td, h\_elem, i, j;

for(i=1; i<=m; i++)

{

h\_tr = document.createElement('tr');

if(i == 1)

{

h\_td = document.createElement('td');

h\_td.rowSpan = m;

h\_td.style.verticalAlign = 'middle';

h\_td.innerHTML = "<i>H</i>&nbsp;=&nbsp;";

h\_tr.appendChild(h\_td);

}

for(j=1; j<=n; j++)

{

h\_td = document.createElement('td');

h\_elem = document.createElement('input');

h\_elem.type = 'text';

h\_elem.size = 3;

h\_elem.id = 'H('+i+','+j+')';

h\_td.appendChild(h\_elem);

if(j == 1)

{

h\_td.style.borderLeft = '2px solid black';

}

else if(j == n)

{

h\_td.style.borderRight = '2px solid black';

}

h\_tr.appendChild(h\_td);

}

h\_tbody.appendChild(h\_tr);

}

h\_table.appendChild(h\_tbody);

input2.appendChild(h\_table);

var div = document.createElement("div");

div.style.marginBottom = '10px';

div.innerHTML = "<br />Число&nbsp;знаков&nbsp;после&nbsp;точки:&nbsp;<input type='text' size='3' id='after\_point' value='2'><br />";

input2.appendChild(div);

input2.innerHTML += "<input type='button' value='Далее &rarr;' onclick='GameGo();' />";

input2.innerHTML += "<div style='height: 0.5cm;'>&nbsp;</div>";

}

Переход в состояние вывода результатов осуществляется с использованием трех функций: «GameGo», «Clean» и «Smooth». Функция «GameGo» осуществляет ввод платежной матрицы в массив. После этого вызывается функция «Clean», в которой осуществляется попытка решить игру в чистых стратегиях. В случае неудачи вызывается функция «Smooth» - решение игры в смешанных стратегиях. Решение игры в смешанных стратегиях осуществляется с помощью линейного программирования и, соответственно, осуществляется обращение к серверу, к CGI-приложению для решения задач линейного программирования.

### 2.4.5 Решение задачи игры с природой

Интерфейс задачи решения задачи игры с природой имеет три состояния: 1. ввод размерностей и выбор критериев принятия решения, 2. ввод матрицы выигрышей, 3. вывод результатов. Рассмотрим каждое из них:

1. Ввод размерностей и выбор критериев принятия решения.

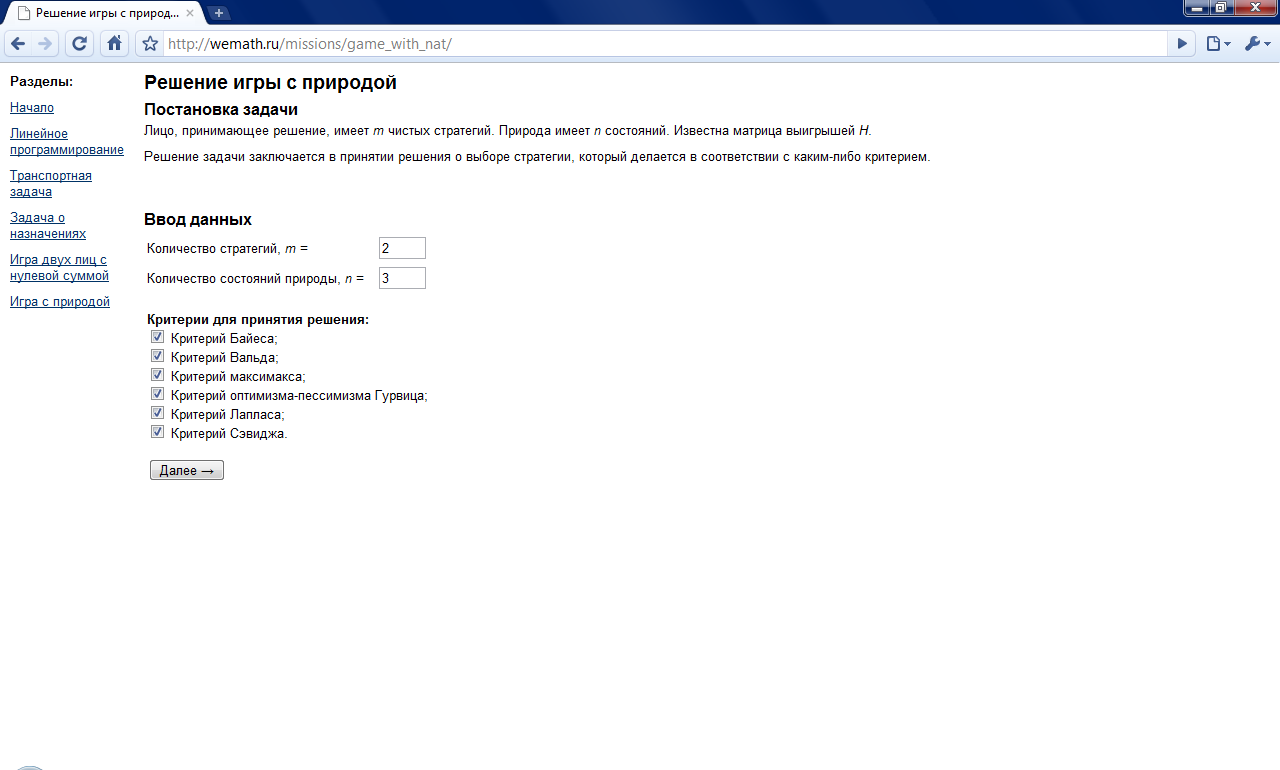


Рисунок 17. – Игра с природой, ввод размерностей.

2. Ввод матрицы выигрышей.

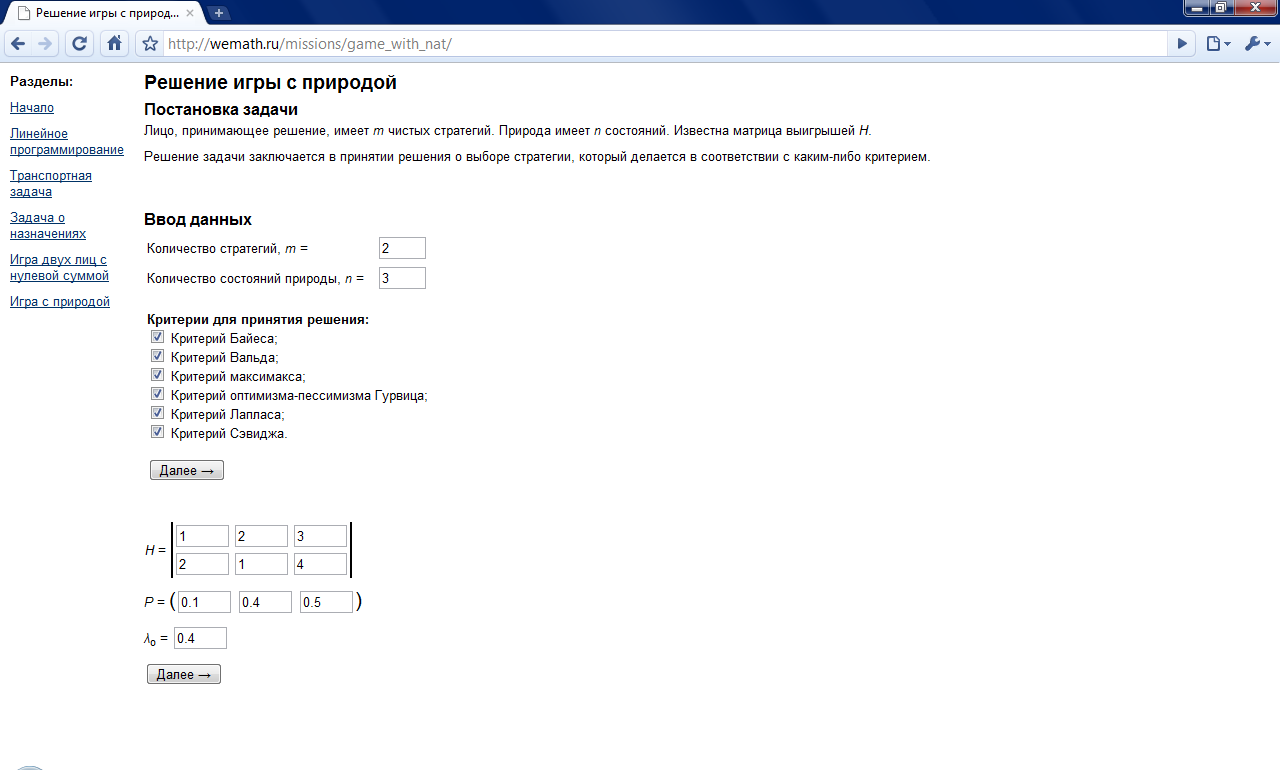


Рисунок 18. – Игра с природой, ввод матрицы выигрышей.

3. Вывод результатов.

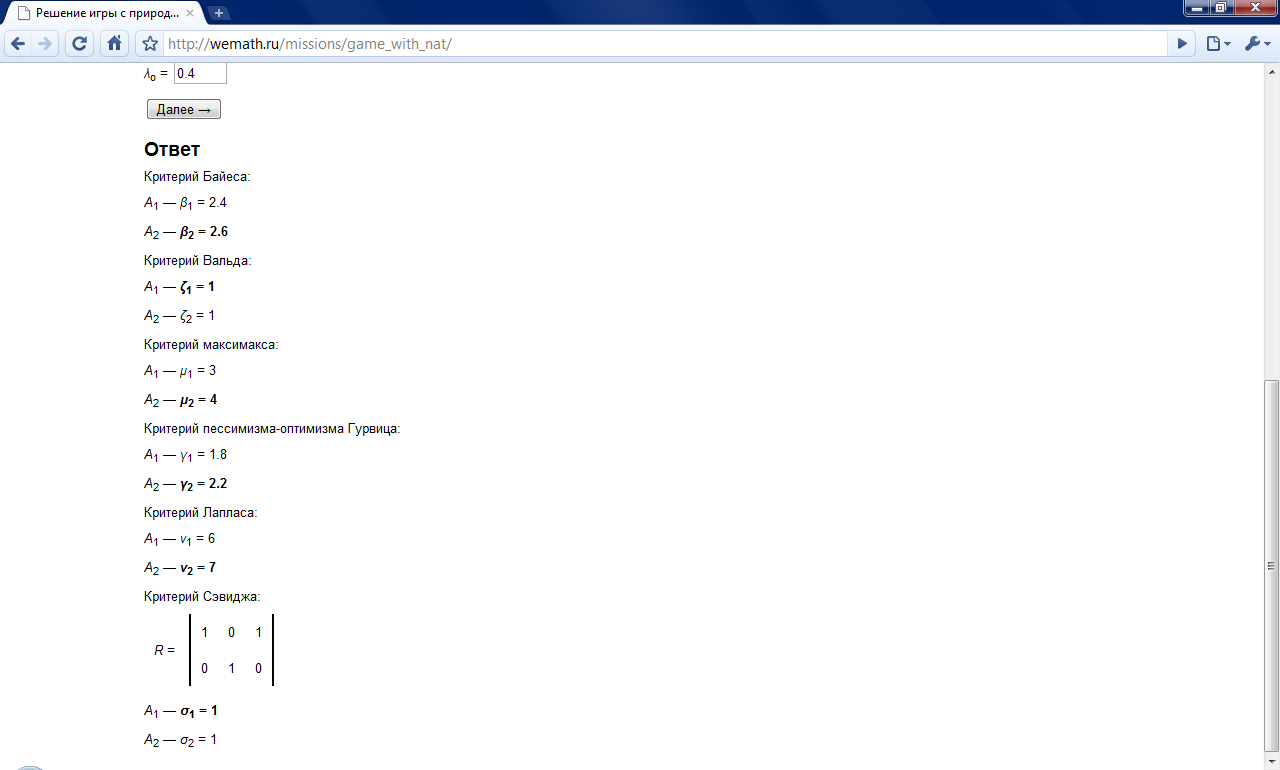


Рисунок 19. – Игра с природой, вывод результатов.

Решение данной задачи не требует математических методов, требующих большого объема вычислений, и при ее решении можно обойтись сценариями на языке JavaScript без обращений к серверу.

## 2.5. Описание приложения для решения задачи линейного программирования

### 2.5.1 Общее описание приложения

Все вышеперечисленные задачи используют для получения результата линейное программирование. Для решения задачи линейного программирования в разработанной системе используется CGI-приложение linprog.cgi.

Приложение решает задачу максимизации целевой функции при ограничениях-равенствах и при условии неотрицательности всех переменных:

Приложение создано с помощью совместной компиляции кода на языках программирования C++ и Fortran.

Непосредственно для вычислений был выбран язык Fortran (диалект Fortran 95), но поскольку существует дополнительная задача разбора XML, необходимо было также привлечь C++ для решения этой задачи, так как в языке Fortran недостаточно средств для этого.

При вычислениях были использованы функции из библиотеки Lapack, которые, в свою очередь, используют функции библиотеки Atlas. Используются следующие функции:

* DSYEV – собственные числа симметричной матрицы, двойная точность. Данная функция необходима для определения ранга матрицы *A* системы ограничений.
* DGESV – решение системы линейных уравнений, двойная точность. Данная функция используется для нахождения решения двойственной задачи.
* DGETRF – LU-разложение матрицы, двойная точность. Данная функция используется для подготовки матрицы для нахождения обратной к ней.
* DGETRI – нахождение обратной матрицы, двойная точность. В качестве аргумента принимает результат LU-разложения матрицы. Данная функция необходима для анализа задачи на чувствительность к изменению вектора правых частей.

### 2.5.2 Блок-схема приложения «Линейное программирование»

Работу данного приложения можно представить в виде следующей схемы:

Да

Нет

Ввод данных (матрица A, векторы b и c) в формате XML

Разбор XML с помощью функций C++: запись матрицы *A*, векторов *b* и *c* в массивы.

Ранг матрицы *A* равен числу ее строк?

Начало

Вывод сообщения об ошибке в формате XML

1

Решение задачи ЛП симплекс-методом, результаты: *X* – вектор решения, *BP* – массив номеров базисных переменных.

2

Да

Нет

Да

Нет

Нужно решение двойственной задачи?

2

3

Составление матрицы *W*, столбцы которой соответствуют столбцам матрицы A, соответствующим коэффициентам при базисных переменных.

Нахождение двойственных оценок *y* как корней системы уравнений

Нужен анализ задачи на чувствительность?

3

Составление матрицы *W*, столбцы которой соответствуют столбцам матрицы A, соответствующим коэффициентам при базисных переменных.

Нахождение матрицы ,

нахождение интервалов устойчивости.

Вывод данных в формате XML

3

2

Конец

Рисунок 20. – Блок-схема приложения «Линейное программирование».

### 2.5.3 XML для ввода и вывода данных

В приложение вводятся следующие данные: матрица коэффициентов при переменных *A*, вектор правых частей *b*, вектор *c* коэффициентов целевой функции. Структура XML для передачи этих данных следующая:

*<?xml version=”1.0” encoding=”utf-8”?>*

**<linprog>**

**<m>**«количество строк в матрице *A*»**</m>**

**<n>**«количество переменных в задаче»**</n>**

**<eps>**«точность вычислений»**</eps>**

**<findmin>**«1 – нахождение минимума, 0 – нахождение максимума»**</findmin>**

**<need\_dual>**«1 – нужно решение двойственной задачи, 0 – не нужно»**</need\_dual>**

**<need\_intervals>**«1 – нужно найти интервалы устойчивости, 0 – не нужно»**</need\_intervals>**

**<vect\_c>**

**<c j='1'>**«элемент 1 вектора *c*»**</c>**

**<c j='2'>**«элемент 2 вектора *c*»**</c>**

**«…»**

**<c j='*n*'>**«элемент *n* вектора *c*»**</c>**

**</vect\_c>**

**<mat\_A>**

**<a i='1' j='1'>**«элемент 1,1 матрицы *A*»**</a>**

**<a i='1' j='2'>**«элемент 1,2 матрицы *A*»**</a>**

**«…»**

**<a i='*m*' j='*n*'>**«элемент *m,n* матрицы *A*»**</a>**

**</mat\_A>**

**<vect\_b>**

**<b i='1'>**«элемент 1 вектора *b*»**</b>**

**<b i='2'>**«элемент 2 вектора *b*»**</b>**

**«…»**

**<b i='*m*'>**«элемент *m* вектора *b*»**</b>**

**</vect\_b>**

**</linprog>**

Данные для вывода следующие: *x\** – вектор решения прямой задачи, *bp* – массив номеров базисных переменных, значение целевой функции прямой задачи *f(x\*)*, *y\** – вектор решения двойственной задачи, значение целевой функции двойственной задачи *g(y\*)*, матрица *D*, интервалы устойчивости и . Структура XML для передачи этих данных следующая:

*<?xml version=”1.0” encoding=”utf-8”?>*

**<linprog\_answer>**

**<primal>**

**<BP>**

«список номеров базисных переменных через пробел»

**</BP>**

**<X>**

«элементы вектора *x*\* через пробел»

**</X>**

**<F>**

«значение целевой функции *f(x\*)*»

**</F>**

**</primal>**

**<dual>**

**<Y>**

«элементы вектора *y\** через пробел»

**</Y>**

**<G>**

«значение целевой функции *g(y\*)*»

**</G>**

**</dual>**

**<intervals>**

**<i>**

1

**<bm>**«значение »**</bm>**

**<bp>**«значение »**</bp>**

**</i>**

**<i>**

2

**<bm>**«значение »**</bm>**

**<bp>**«значение »**</bp>**

**</i>**

**«…»**

**<i>**

*m*

**<bm>**«значение »**</bm>**

**<bp>**«значение »**</bp>**

**</i>**

**<D>**

*<![CDATA[*

«матрица *D*, элементы строк которой разделены пробелами, а строки – переносом строки»

*]]>*

**</D>**

**</intervals>**

**</linprog\_answer>**

Помимо вывода данных о результате вычислений существуют и сообщения об ошибках:

* сообщение об ошибке разбора XML,
* сообщение об ошибке решения прямой задачи,
* сообщение об ошибке решения двойственной задачи,
* сообщение об ошибке вычисления интервалов устойчивости.

XML с сообщением об ошибке выглядит следующим образом:

*<?xml version=”1.0” encoding=”utf-8”?>*

**<linprog\_answer>**

**<error mission=”**«код этапа решения задачи»**”>**

«код ошибки»

**</error>**

**</linprog\_answer>**

### 2.5.6 Программный код приложения

Программный код приложения содержится в следующих файлах:

* linprog.for – функции для решения задачи линейного программирования: решение прямой задачи, решение двойственной задачи, нахождение интервалов учтойчивости.
* cpp\_functions.cpp – функции на языке C++ для получение данных из XML.
* cpp\_functions.h – заголовочный файл для cpp\_functions.cpp.
* linprog\_cgi.for – основной код CGI-приложения.

Для совместной компиляции кода на C++ и Fortran предназначен сценарий make.sh. Рассмотрим его код:

*#!/bin/bash*

g++ cpp\_functions.cpp –c

gfortran cpp\_functions.o linprog.for linprog\_cgi.for -o /var/www/wemath.ru/cgi-bin/linprog.cgi -ffree-form -ffree-line-length-none -lstdc++ -llapack

rm cpp\_functions.o

Рассмотрим так же основной код CGI-приложения:

PROGRAM LinProgCGI

IMPLICIT NONE

EXTERNAL GET\_XML, PARSE\_XML\_MN, PARSE\_XML\_ABC, Simplex, Dual, Interv\_of\_stab

**!Переменные для решения задачи**

«…»

**!HTTP-заголовки**

print "(A)", "Content-Type: text/xml; charset=utf-8"//CHAR(10)

**!Получение данных**

CALL GETENV("CONTENT\_LENGTH", CL0)

READ(CL0, \*) CL

ALLOCATE( CONT(CL+1) )

CALL GET\_DATA(CONT, CL)

**!Преобразование данных**

ALLOCATE( XML(CL+1) )

CALL GET\_XML(CONT, CL, XML)

DEALLOCATE( CONT )

CALL PARSE\_XML\_MN(XML, M, N, NEED\_DUAL, NEED\_INTERV, eps)

ALLOCATE( A0(M\*N) )

ALLOCATE( b(M) )

ALLOCATE( c(N) )

CALL PARSE\_XML\_ABC(XML, M, N, A0, b, c, INFO2)

DEALLOCATE( XML )

IF (INFO2 .EQ. 0) THEN

ALLOCATE( A(m,n) )

k = 1

DO i=1,m

DO j=1,n

A(i,j) = A0(k)

k = k + 1

END DO

END DO

END IF

DEALLOCATE( A0 )

**!Заголовок XML**

print "(A)", "<?xml version='1.0' encoding='utf-8'?>"

print "(A)", "<linprog\_answer>"

IF (INFO2 .NE. 0) THEN

print "(A)", " <error mission='parse'>"

print "(I2)", INFO2

print "(A)", " </error>"

DEALLOCATE( b )

DEALLOCATE( c )

print "(A)", "</linprog\_answer>"

RETURN

END IF

**!Решение прямой задачи**

AlLOCATE( BP(m) )

ALLOCATE( X(n) )

CALL Simplex(A, b, c, m, n, eps, X, F, BP, INFO)

IF (INFO .EQ. 0) THEN

«вывод результатов вычислений в XML»

ELSE

«вывод сообщения об ошибке в XML»

END IF

**!Решение двойственной задачи**

IF (NEED\_DUAL) THEN

ALLOCATE( Y(m) )

CALL Dual(A, b, c, X, F, BP, m, n, eps, Y, G, INFO)

IF (INFO .EQ. 0) THEN

«вывод результатов вычислений в XML»

ELSE

«вывод сообщения об ошибке в XML»

END IF

DEALLOCATE( Y )

END IF

**!Расчет интервалов устойчивости**

IF (NEED\_INTERV) THEN

ALLOCATE( D(m,m) )

ALLOCATE( DB(m,2) )

CALL Interv\_of\_stab(A, b, X, BP, m, n, eps, D, DB, INFO)

IF (INFO .EQ. 0) THEN

«вывод результатов вычислений в XML»

ELSE

«вывод сообщения об ошибке в XML»

END IF

DEALLOCATE( D )

DEALLOCATE( DB )

END IF

**!Окончание вывода данных**

print "(A)", "</linprog\_answer>"

DEALLOCATE( A )

DEALLOCATE( b )

DEALLOCATE( c )

DEALLOCATE( X )

DEALLOCATE( BP )

END

SUBROUTINE GET\_DATA(CONT, CL)

IMPLICIT NONE

**!Параметры процедуры**

INTEGER :: CL !Длина POST-запроса

CHARACTER(CL) :: CONT0 !Строка POST-запроса

CHARACTER, DIMENSION(CL+1) :: CONT !C-строка запроса

**!Прочие переменные**

INTEGER :: i

**!Ввод данных**

READ(\*, "(A)") CONT0

**!Преобразование данных**

DO i=1,CL

CONT(i) = CONT0(i:i)

END DO

CONT(CL+1) = CHAR(0)

END

## 2.6 Решение экономико-математических задач с помощью разработанной системы

### 2.6.1 Задача оптимального планирования производства

Данная задача является задачей линейного программирования.

Пусть на некотором предприятии производится 4 вида продукции. Затраты ресурсов на производство единицы каждого из видов продукции приведены в таблице 1.

Таблица 1. Нормативные затраты.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Продукция 1 | Продукция 2 | Продукция 3 | Продукция 4 |
| Ресурс I | 4 | 5 | 2 | 3 |
| Ресурс II | 6 | 0 | 4 | 1 |
| Ресурс III | 0 | 7 | 6 | 5 |

Известна прибыль от продажи единицы каждого из видов продукции:

1. 36 у. е.
2. 30 у. е.
3. 16 у. е.
4. 12 у. е.

Запасы ресурсов при этом составляют:

1. 180,
2. 150,
3. 140.

Необходимо найти такие объемы выпуска каждого из видов продукции, которые обеспечат максимальную прибыль, при этом необходимо выполнить ограничения по ресурсам. Так же необходимо провести анализ чувствительности модели к изменению запасов ресурсов.

Построим экономико-математическую модель.

Оптимальный выпуск продукции найдем с помощью прямой задачи линейного программирования.

Переменные: *xi* – выпуск продукции вида *i*. .

Целевая функция, выражающая суммарную прибыль:

Ограничения:

Для определения влияния изменения запаса каждого из видов ресурсов на значение целевой функции необходимо найти двойственные оценки ресурсов.

Задача, двойственная к задаче определения оптимальных объемов производства выглядит следующим образом:

Решим данную задачу с помощью разработанной системы.

1. Введем размерности.

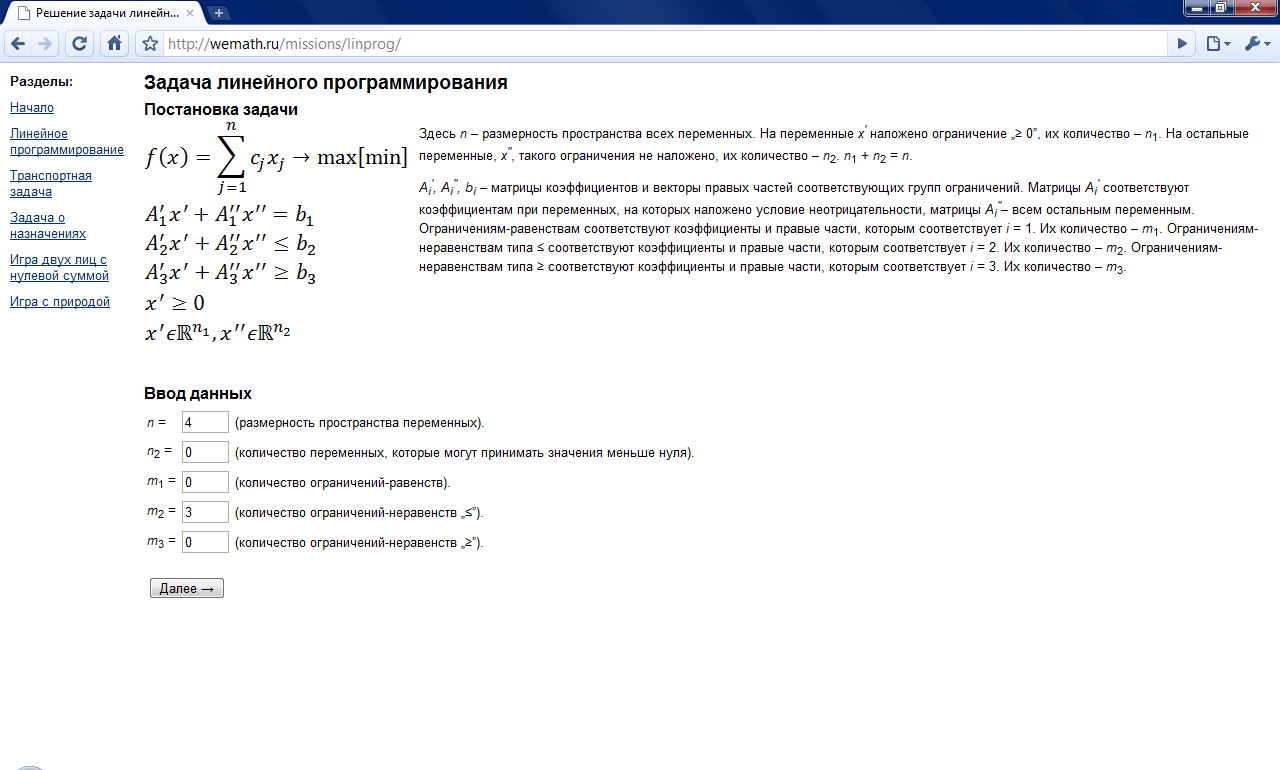


Рисунок 21. – Планирование производства, ввод размерностей.

2. Введем данные и выберем опции «Решить двойственную задачу» и «Анализ задачи на чувствительность».

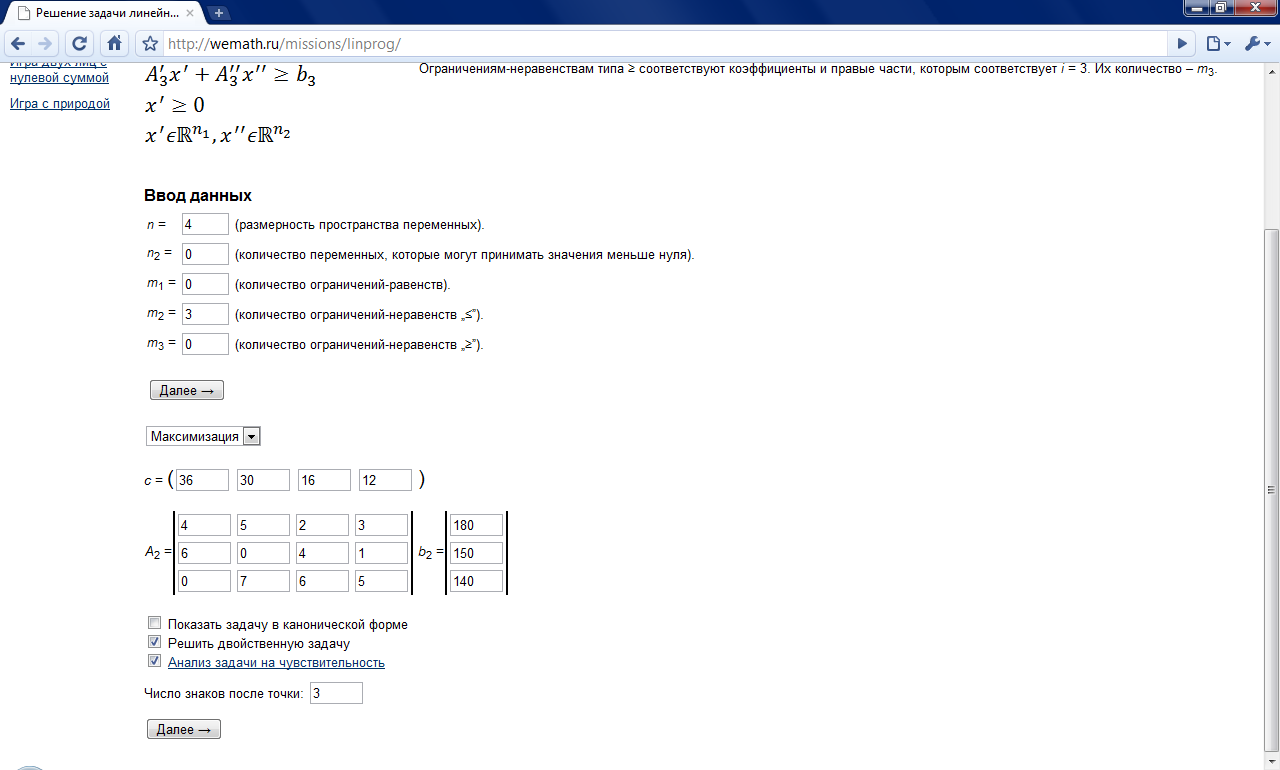


Рисунок 22. – Планирование производства, ввод исходных данных.

3. Получим решение прямой и двойственной задач, а так же интервалы устойчивости.

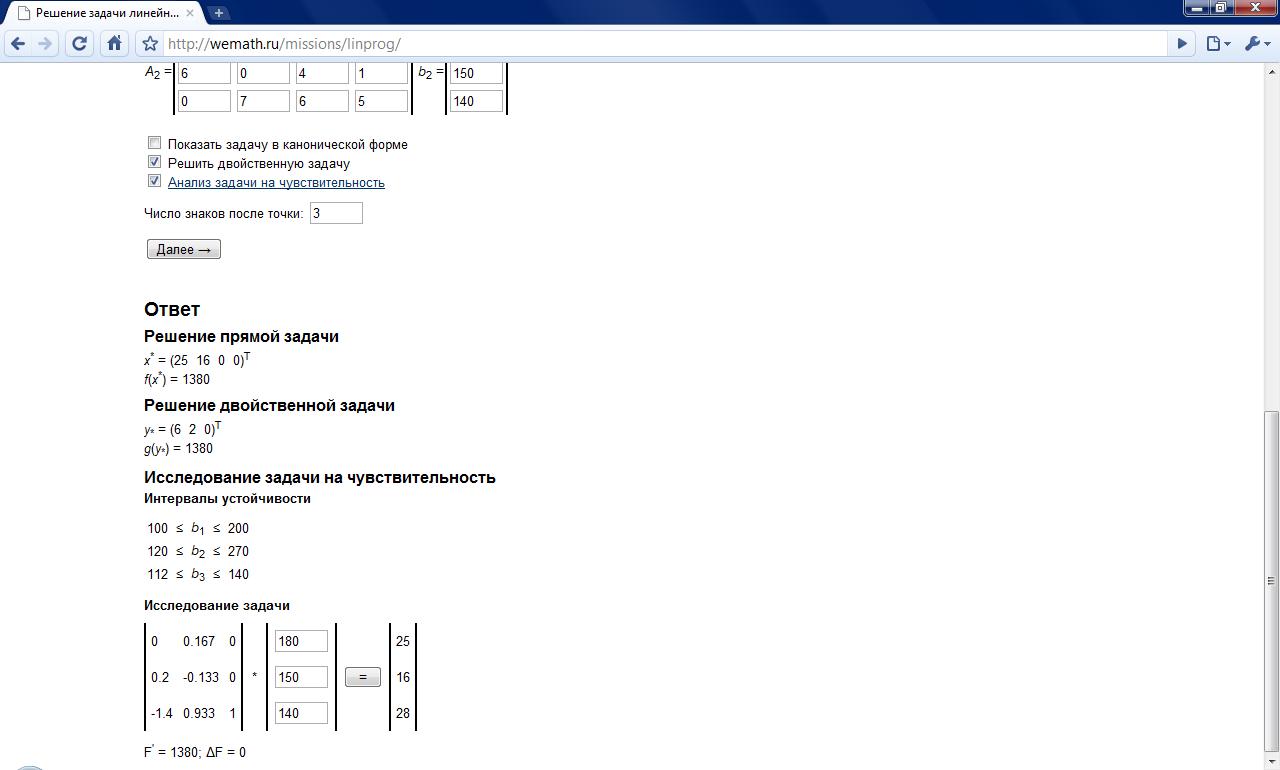


Рисунок 23. – Планирование производства, вывод данных.

Таким образом, максимальную прибыль при выполнении ограничений по ресурсам обеспечивает следующее распределение выпуска продукции:

1. 25,
2. 16,
3. 0,
4. 0.

Прибыль при этом составит 1380 у. е.

Двойственные оценки, показывающие влияние изменений каждого из видов ресурсов, равны:

1. 6,
2. 2,
3. 0.

Теперь посмотрим, как изменится значение целевой функции и оптимальный план, если добавить количество наиболее значимого ресурса I до верхней границы интервала устойчивости. Это можно осуществить с помощью специального интерфейса.

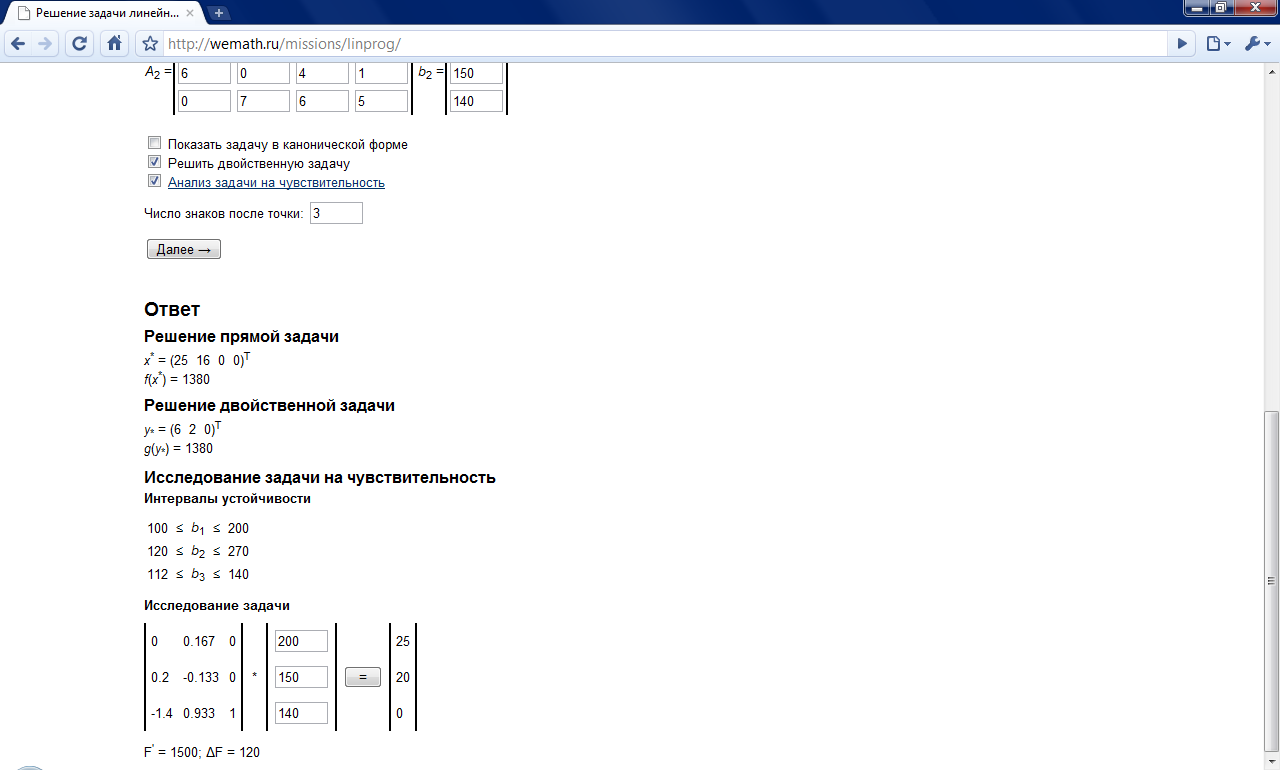


Рисунок 24. – Планирование производства, исследование на чувствительность.

Видно, что при этом оптимальный план изменился так, что прибыль увеличилась на 120 у. е.

### 2.6.2 Задача об оптимальном плане перевозок

Данная задача является транспортной задачей.

У поставщиков *A1*, *A2*, *A3*, *A4* имеются запасы однотипной продукции в размере 48, 30, 27 и 20 соответственно. У потребителей *B1*, *B2*, *B3* , *B4* и *B5* имеются потребности в этой продукции в размере 10, 27, 42, 12 и 26 соответственно.

Стоимости перевозок приведены в таблице 2.

Таблица 2. Матрица стоимостей перевозок.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | B1 | B2 | B3 | B4 | B5 |
| A1 | 10 | 8 | 5 | 6 | 9 |
| A2 | 6 | 7 | 8 | 6 | 5 |
| A3 | 8 | 7 | 10 | 8 | 7 |
| A4 | 7 | 5 | 4 | 6 | 8 |

Найдем такой план перевозок с помощью разработанной системы, которые обеспечит минимальные суммарные затраты.

1. Введем размерности.

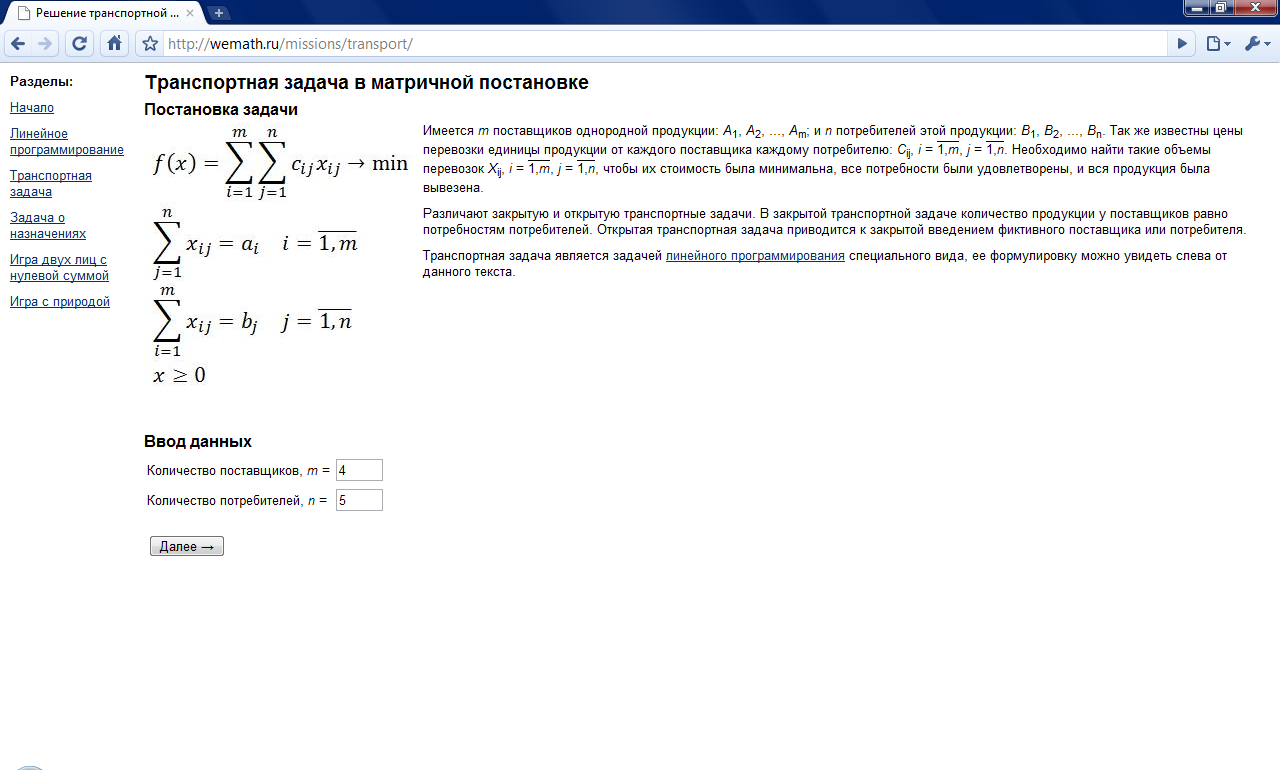


Рисунок 25. – Задача о перевозках, ввод размерностей.

2. Ввод векторов запасов и потребностей, а так же матрицы стоимостей.

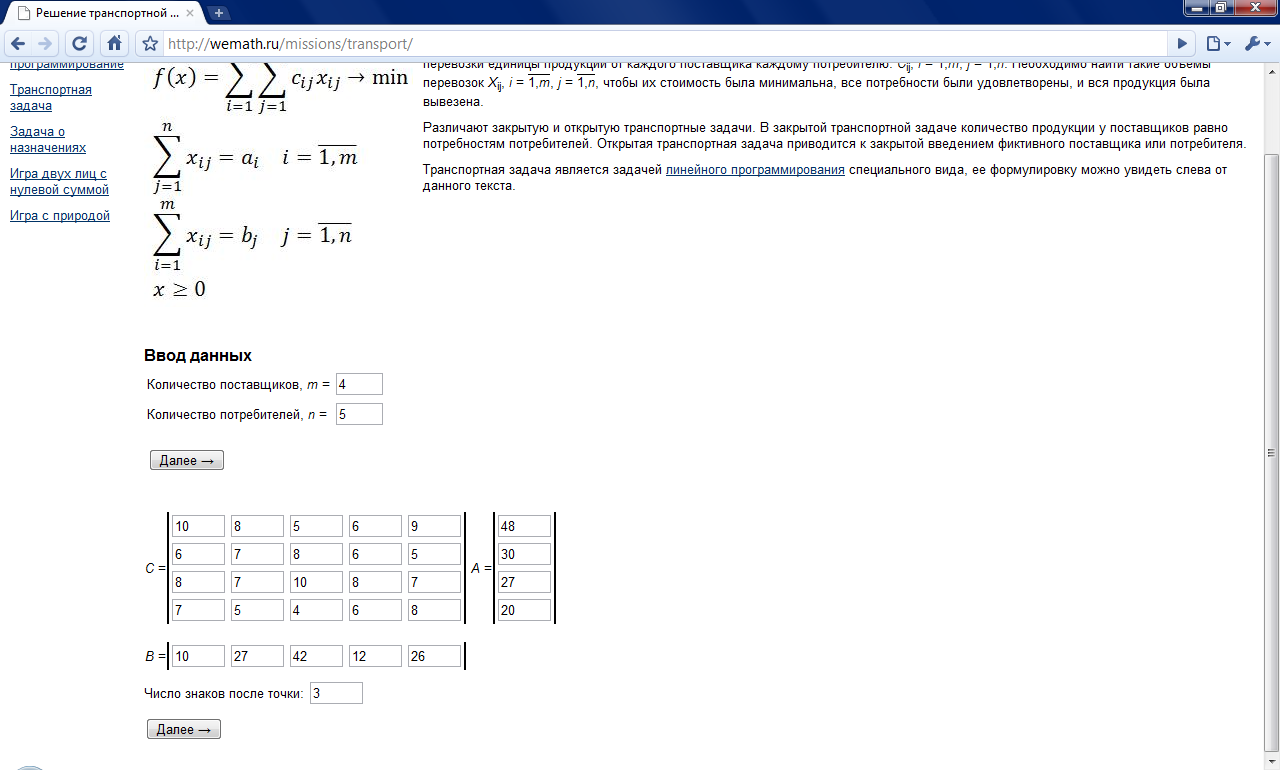


Рисунок 26. – Задача о перевозках, ввод исходных данных.

3. Получим результат.

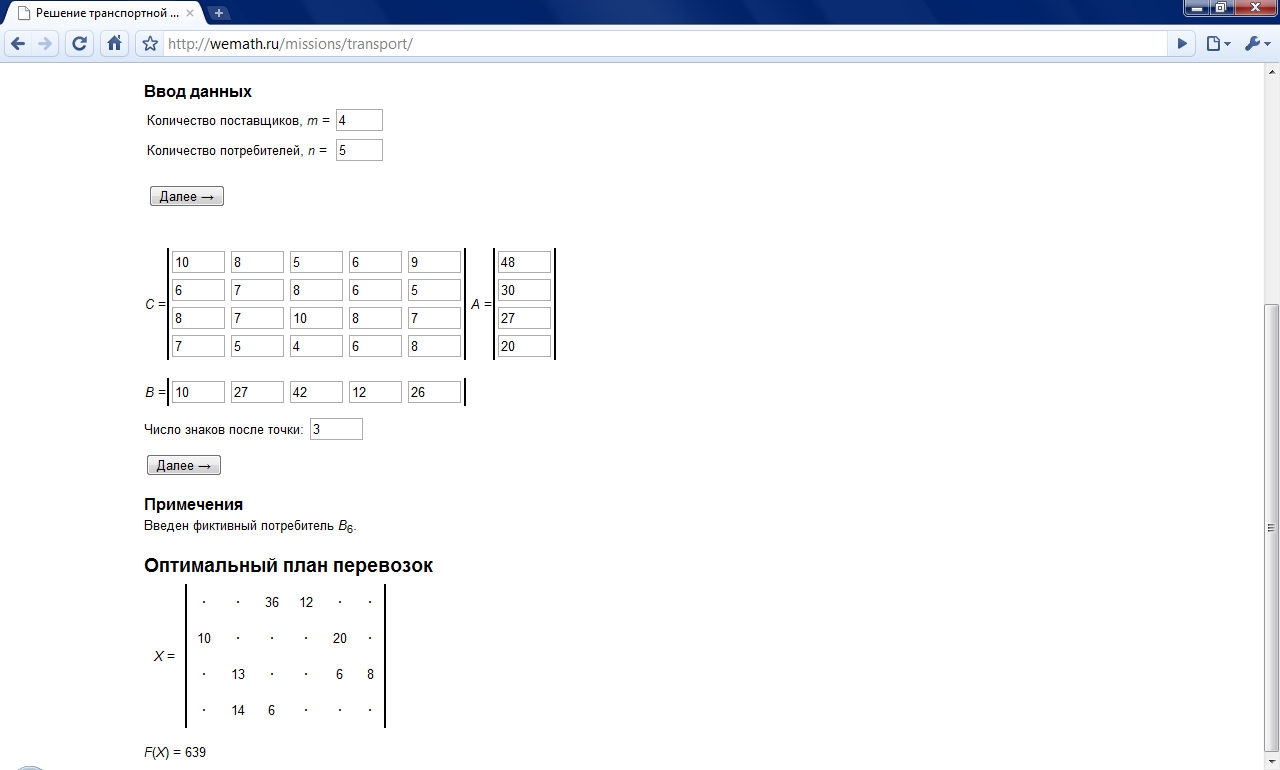


Рисунок 27. – Задача о перевозках, вывод данных.

Таким образом, мы нашли план перевозок, обеспечивающий минимальные затраты, которые составят 639 у. е.

### 2.6.3 Задача о назначении рабочих на виды работ

Данная задача является задачей о назначениях.

Имеется 4 рабочих, которых необходимо распределить по 4 видам работ так, чтобы суммарная стоимость работ была минимальна. Стоимости выполнения каждым из рабочих каждого из видов работ приведены в таблице 3.

Таблица 3. Матрица стоимостей выполнения работ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Работа 1 | Работа 2 | Работа 3 | Работа 4 |
| Рабочий 1 | 50 | 30 | 60 | 20 |
| Рабочий 2 | 70 | 40 | 20 | 30 |
| Рабочий 3 | 90 | 30 | 50 | 120 |
| Рабочий 4 | 70 | 20 | 60 | 70 |

Решим задачу с помощью разработанной системы.

1. Введем данные.

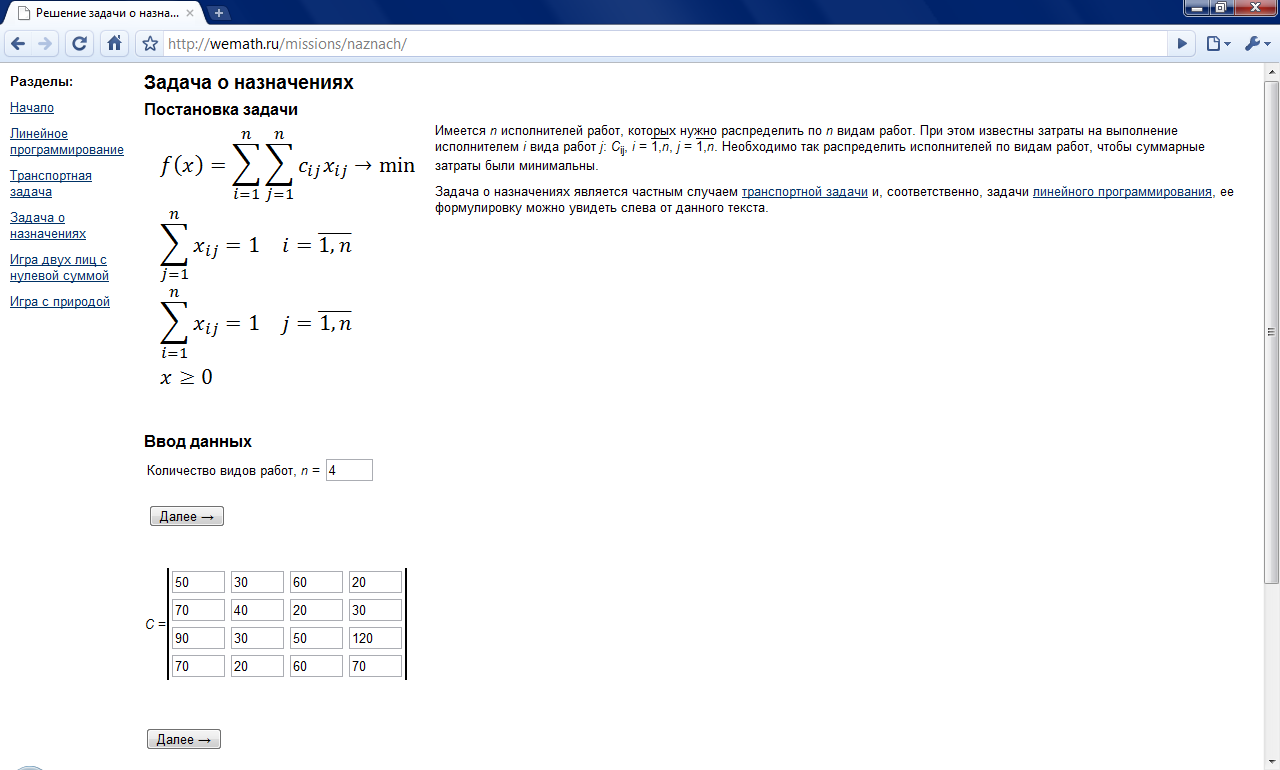


Рисунок 28. – Задача о назначениях, ввод исходных данных.

2. Получим результат.

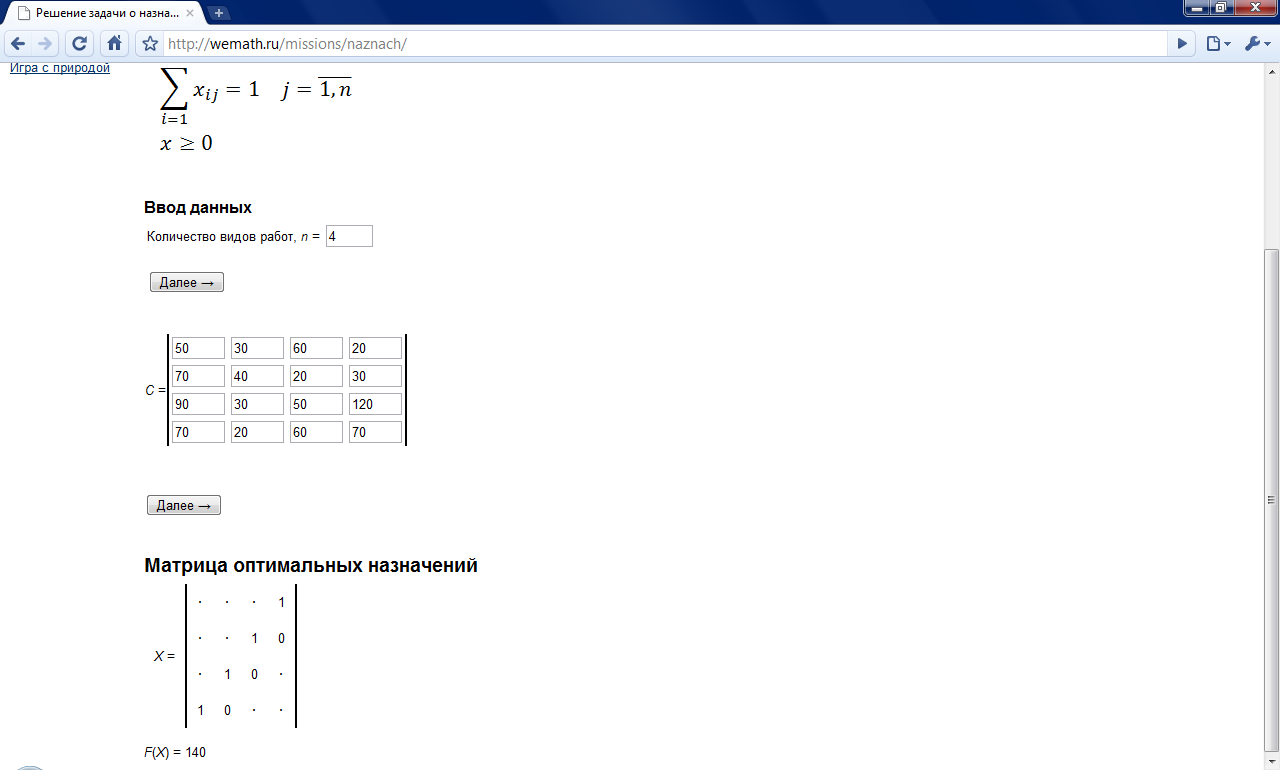


Рисунок 29. – Задача о назначениях, вывод результатов.

Таким образом, рабочий 1 должен быть назначен на работу 4, рабочий 2 – на работу 3, рабочий 3 – на работу 2, а рабочий 4 – на работу 1. Суммарная стоимость работ при этом составит 140 у. е.

### 2.6.4 Задача об оптимальной стратегии поведения на фондовом рынке

Данная задача является задачей решения игры двух лиц с нулевой суммой.

Компании А и В занимают монопольное положение на некотором сегменте фондового рынка. У компании А существует две стратегии поведения на данном рынке, у компании В - четыре. Платежная матрица при различных сочетаниях поведения компаний А и В представлена в таблице 4 (в млн. руб.). Считать возможную игру как игру двух лиц с нулевой суммой. Решить игру.

Таблица 4. Платежная матрица.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | B1 | B2 | B3 | B4 |
| А1 | 0 | 6 | -3 | 4 |
| А2 | 4 | -2 | 3 | -4 |

1. Введем размерности.

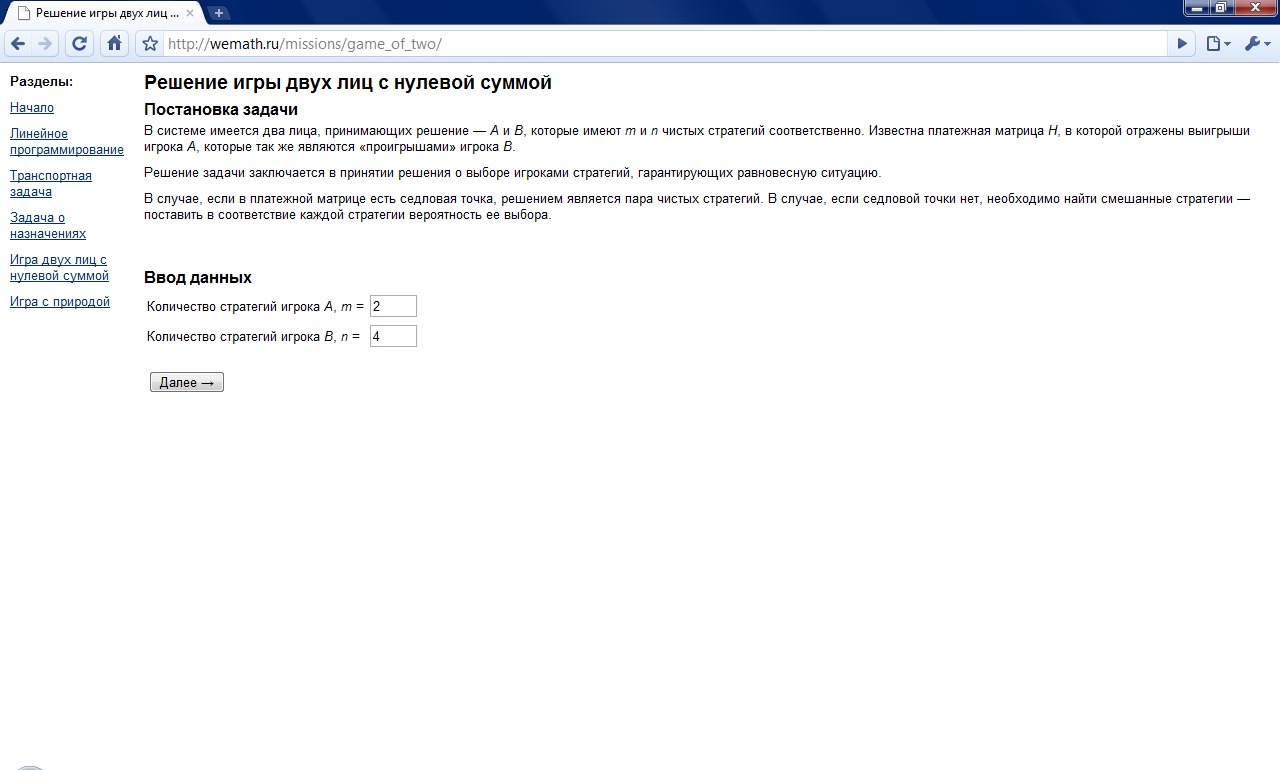


Рисунок 30. – Стратегия на фондовом рынке, ввод размерностей.

2. Введем платежную матрицу.

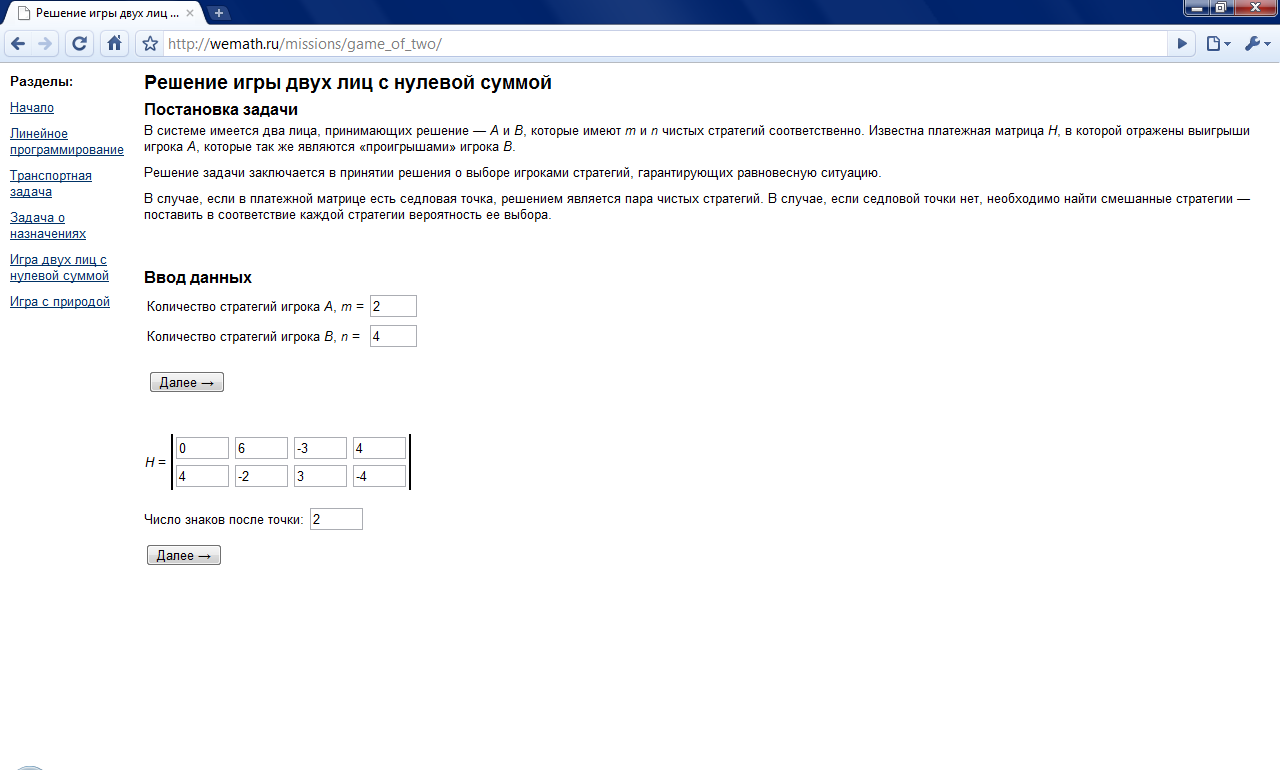


Рисунок 31. – Стратегия на фондовом рынке, ввод платежной матрицы.

3. Получим результат.

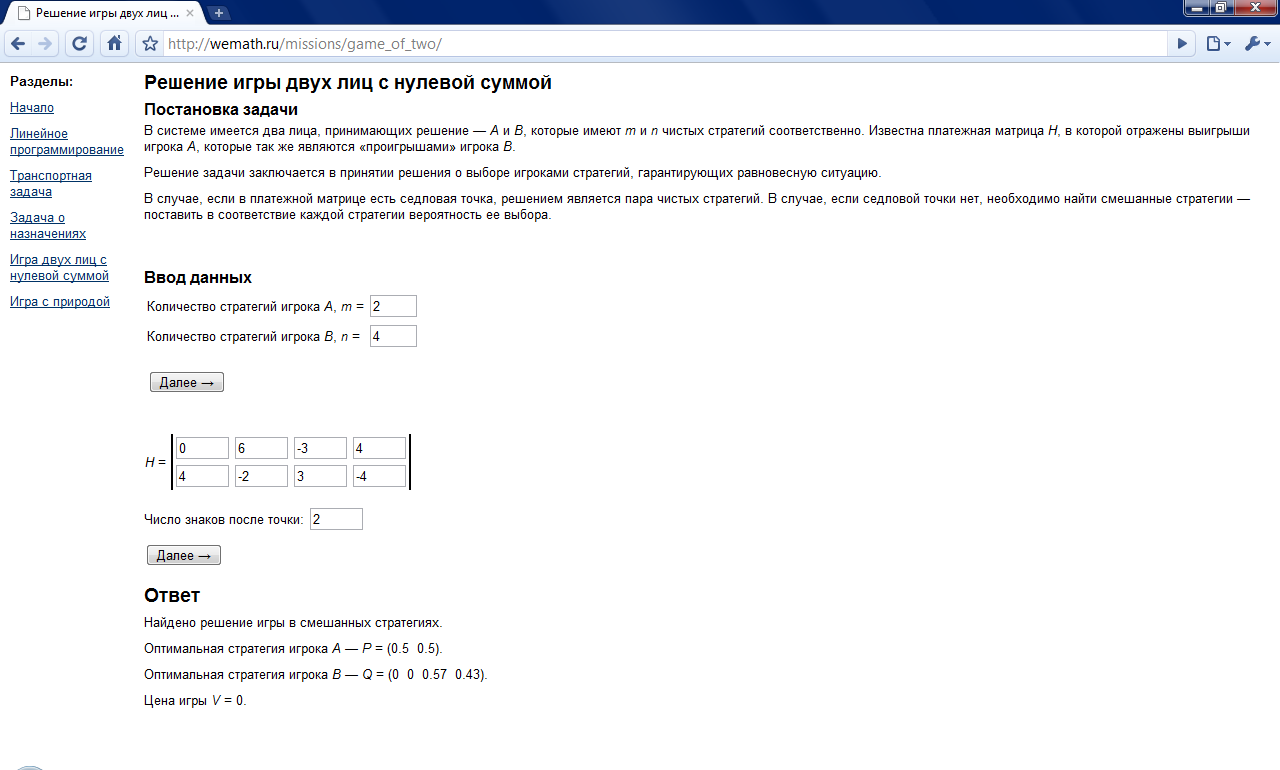


Рисунок 32. – Стратегия на фондовом рынке, вывод результатов.

Таким образом, в данной задаче нет седловой точки, и решение возможно только в смешанных стратегиях. Ожидаемый выигрыш равен нулю, он может быть получен игроками при выборе стратегий с вероятностями *P* и *Q*.

### 2.6.5 Задача о покупке акций

Данная задача является задачей решения игры с природой.

Инвестор может купить акции одной из трех компаний: *K1, K2, K3*. При этом рынок ценных бумаг может находиться в одном из четырех состояний: *П1, П2, П3* или *П4*.

Доходность акций определяется матрицей выигрышей, представленной в таблице 5.

Таблица 5. Матрица выигрышей.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | П1 | П2 | П3 | П4 |
| A1 | 8 | 4 | 6 | 20 |
| A2 | 7 | 7 | 7 | 7 |
| A3 | 6 | 12 | 8 | 10 |

Рассмотреть три ситуации:

1. Нет информации, позволяющей предположить вероятности состояний природы. Рассмотреть показатели критериев Вальда (крайнего пессимизма), максимакса (крайнего оптимизма), Лапласа.
2. Рассмотреть показатель критерия пессимизма-оптимизма Гурвица при .
3. Есть информация о вероятностях состояний природы .

Решим задачу с помощью разработанной системы.

1. Введем размерности и выберем критерии.

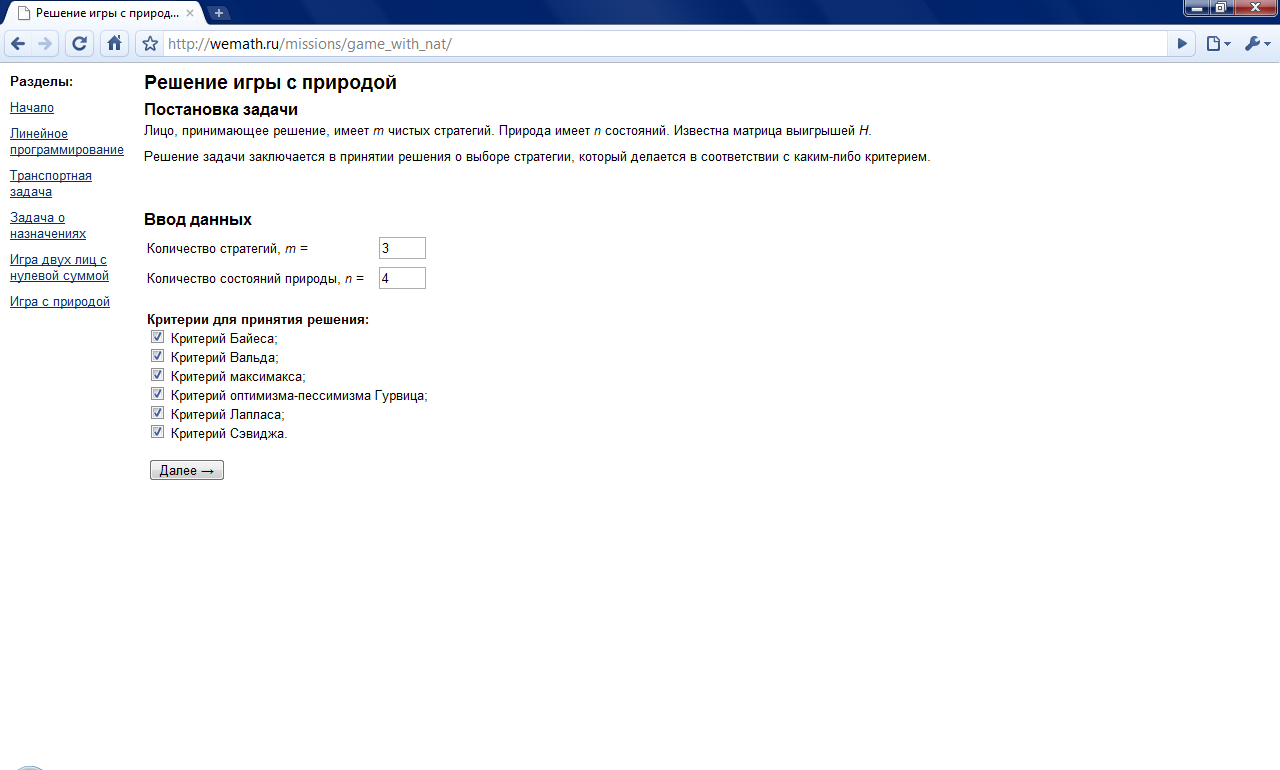


Рисунок 32. – Покупка акций, ввод размерностей и выбор критериев.

2. Введем матрицу выигрышей, вероятности и показатель оптимизма.

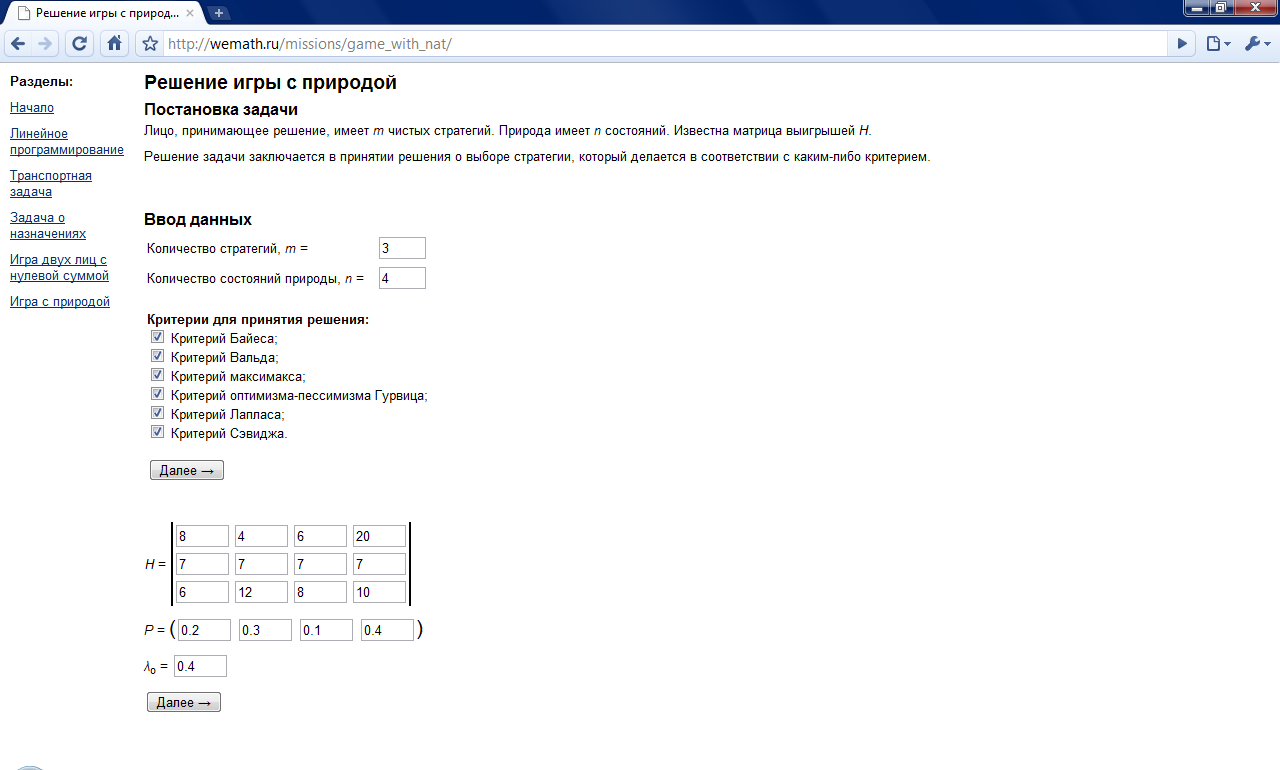


Рисунок 33. – Покупка акций, ввод матрицы выигрышей.

3. Получим результат.

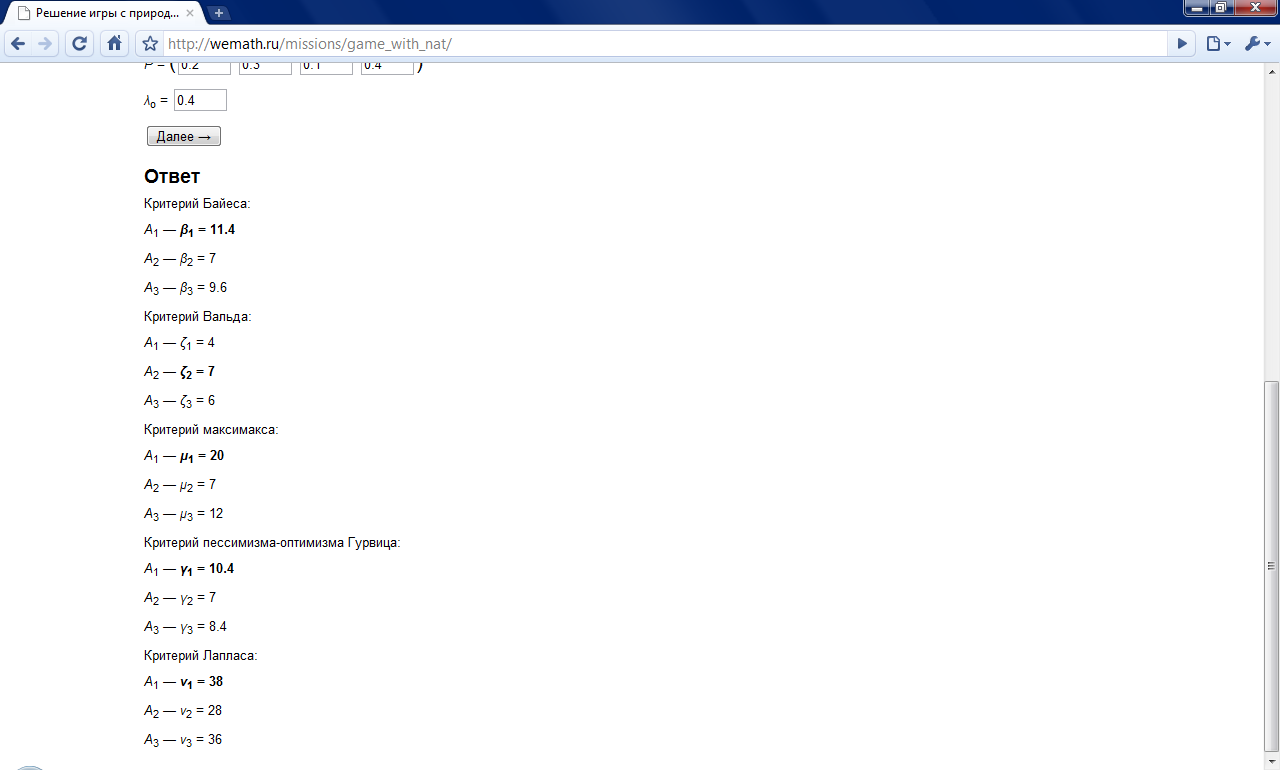


Рисунок 34. – Покупка акций, вывод результатов.

Таким образом, в случае 1 критерий Вальда показывает, что минимальный проигрыш обеспечит стратегия A1, критерий максимакса показывает, что максимальный выигрыш обеспечивает так же стратегия A1. Критерий Лапласа показывает, что если предположить равные вероятности стратегий, наибольшее математическое ожидание выигрыша приходится на стратегию A1. Таким образом, оптимальным выбором будет стратегия 1.

В случаях 2 и 3 оптимальной является так же стратегия A1.

# 3 Экономическая значимость системы

Экономическая система – совокупность всех экономических процессов, совершающихся в обществе на основе сложившихся в нём отношений собственности и хозяйственного механизма. В любой экономической системе первичную роль играет производство в совокупности с распределением, обменом, потреблением. Во всех экономических системах для производства требуются экономические ресурсы, а результаты хозяйственной деятельности распределяются, обмениваются и потребляются.

У функционирования экономических систем есть определенные числовые характеристики, которые могут быть использованы для принятия решений о выборе оптимальных параметров функционирования систем или для выбора оптимальных стратегий с помощью математических методов.

В разработанной системе есть возможность решать такие задачи как задача линейного программирования, транспортная задача, задача о назначениях, задача об игре двух лиц с нулевой суммой, задача об игре с природой. Применение указанных методов делает возможным принятие оптимальных решений в ситуациях, когда без математических методов это было бы невозможным.

Реализация решения задачи линейного программирования в разработанной системе позволяет решить достаточно широкий круг задач, в которых требуется найти оптимальные параметры функционирования экономической системы, где важные зависимости описываются линейными функциями. Важным является то, что есть возможность получить решение двойственной задачи и интервалы устойчивости решения относительно изменений вектора правых частей ограничений.

Линейное программирование можно использовать для принятия оптимальных решений в следующих ситуациях:

1. Планирование объемов производства нескольких видов продукции, на каждый из которых используются определенные ресурсы. Можно найти такие объемы выпуска каждого из видов, при которых прибыль будет максимальной. Для этого предварительно необходимо оценить прибыль от продажи единицы каждого из видов продукции. С помощью двойственной задачи можно оценить дефицитность каждого из видов ресурсов, а с помощью интервалов устойчивости – допустимые пределы вариации запасов ресурсов.
2. Планирование площадей для посевов. Если имеется несколько участков, на которых возможно высадить какие-либо сельскохозяйственные культуры, при этом есть возможность оценить прибыль от продажи единицы объема или массы каждой из культур, а так же ее урожайность на каждом из участков, можно так распределить посевы по участкам, что прибыль от их продажи после сбора урожая будет максимальной.
3. Задача о выборе технологий для производства продукции. Данная задача имеет место, когда на предприятии есть возможность использовать разные технологии для производства нескольких видов продукции. Необходимо так распределить выпуск между технологиями, чтобы прибыль была максимальной.
4. Формирование портфеля ценных бумаг без учета колебаний курсов ценных бумаг при ограничении минимальной ожидаемой прибыли и максимальных вложений в покупку ценных бумаг. Данная модель несовершенна, так как не учитывает колебаний курсов, но может быть применена в некоторых случаях.
5. Решение игры двух лиц в смешанных стратегиях. Данную задачу теории игр можно решить с помощью линейного программирования. В данном случае рассчитываются вспомогательные величины, через которые можно найти смешанные стратегии и ожидаемый выигрыш.

Эти, а так же многие другие задачи линейного программирования могут быть решены с помощью разработанной системы.

Транспортная задача в классической постановке является задачей о создании такого плана перевозок продукции от поставщиков к потребителям, которому соответствуют минимальные денежные затраты либо затраты времени. Исходными данными являются: запасы продукции у поставщиков, потребности в продукции у потребителей и затраты на перевозку единицы продукции от каждого из поставщиков к каждому из потребителей. На выходе получается план перевозок, обеспечивающий минимальные затраты. Так же к транспортной могут быть сведены некоторые другие задачи.

Задача о назначениях в классической постановке является задачей о назначении работников на рабочие места в случае, если известны стоимости выполнения каждым рабочим каждого из видов работ. Оптимальным в данной задаче является план назначений, которому соответствовала бы минимальная суммарная стоимость выполнения работ.

Задачи теории игр, игра двух лиц с нулевой суммой и игра с природой, необходимы в случаях принятия решения при игровой неопределенности, то есть, в случаях конкуренции, конфликта интересов. В условиях рыночной экономики подобные задачи возникают достаточно часто. К подобным задачам можно отнести, например такие задачи как задача определения оптимальных объемов товаров для продажи в условиях конкуренции или задача выбора оптимальных моментов времени для выставления товара на продажу в условиях конкуренции.

Для решения перечисленных задач могут быть использованы и другие системы, но подавляющее их большинство является платными программными продуктами, ориентированными на профессиональную аудиторию, что делает их стоимость особенно высокой. Это затрудняет использование таких систем в небольших фирмах или учебных заведениях.

Так же надо сказать, что часто такие системы слишком сложны для пользователей, и для их использования необходимо дополнительное обучение.

Разработанная система лишена этих недостатков, так как является бесплатной, базируется на бесплатном программном обеспечении, и в ней используется ввод данных в формы, которые являются шаблонами ввода, так же в ней используются общепринятые математические обозначения и экономические термины.

# 4 Эргономическое обеспечение системы

При разработке системы для решения задач математической экономики стояла задача предусмотреть требования по эргономическому обеспечению программного продукта.

Группа эргономических стандартов включает требования и рекомендации по охране здоровья и условий труда. Они касаются освещения, конструкции аппаратуры, удобства расположения органов управления и экрана монитора относительно уровня глаз, возможностей поворота дисплея для обеспечения его удобного положения. К числу эргономических стандартов относятся международный стандарт BS 7179 и пришедший ему на смену ISO 9241-3. Эргономические нормы включены в комплексный шведский стандарт ТСО-99.

Наиболее важные эргономические требования к мониторам, связанные с частотой кадровой развертки не ниже 75 Гц (по TCO-99 не ниже 85 Гц), заключены в стандарте ErgoVGA ассоциации VESA.

Так же к эргономическому обеспечению можно отнести оформление интерфейса системы, так как работа с интерфейсом, оформленным неправильно, может нанести вред здоровью пользователя.

Интерфейс является физическим динамическим устройством, взаимодействующим с пользователем, который в некотором роде соответствует обычному текстовому объекту (размер шрифта, цветовое оформление, размер и толщина книги, защита от старения и разрушения, навигация по страницам и т.д.). Возникает эргономический аспект. В случае компьютерного интерфейса появляются новые особенности, связанные с комфортностью экранного представления, достаточной оперативностью реакции программного средства на действия пользователя, удобством манипулирования мышью и клавиатурой (и их скоростными показателями).

Нормативные требования по эргономике пользовательского интерфейса относятся к психофизиологическим свойствам конкретной реализации уже выбранного типа пользовательского интерфейса (и соответствующего стандарта) в конкретном приложении. В этих условиях эргономические стандарты могут лишь требовать достижения некоторых общих руководящих эргономических принципов, которым должно удовлетворять реализация в приложении выбранного типа.

В веб-интерфейсе разработанной системы учтены эргономические требования. Шрифт имеет достаточный размер для комфортного восприятия (10-14 пунктов). Текст хорошо контрастирует с фоном (черный текст на белом фоне). Ссылки на другие страницы интерфейса подсвечены и легко распознаются в тексте. Элементы управления расположены так, чтобы их использование было максимально удобным.

# Заключение

Результатом работы стала система для решения задач математической экономики, в которой возможно решить следующие задачи: задача линейного программирования, транспортная задача, задача о назначениях, игра двух лиц с нулевой суммой, игра с природой.

Система имеет клиент-серверную архитектуру и веб-интерфейс. Ввод данных осуществляется с помощью шаблонов.

Система свободно доступна по адресу http://wemath.ru. Использовать данную систему можно на различных аппаратных и программных платформах, для ее использования не нужны специальные знания, относящиеся к правилам использования данной системы. Система соответствует новым тенденциям переноса работы над данными в Интернет.

Дальнейшими шагами в развитии системы может быть как техническое развитие системы – дополнение ее новыми задачами, совершенствование интерфейса и т. д., – так и привлечение к системе пользователей. Возможным вариантом развития системы является превращение ее в проект с открытым исходным кодом.

# Список использованных источников

1. Бабешко Л. О. Экономико-математическое моделирование – М.: ЭКЗАМЕН XXI, 2004;
2. Кузнецов А. В. Математическое программирование – Мн.: Вышэйшая школа, 1994;
3. Федосеев В. В. Экономико-математические методы и прикладные модели – М.: ЮНИТИ, 1999;
4. Васильев Ф. П. Линейное программирование – М.: Факториал, 1998;
5. Пантелеев А. В. Методы оптимизации в примерах и задачах – М.: Высшая школа, 2002;
6. Шиндер Д. Основы компьютерных сетей – М.: Издательский дом «Вильямс», 2003;
7. Котеров Д. В. PHP5 в подлиннике – СПб.: БХВ-Петербург, 2006;
8. Рыжиков Ю. В. Современный Фортран – М.: КОРОНА ПРИНТ, 2007;
9. Савич У. Программирование на C++ – СПб.: Питер, 2004;
10. Коршунов Ю. М. Математические основы кибернетики – М.: Энергия, 1980.