ВВЕДЕНИЕ

Одним из важнейших этапов решения задач математического программирования является построение экономико-математической модели. Когда докомпьютерная модель уже формализована, необходимо выбрать программное обеспечение для реализации модели. Это могут быть самые разнообразные приложения, например, табличный процессор MS Excel; пакет моделирования систем массового обслуживания GPSS, пакеты для моделирования экономической динамики IThink или Poversim, пакеты моделирования математических и технических систем MatLab и Simulink, MathCAD, Maple и многое другое.

Можно применить универсальные языки программирования типа C++, C#, Java, Python, но в ряде случаев трудоемкость программирования интерфейса превышает трудоемкость программирования самой модели в 5-10 раз.

Иногда, даже в таких специализированных пакетах, как MS Excel, приходится программировать процедуры, которых не хватает для полного решения задачи. Для этого специализированные пакеты имеют встроенные языки программирования, в частности, в MS Office - это Visual Basic for Application (VBA), в MatLab - язык m-файлов. Для разработки качественного приложения в MS Excel экономисту придется отвлечься от экономики и заняться профессиональным программированием в VBA. Поэтому для некоторых задач уместно применять системы компьютерной математики, такие как Maple. Для этих систем разработано много специализированных пакетов для анализа различных оптимизационных задач. Использование подобных программных пакетов, а также традиционных языков программирования для решения всевозможных оптимизационных задач является весьма актуальным для многих приложений. В связи с этим актуальным является сравнительный анализ программных пакетов,

обладающих встроенными языками сверхвысокого уровня, и общественных языков программирования для решения линейных оптимизационных задач.

Объектом данной выпускной квалификационной работы являются задачи линейного программирования.

Предметом исследования являются программные продукты и языки программирования, которые используются для решения задач линейного программирования.

Целью работы является сравнительный анализ программных средств для решения линейных оптимизационных задач с перспективой их применения в школьном курсе информатики на профильном уровне.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Ознакомится с наиболее популярными классами линейных задач;
2. Ознакомится с часто используемыми программными продуктами;
3. Выполнить сравнительный анализ программных продуктов на основе набора тестовых задач.

Глава 1 Оптимизационные задачи линейного программирования

1.1 Решение оптимизационных задач линейного программирования

Большое количество планово-производственных и экономических задач связана с распределением каких-либо, как правило, ограниченных ресурсов (сырья, рабочей силы, энергии, топлива и т.д.). Часто распределение ресурсов можно осуществить не единственным образом. Например, данную продукцию можно получить различными способами, по-разному выбирая технологию, сырье, применяя оборудования, организацию процесса. При этом каждый способ распределения ресурсов, оценивается с позиции некоторого критерия (прибыль, объем выпуска продукции и т.д.), характеризуется определенным значением показателя этого критерия. Естественным является намерение найти такой вариант распределения (программу, план), который бы гарантировал наибольший экономический эффект. Такой план называют оптимальным.

Реальные экономические процессы достаточно сложны. При их математическом описании приходится учитывать много различных факторов. Поэтому математическая модель содержит большое число условий ограничений со многими неизвестными. Если неизвестные входят в модель только в первой степени, то задача относится к разделу линейного программирования, в противном случае - в раздел нелинейного программирования. Оптимизационные задачи, в которых нужно учитывать последовательность действий или фактор времени, рассматриваются в разделе динамического программирования. Если в задаче фигурируют параметры, являются случайными величинами, то она относится к задачам стохастической оптимизации.

Предметом исследования математического программирования являются математические модели, связанные в большинстве случаев с определенными экономическими процессами, описывающих экономику

предприятия, промышленного объединения, народного хозяйства или отдельных экономических процессов в них.

Математическое программирование – это раздел математики, который разрабатывает теорию и численные методы решения многомерных задач с ограничениями, то есть задач на экстремум функции многих переменных с ограничениями на область изменения этих переменных. В отличие от классической теории экстремальных задач, которая является частью математического программирования, основное внимание уделяется тем задачам, в которых активно участвуют ограничения на область изменения переменных.

1.2 Примеры задач линейного программирования

1. Задача оптимального производственного планирования

Одной из наиболее распространенных задач данной группы является задача о максимальном выпуске продукции с имеющимися ограниченными запасами сырья при различных технологиях производства. Сюда же можно отнести задачу достижения максимальной рентабельности предприятия при производстве различных видов продукции.

Пример 1. Для изготовления трех видов изделий A, B и C используется токарное, фрезерное, сварочное и шлифовальное оборудование. Затраты времени на обработку одного изделия для каждого из типов оборудования указаны в таблице 1.1. В ней же указан общий фонд рабочего времени каждого из типов используемого оборудования, а также прибыль от реализации одного изделия каждого вида.

Таблица 1.1

Тип оборудования	Затраты времени (станков в час.) На обработку одного изделия вида			Общий фонд рабочего времени оборудования (ч.)
	A	B	C	
Фрезерное токарное сварочное шлифовальный	2 1 7 4	4 8 4 6	5 6 5 7	120 280 240 360
Прибыль (руб.)	10	14	12	

Требуется определить, сколько изделий и какого вида нужно изготовить предприятию, чтобы прибыль от их реализации была максимальной. Составить математическую модель задачи.

2. Задача об оптимальном составе смеси

Достаточно широкий класс задач программирования составляют так называемые задачи на составление смесей или задачи на использование заменителей.

Пример 2. При откорме животных каждое животное ежедневно должен получать не менее 60 единиц питательного вещества А, не менее 50 единиц вещества В и не менее 12 единиц вещества С. Указанные питательные вещества содержатся в трех видах корма. Состав единиц питательных веществ в 1 кг каждого из видов корма приведен в таблице 2.2:

Таблица 2.2

питательные вещества	Количество единиц питательных веществ в 1 кг корма вида		
	I	II	III
A *B*	1 2	3 4	4 2

C	1	4	3

Составить дневной рацион, обеспечивающий получение необходимого количества питательных веществ при минимальных денежных затратах, если цена 1 кг корма I вида составляет 9 коп., Корма II вида - 12 коп. и корма III вида - 10 коп.

3. Задача об оптимальном плане перевозок

Эту задачу часто называют транспортной. В простом варианте транспортная задача возникает при необходимости наиболее рационального перевозки некоторого однородного груза. При этом потребителям безразлично, из каких пунктов он поступает, важно, чтобы был удовлетворен спрос, а каждый поставщик имеет возможность поставлять груз любому потребителю. Обратные перевозки не предвидятся.

Пример 3. В трех пунктах отправления сосредоточен однородный груз в количествах, соответственно равных 420, 380 и 400 т. Этот груз необходимо перевезти в три пункта назначения в количествах, соответственно равных 260, 520 и 420 т. Стоимости перевозок 1 т груза с каждого пункта отправления в каждый пункт назначения известны величинами и задаются матрицей

$$(c_{ij}) = \begin{pmatrix} 2 & 4 & 3 \\ 7 & 5 & 8 \\ 6 & 9 & 7 \end{pmatrix}.$$

Найти план перевозок, обеспечивающий вывоз имеющегося в пунктах отправления и завоз необходимого в пункты назначения груза при минимальной общей стоимости перевозок.

4. Задача об оптимальном размещении производства. Это одна из важных модификаций транспортной задачи.

Математическая модель задачи:

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} (c_i + Ek_i + c_{ij}) x_{ij} \quad \rightarrow \quad \min$$

при ограничениях:

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = a_i, \quad i = \overline{1, m} \\ \sum_{i=1}^{m} x_{ij} \le b_j, \quad j = \overline{1, n} \\ x_{ij} \ge 0, \quad i = \overline{1, m}, \quad j = \overline{1, n} \end{cases},$$

где c_i – затраты на производство единицы продукции на действующем предприятии; E – нормативный коэффициент эффективности капитальных вложений; k_i – капитальные удельные вложения в *i*-ое предприятие; x_{ij} – искомый объем перевозок продукции *i*-го поставщика *j*-му потребителю; a_i – мощность *i*-го предприятия-поставщика; b_j – спрос *j*-го потребителя;

Эта модель представляет собой открытую транспортную задачу, которая приводится к закрытой введением условного потребителя. Поставщиков, которые в оптимальном плане «прикрепились» к условному потребителю, использовать нецелесообразно. Если это относится к проектируемому предприятию, то соответствующий вариант отбрасывается, как нерациональный.

Если же к условному потребителю «прикрепилось» какое-либо действующее предприятие, то следует рассмотреть вопрос о целесообразности его дальнейшей эксплуатации. Как правило, в оптимальный план попадают действующие предприятия.

Приведенные затраты на единицу продукции на этих предприятиях ниже, чем на проектируемых. Здесь к затратам на производство продукции добавляются капитальные затраты.

Основная трудность при решении такого рода задач заключается в возможности получения нецелочисленных решений, когда в оптимальном плане часть мощности какого-либо поставщика относится на действительных, а часть – на условных потребителей. В таких случаях приходится останавливаться на приближенных решениях.

Пример 4. Два действующих предприятия A_1 и A_2 имеют дневную производительность 240 и 360 единиц продукции. Предприятия обеспечивают трех потребителей с потребностями 220, 280 и 200 единиц продукции. Недостающий прирост мощности планируется за счет реконструкции первого предприятия A_1 и (или) строительства нового предприятия A_3. Себестоимость единицы продукции на действующих предприятиях 3,5 и 3 тыс. руб., после реконструкции действующего предприятия – 3 тыс. руб., в результате строительства нового предприятия – 2,5 тыс. руб. Удельные капитальные затраты на реконструкцию – 2 тыс. руб., а на строительство – 3,5 тыс. руб. Матрица транспортных затрат за доставку единицы продукции известна:

$$C = \begin{pmatrix} 3 & 6 & 5 \\ 3 & 3 & 4 \\ 4 & 5 & 2 \end{pmatrix}.$$

Определить оптимальный план перевозок и прироста мощностей.

Решение: $\sum_i a_i = 240 + 360 = 600$, $\sum_j b_j = 220 + 280 + 200 = 700.$

Каждому проектируемому варианту прироста мощности выделяем отдельную строку и даем недостающую мощность 100 ед. Затраты для действующих предприятий складываются из затрат на производство единицы продукции и транспортировку $c_i + c_{ij}$; для вариантов реконструкции и строительства учитываются еще удельные капитальные затраты: $c_i + c_{ij} + Ek_i$. Первоначальный опорный план X_0 составляем по методу минимальной стоимости. Последовательность заполнения клеток: (2, 2); (2, 1); (1, 1); (4, 3); (1, 3); (3, 3); (3, 4)

Матрица планирования	220	280	200	100	u_i
A_1: 240	+ 6,5 140	9,5	- 8,5 100	0	$u_1 = 6,5$
A2: 360	- 6 80	6 280	+ 7 x	0	$u_2 = 6$

Рек. A_1: 100	8	11	10 0	0 100	$u_3 = 8$
Стр. A_3: 100	10	11	8 100	0	$u_4 = 6$
v_j	$v_1 = 0$	$v_2 = 0$	$v_3 = 2$	$v_4 = -8$	**4720**

Вычисляем стоимость этого плана:

$Z_0 = 140 \cdot 6,5 + 100 \cdot 8,5 + 80 \cdot 6 + 280 \cdot 6 + 0 \cdot 10 + 100 \cdot 0 + 100 \cdot 80 = 4720$.

Задачу решаем методом потенциалов. Последовательность определения потенциалов: $v_1 = 0$; $u_1 = 6,5$; $u_2 = 6$; $v_2 = 0$; $v_3 = 2$; $u_3 = 8$; $u_4 = 6$; $v_4 = -8$.

Последовательность определения оценок незанятых клеток:

$\Delta_{12} = -3$; $\Delta_{14} = -1,5$; $\Delta_{23} = 1$; $\Delta_{24} = -2$; $\Delta_{31} = 0$; $\Delta_{32} = -3$; $\Delta_{41} = -4$; $\Delta_{42} = -5$; $\Delta_{44} = -2$.

План X_0 не является оптимальным, так как содержит положительную оценку $\Delta_{23} = 1$, строим для нее цикл: (2, 3) – (2, 1) – (1, 1) – (1, 3) – (2, 3).

Свободную клетку помечаем знаком + и далее чередуем знаки по вершинам контура.

Величина перемещаемой поставки равна $x = \min\{x_{ij}\} = \min\{80, 100\} = 80$. После перераспределения объемов поставок по контуру, получим новый план X_1 при этом целевая функция Z уменьшится на величину: $\Delta_z = \Delta_{23} \cdot x = 1 \cdot 80 = 80$.

Матрица планирования	220	280	200	100	u_i
240	6,5 220	9,5	8,5 20	0	$u_1 = 0$
360	6	6 280	7 80	0	$u_2 = -1,5$
100	8	11	10	0 100	$u_3 = 0$
100	10	11	8 100	0	$u_4 = 0,5$
v_j	$v_1 = 6,5$	$v_2 = 7,5$	$v_3 = 8,5$	$v_4 = 0$	**4640**

Из таблицы оптимального плана получаем, что оптимальным вариантом прироста мощности является вариант строительства нового предприятия, так как вариант реконструкции приходится на «фиктивного» потребителя.

Полученное оптимальное решение является целочисленным, т.е. все мощности предприятий не дробятся между реальными и фиктивными потребителями.

5. Задача динамики производства и создания запасов. Эта задача заключается в оптимальном распределении некоторой продукции, и ее запасов. В условиях действующего предприятия всякое изменение объема выпуска продукции связана с дополнительными затратами. Хранение готовой продукции также требует определенных затрат. В случаях, когда спрос на готовую продукцию в отдельные отрезки времени не постоянен, возникает потребность поиска такого компромиссного плана выпуска, при котором суммарные затраты на расширение и свертывания производства, а также на хранение остатков продукции были бы минимальными при условии своевременного и полного удовлетворения потребностей.

6. Задача о назначениях.

Одним из ответвлений транспортной задачи является задача о назначениях, которая подразумевает равенство пунктов производства и пунктов, а поэтому таблица является квадратной. Помимо этого каждый пункт имеет объем потребностей, который равен 1, в свою очередь величина предложения тоже равна 1. Чтобы решить данную задачу можно воспользоваться методами решения задач линейного программирования, а можно непосредственно обратиться к методу решения транспортной задачи.

7. Стохастическая задача комплектования станочного парка

В предыдущем параграфе были рассмотрены примеры задач линейного программирования. Во всех этих задачах нужно найти максимум или

минимум линейной функции при условии, что ее переменные приобретают неотрицательные значения и удовлетворяют некоторой системе линейных уравнений или линейных неравенств или системе, содержащей как линейные неравенства, так и линейные уравнения. Каждая из этих задач является частным случаем общей задачи линейного программирования.

1.3 Общая и основная задачи линейного программирования

Общей задачей линейного программирования называется задача, которая заключается в определении максимального (минимального) значения линейной функции от нескольких переменных

$$F = \sum_{j=1}^{n} c_j x_j \tag{1.1}$$

с учетом наложенных на нее ограничений вида

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \left(i = \overline{1,k}\right), \tag{1.2}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \left(i = \overline{k+1,m}\right), \tag{1.3}$$

$$x_i \ge 0 \left(j = \overline{1,l}, l \le n\right), \tag{1.4}$$

где a_{ij}, b_i, c_j - заданные постоянные величины и $k \le m$.

Функция (1.1) называется целевой функцией (или линейной формой) задачи (1.1) - (1.4), а условия (1.2) - (1.4) - ограничениями данной задачи.

Стандартной (или симметричной) задачей линейного программирования называется задача, которая заключается в определении

максимального значения функции (1.1) при выполнении условий (1.2) и (1.4), где k = m и l = n.

Канонической (или основной) задачей линейного программирования называется задача, которая заключается в определении максимального значения функции (1.1) при выполнении условий (1.3) и (1.4), где k = 0 и l = n.

Совокупность чисел X = (x_1, x_2, …, x_n), удовлетворяющих ограничениям задачи (1.3) - (1.4), называется допустимым решением (или планом).

План X` = ($x_1$`, x_2`, …, $x_n$`), при котором целевая функция задачи (1.1) приобретает своего максимального (минимального) значения, называется оптимальным.

Значение целевой функции (1.1) при плане X будем обозначать через F(X). Итак, X` - оптимальный план задачи, если для любого X выполняется неравенство $F(X) \le F(X`)$ [соответственно $F(X) \ge F(X`)$].

Указанные выше три формы задачи линейного программирования эквивалентны в том смысле, что каждая из них с помощью несложных преобразований может быть переписана в форме другой задачи. Это означает, что если есть способ нахождения решения одной из указанных задач, то тем самым может быть определен оптимальный план любой из трех задач.

Чтобы перейти от одной формы записи задачи линейного программирования в другую, нужно в общем случае уметь, во-первых, сводить задачу минимизации функции к задаче максимизации, во-вторых, переходить от ограничений-неравенств к ограничениям-равенства и наоборот, в-третьих, заменять переменные, которые не удовлетворяют условиям не отрицательности.

В том случае, когда требуется найти минимум функции $F = c_1x_1 + c_2x_2 + \dots + c_nx_n$, можно перейти к нахождению максимума функции $F = -F = -c_1x_1 - c_2x_2 - \dots - c_nx_n$, поскольку $\min F = -\max(-F)$.

Ограничение-неравенство исходной задачи линейного программирования, имеет вид «¿ », Можно превратить в ограничения-равенства добавлением к его левой части дополнительной искусственной переменной, а ограничения-неравенства вида«¿ »- в ограничении-равенство вычитанием из его левой части дополнительной искусственной переменной. Таким образом, ограничение-неравенство

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n\leq\ b_i$$

превратится в ограничения-равенства

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n+x_{n+1}=b_i\left(x_{n+1}\ \geq\ 0\right),$$

(1.5)

а ограничение-неравенство

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n\geq b_i$$

- в ограничении-равенство

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n-x_{n+1}=b_i\left(x_{n+1}\geq\ 0\right).$$

(1.6)

В то же время каждое уравнение системы ограничений

$$a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n=b_i$$

можно записать в виде неравенств:

$$\left\{a_{i1}x_1+a_{i2}x_2+\ldots+a_{in}x_n\leq b_i,\text{¿¿¿¿}\right.$$

(1.7)

Число дополнительных неотрицательных переменных, вводимых при преобразовании ограничений-неравенств в ограничения-равенства равна числу преобразуемых неравенств

Отметим, наконец, что если переменная x_k НЕ удовлетворяет условию не отрицательности, то ее следует заменить двумя неотрицательными переменными u_k и v_k, приняв $x_k = u_k - v_k$.

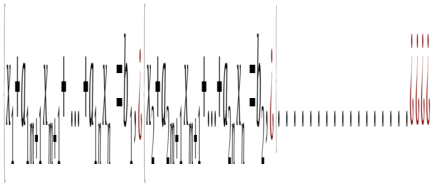
1.4 Симплекс-метод

Суть данного метода заключается в том, что происходит пошаговый переход между различными оптимальными планами. Данный переход является возможным, благодаря уже полученному, начальному плану. Теперь стоит привести задачу, которая покажет, что оптимальное решение можно получить напрямую.

Рассмотрим функцию для которой необходимо найти максимальное значение:

$$F=c_1 x_1+c_2 x_2+\ldots+c_n x_n$$

при следующих ограничениях:



$$x_j \geq 0 \left(j=\overline{1,n}\right).$$

здесь a_{ij}, b_i и $c_j \left(i=\overline{1,m}; j=\overline{1,n}\right)$ - заданные постоянные числа

Векторная форма поставленной задачи будет иметь вид: определить максимум функции

$$F=\sum_{j=1}^{n} c_j x_j \tag{1.8}$$

при условии

$$x_1 P_1+x_2 P_2+\ldots+x_m P_m+\ldots+x_n P_n=P_0 \tag{1.9}$$

$$x_j \geq 0 \left(j=\overline{1,n}\right) \tag{1.10}$$

где

$$P_1 = ¿ \begin{pmatrix} 1 ¿ \\ 0 ¿ \\ ¿ \\ \vdots ¿ \end{pmatrix} ¿ ¿ ¿$$

поскольку

$$b_1 P_1 + b_2 P_2 + \ldots + b_m P_m = P_0,$$

то по определению опорного плана X = (b_1; b_2; …; 0; …; 0) является опорным планом для описанной задачи (последние n – m компонент вектора X равны нулю). Чтобы определить опорный план стоит обратиться к системе векторов, которая составляет базис m-мерного пространства. Каждый из векторов представим своей линейной комбинацией из векторов, входящих в данный базис.

Пусть

$$P_j = \sum_{i=1}^{m} x_{ij} P_i \left(j = \overline{0,n}\right).$$

положим $z_j = \sum_{i=1}^{m} c_i x_{ij} \left(i = \overline{1,n}\right); \Delta_j = z_j - c_j \left(j = \overline{1,n}\right).$ поскольку векторы P_1, P_2, …, P_m - единичные, то $x_{ij} = a_{ij}$ i $z_j = \sum_{i=1}^{m} c_i a_{ij}$, а $\Delta_j = \sum_{i=1}^{m} c_i a_{ij} - c_j$.

Теорема 1.Признак оптимальности опорного плана. Опорный план X` = (x`^{1}; x`$^2$; …; x`^{m}; 0; 0; …; 0) задачи линейного программирования считается оптимальным, если выполняется условие $\Delta_j \geq 0$ для любого j(j = 1, n).

Теорема 2. Целевая функция задачи линейного программирования будет неограниченной, если $\Delta_k < 0$ и для некоторого j = k среди чисел a_{ik}(i = $\overline{1,m}$) нет положительных ($a_{ik} \leq 0$).

Теорема 3. Если опорный план X для задачи линейного программирования является невырожденным и $\Delta_k < 0$, но среди чисел a_{ik} существуют положительные (не все $a_{ik} \leq 0$), то можно найти опорный план X` такой, что F(X`) > F(X).

Доказательство данных теорем можно найти в учебнике Таха.Х.А. (глава 3, ст.95-108)

ГЛАВА 2 ОБЗОР ПРОГРАММНЫХ ПРИЛОЖЕНИЙ ДЛЯ РЕШЕНИЯ ЛИНЕЙНЫХ И НЕЛИНЕЙНЫХ ЗАДАЧ

2.1 Средства оптимизации в Microsoft Excel

Microsoft Excel - это программа выполнения для расчетов и управления так называемыми электронными таблицами.

Под электронной таблицей понимают основной компонент, который используется в обработке, а также анализе численных данных в различных офисных приложениях. Несмотря на прямую связь таблиц с числами и различными операциями, они пригодны для решения задач оптимизации, так как помогают автоматизировать обработку полученных при исследовании данных.

MS Excel может выполнять достаточно сложные математические вычисления, которые используют данные из разных мест таблицы. Чтобы выполнять различного рода расчеты, существует строка в которой вводятся формулы. После введения формулы в определенную ячейку выводится конечный конечный результат. В самом MS Excel организовано большое количество формул, начиная от самых простых математических операций и заканчивая сложнейшими формулами для финансовых расчетов.

Одной из ключевых особенностей MS Excel является то, что если в ячейке изменить данные, то произойдет автоматический пересчет всех показателей, которые фигурируют в таблице, поэтому данный пакет часто используют для учета или контроля чего либо. Также в MS Excel встроена

функция построения графиков и других графических объектов, к примеру диаграмм и гистограмм.

В Excel задачи оптимизации решаются в основном двумя методами:

- Подбор параметра – в таблице находится значение параметра при котором формула дает желаемый результат.
- Поиск решения – это специальная надстройка MS Excel, которая позволяет решать оптимизационные задачи, путем изменения данных в целевых ячейках. Цели бывают разные, это может быть и нахождение максимального и минимального значения целевой функции или же достижение некоторого целевого значения. Проблема решается путем регулировки входных критериев или ограничений, определенных пользователем.

2.2 Методы решения оптимизационных задач в MathCAD

MathCAD – система автоматизированного проектированная, предназначенная для инженерного использования. Достаточно известная система, которая позволяет как и MS Excel проводить сложные расчеты, а также выполнять различные документы, содержащие различные формулы и схемы, а также графические объекты. Данный пакет позволяет более наглядно интерпретировать информацию.

В MathCAD оптимизационные задачи разделяют на два класса:

1. Задачи безусловной оптимизации (без ограничений).

В данной классе используются функции такие, как:

- Minimize(f,<список параметров>) – вычисление точки минимума;
- Maximize(f,<список параметров>) – вычисление точки максимума,

где f – имя минимизируемого функционала, определенного до обращения к функции; <список параметров> - содержит перечисление имен параметров, относительно которых решается оптимизационная задача.

2. Задачи условной оптимизации (с ограничениями).

Для решения задач второго класса также используются функции Minimize и Maximize, но они входят в блок Given и ограничения ставятся в виде равенств или неравенств, определяющие допустимую область значений параметров оптимизации.

MathCAD позволяет выполнять следующие операции:

- вычисления сумм, произведений, рядов, пределов, степеней, корней, производных, интегралов и прочих алгебраических структур;
- работа с комплексными числами, матрицами и векторами;
- решение линейных, нелинейных и дифференциальных уравнений и систем;
- построение графиков и диаграмм;
- проведение статистического анализа и др.

Все производимые вычисления напрямую зависят от поставленных целей пользователя, поэтому можно получать расчеты моментально, а можно сделать программу, которая будет выполнять расчет по команде. Также можно производить различные изменения параметров и переменных, которые участвуют в процессе расчета, таким образом повышая наглядность.

Пользователь может сразу видеть все данные, которые вводит, а следовательно и результат появляется мгновенно, что оптимизирует работу и избавляет от долгих вычислений вручную. MathCAD имеем огромную базу функций и операторов, а также встроенных алгоритмов, которые помогают решать задачи практически любого уровня сложности.

2.3 Библиотеки Python для решения линейных и нелинейных систем алгебраических уравнений.

В языках программирования высокого уровня есть много функций для решения аналитических задач.

Python является одним из самых известных языков программирования для анализа данных. Интерактивная природа этого языка и его научные системные библиотеки делают его предпочтительным для разработки аналитических программ и изучения скрытых фактов в наборах данных [1-2].

Ориентируясь на научное компьютерное сообщество, легко заметить, как увеличивается использование языка Python (начиная с начала 2000 года), в обеих отраслях: создание аналитических приложений и академических исследованиях [3].

Python имеет свою научную экосистему, а также много полезных библиотек:

Numpy - это базовая библиотека для базовых структур данных и основной пакет в языке Python. Зная, что все входные данные в Python представлены как массив numpy; легко сделать вывод, что все библиотеки в этом языке устанавливаются поверх этого пакета [2-3].

SciPy: пакет с коллекцией эффективных алгоритмов для решения задач линейной алгебры, работы с разреженным матричным представлениями, специальные функции и основные статистические функции [2].

2.4 Maple. Система компьютерной алгебры

Maple является системой нового поколения, так как позволяет решать практически любые задачи, которые можно поставить перед наукой. Интерфейс программы позволяет одновременно совершать множество операций, анализировать большие объемы данных.

Основой программы Maple является достаточно сложный и огромный по мощности язык программирования, который включает в себя различные библиотеки по математическим предметам [8-11].

Для решения задач оптимизации используется пакет Optimization (>with(Optimization);), который представлен более модернизированными алгоритмами. С помощью данного пакета можно решать задачи нелинейного программирования, а также квадратичного программирования.

Пакет использует при вычислениях алгоритмы NAG. Также пакет имеет 4 решателя уравнений с заданными ограничениями, реализующих следующие методы:

- LPSolve – линейное программирование;
- LSSolve – улучшенная реализация метода наименьших квадратов;
- QPSolve – квадратичное программирование;
- NLPSolve – нелинейное программирование.

Также для решения оптимизационных задач используется пакет simplex (>with(simplex);), в котором имеется возможность выполнять оценки промежуточных этапов симплексного алгоритма, например, определить базисные переменные.

simplex содержатся встроенные функции такие, как:

- minimize – находит максимум целевой функции;
- maximize – находит минимум целевой функции.

2.5 IBM ILOG CPLEX STUDIO

IBM ILOG CPLEX Optimizer - это инструмент для решения задач линейной оптимизации, обычно называемые проблемами ЛП.

IBM ILOG CPLEX Optimization Studio предлагает быструю разработку и развертывание моделей оптимизации посредством отладки, тестирования, настройки и генерации приложений.

CPLEX также может решать некоторые дополнительные задачи ЛП:

- Сетевые задачи, особый случай задач ЛП, которые CPLEX может решить гораздо быстрее.
- Задачи квадратичного программирования (КП), где целевая функция ЛП расширена добавлением включить квадратичных условий.
- Задачи квадратичного программирования, которые включают квадратичные условия среди ограничений.
- Задачи смешанного целочисленного программирования.

Компоненты CPLEX: CPLEX Interactive Optimizer, Concert Technology, Callable Library. CPLEX поставляется в различных формах для удовлетворения широкого круга потребностей пользователей:

- CPLEX InteractiveOptimizer - это исполняемая программа, которая может читать задачи в интерактивном режиме или из файлов в определенных стандартных форматах, решать задачу, и представлять решение в интерактивном режиме или в текстовом файле. Программа состоит из файла cplex.exe на платформах Windows или cplex для UNIX-платформ.
- ConcertTechnology - это набор библиотек классов C ++, Java и .NET, предлагающих API, которые включают средства моделирования, позволяющие программисту встраивать оптимизаторы CPLEX в приложения C ++, Java или .NET.
- CPLEX CallableLibrary - это библиотека C, которая позволяет программисту встраивать оптимизаторы CPLEX в приложения, написанные на C, VisualBasic, FORTRAN или на любом другом языке, который может вызывать функции C. Библиотека предоставляется в файлах cplexXXX.lib и cplexXXX.dll на платформах Windows, а также в libcplex.a, libcplex.so и libcplex.sl на платформах UNIX.

- Python API для CPLEX - полнофункциональный интерфейс программирования приложений Python, поддерживающий все аспекты оптимизации CPLEX.
- Коннектор CPLEX для MathWorks MATLAB позволяет пользователю определять проблемы оптимизации и решать их в MATLAB, используя либо MATLAB Toolbox, либо класс CPLEX на языке MATLAB.

CPLEX Optimization Studio предоставляет наиболее эффективный способ построения моделей оптимизации. Интегрированная среда разработки (IDE), язык описательного моделирования и алгоритмы позволяют поддерживать весь процесс разработки модели.

CPLEX Optimization Studio включает в себя:

Язык моделирования OPL, который позволяет разрабатывать, отлаживать, тестировать и настраивать модели оптимизации.

DOcplex, слой моделирования Python для CPLEX и CP Optimizer.

Dataandrunmanagemen: разделение модели / данных позволяет модели переключаться между источниками данных, облегчая миграцию на этапах тестирования и развертывания.

Обнаружение и отображение конфликтов и рекомендуемое разрешение конфликтов: обнаружение и отображение конфликтующих ограничений в недопустимых моделях с помощью средства уточнения конфликтов CPLEX или CP Optimizer.

Отслеживание выполнения и профилирование в IDE. Полный доступ к файлам журналов CPLEX и CP Optimizer позволяет отслеживать ход решения. Используйте профилировщик для понимания использования памяти и времени на разных этапах процесса выполнения.

Автоматическая настройка производительности: позволяет использовать инструмент настройки CPLEX для автоматической настройки производительности модели математического программирования.

Интерфейсы прикладного программирования: доступ к оптимизаторам CPLEX Simplex, Barrier, MIP и CP через среду IDE CPLEX Studio, API-интерфейс ConcertTechnology в C ++, Java или Microsoft .NET. Кроме того, вы можете получить доступ к CPLEX Simplex, Barrier и MIP Optimizer через библиотеку CPLEX Callable, Python и MATLAB.

2.6 Сравнительный анализ программ решения оптимизационных задач

На сегодня, имеется тенденция к стремительному росту научно-технического прогресса и в создании вычислительных пакетов тоже. Стоит выделить основные преимущества и недостатки данных программ и сделать на основе полученной информации оптимальный выбор программы, для решения задач оптимизации.

2.6.1 MS Excel

Пожалуй одним из главных преимуществ данной программы является изобилии различных встроенных функций, которые позволяют за сравнительно малые временные затраты осуществить большие по объему вычисления. Под встроенной функцией понимается формула, которая уже заранее подготовлена к работе. Перед тем как вводить формулу в строке указывается знак «=». В MS Excel встроено несколько типов функций, что позволяет решать обширный круг задач и тем самым оптимизируя процесс работы. С помощью таблично заданных значений можно достаточно легко пользоваться статистическими функциями, благодаря которым происходит экономический анализ.

В MS Excel реализуется оперирование понятиями пространства и отношения, которое позволяет посредством задания определенных связей

связать данные в ячейках. То есть программирование как таково не используется, что только упрощает работу с данными.

Что касается недостатков, их несколько:

Во-первых, формулы, которые прописываются в ячейках при обильном их количестве вызывают затруднения при работе. Этот факт значительно затрудняет понимание формул вне контекста, а также затрудняет проверку корректности программы.

Во-вторых, реализация сложных расчетов требует необычайной концентрации, так как одна ошибка может пагубно сказаться на всех предстоящих расчетах, чтобы не допустить этого используют именованные ячейки и другие приспособления, которые помогают не потерять связь между ячейками.

В-третьих, табличные процессоры зависимы от размерности. При действиях с строками и столбцами нужно проделывать большое количество изменений, что требует больших временных затрат.

Данная программа изучается ещё в школе на уроках информатики, поэтому ученики имеют начальное представление о том, как устроена эта программа, с какими объектами нужно работать и соответственно при решении оптимизационных задач детям будет проще ориентироваться, поэтому данная программа является первым кандидатом на решение подобного рода задач.

2.6.2 Mathcad

Рассмотрим основные преимущества системы MathCAD.

1. Естественный математический язык.
2. Наглядность.
3 Хорошая диагностика ошибок.

5. Высокая точность вычислений.

7. Реализация многих стандартных функций вычислительной математики.

8. Возможности символьных математических преобразований.

Что касается недостатков системы Mathcad, то к ним можно отнести следующее:

- ограниченные возможности существующих операторов;

- трудности реализации сложных алгоритмов.

Из рассмотренных пунктов следует, что система MathCAD не позволяет разрабатывать сложные программы, но она очень удобна для тестирования их отдельных частей. Следовательно, основное применение MathCAD - это проведение несложных расчетов и исследований, а также тестирование составных частей больших вычислительных программ, написанных на современных языках высокого уровня, например, на C++ .

2.6.3 Maple

Данная программа позволяет выполнять различного рода исследования и является очень популярной среди ученых. Символьный анализатор это самый мощный компонент данной программы. Что касается структуры программы, то она состоит из ядра, а именно из компонентов, которые созданы на базе языка C, а также имеют достаточно профессиональную оптимизацию, библиотека, которая создана на специальном Maple-языке, также присутствует внешний интерфейс высокого уровня разработки. Основной функцией ядра является обработка и выполнение большого количества операций, библиотека в свою очередь имеет большой арсенал команд, которые выполняются в режиме интерпретации.

К примеру, Maple позволяет создать сложную математическую модель и далее сделать соответствующий код на языке C, который будет отражать работу данной модели.

Таким образом, преимуществами Maple являются: символьные

вычисления, которые позволяют получить результат с высоким уровнем точности, также пользователь получает доступ к различным программам, которых нет в других подобных системах, также в программе реализованы различные специальные функции, которые помогают в сложных вычислениях.

Недостатком Maple является специализированный язык записи операторов и программирования.

В школе данную программу не найти так как указывалось ранее, она популярна в научной среде. Ученику будет сложно овладеть языком, который реализован в Maple, а на обучение может уйти слишком много времени. Данная программа может подойти одаренным ученикам у которых есть стремление к самообразованию, они сами смогу изучить все особенности данной программы. Но в обычном курсе она будет непригодная из-за своей сложности, а также потому, что очень мало где преподается программирование, без которого будет тяжело освоить данную программу.

2.6.4 Python

Бесспорным преимуществом данного языка программирования является то, что его интерпретатор совместим практически с каждой платформой и операционной системой. Первым языком который позволял работать на любой системе был язык C, но его проблема была в том, что на различных устройствах, файлы могли занимать разное количество памяти, что вызывало неудобства при создании переносимой программы. Python решил эту проблему.

Ещё одним преимуществом является возможность расширения языка, так как изначально по задумке автор он должен был быть расширяемым. В свою очередь это означало, что данный язык могут корректировать любые программисты. Интерпретатор создан на базе C, а исходный код дает возможность совершения различного рода действий. Также можно на C

создавать собственные дополнения к Python, затем совершать компиляцию и получать интерпретатор но с уже новыми навыками и возможностями.

Следующим плюсом является присутствие огромного количества модулей, которые имеют разные возможности. Данные модули можно создавать как на базе C, так и на базе Python. Можно привести несколько примеров таких модулей:

- Numerical Python - расширенные математические возможности, такие как манипуляции с целыми векторами и матрицами;
- Tkinter - построение приложений с использованием графического пользовательского интерфейса (GUI) на основе широко распространенного на X-Windows Tk-интерфейса;
- OpenGL - использование обширной библиотеки графического моделирования двух- и трехмерных объектов Open Graphics Library фирмы Silicon Graphics Inc. Данный стандарт поддерживается, в том числе, в таких распространенных операционных системах как Microsoft Windows 95 OSR 2, 98 и Windows NT 4.0.

Единственным недостатком, замеченным автором, является сравнительно невысокая скорость выполнения Python-программы, что обусловлено ее интерпретируемостью.

Как было отмечено ранее мало где в школах изучается программирование, поэтому обучить учеников использования данного продукта является очень сложность задачей, если и вовсе невыполнимой.

2.6.5 IBM ILOG CPLEX STUDIO

Данная система имеет как свои преимущества, так и достаточно заметные недостатки, которые значительно затрудняют работу.

Первым и главным преимуществом является то, что она совместима с очень многими языками программирования, что очень удобно, когда нужно сделать определенную программу на определенном языке, которые сложно реализуем, можно сделать код программы в IBM CPLEX, затем скомпилировать на тот язык, который нужен и в результате получить такую же программу с теми же функциями, но уже на другом языке.

Вторым преимуществом является большое количество опций для различных вычислений, которые представляет данная программа.

Что касается недостатков, это конечно же отсутствие руководства на русском языке, что уже затрудняет изучение данной программы, большое количество алгоритмов просто не позволяет отыскать нужный. Также трудности вызывает создание кода программы, так как стоит учитывать очень много различных условий, которые могут привести к неверному решению.

В школе данная программа не применима, потому что она проигрывает тому же Excel и MathCad по простоте изложения материала, но в научных расчетах, для решения серьезных задач, данная программа может пригодиться.

ГЛАВА 3 ПЕРЕЧЕНЬ ТЕСТОВЫХ ПРИМЕРОВ

Для сравнительного анализа решим данные задачи в каждой из рассмотренных выше программ.

Задача №1. Составить оптимальный производственный план

$$6x_1 + 3x_2 + 6x_3 + 5x_4 + 2x_5 + 3x_6 \rightarrow max$$

$$5x_1 - x_2 + 2x_3 - x_4 + 5x_5 \leq 4$$

$$4x_1 + 6x_3 + 2x_4 + 4x_6 \leq 5$$

$$+ 5x_2 + 4x_3 + 5x_4 + 3x_5 + x_6 \leq 8$$

Все переменные задачи неотрицательны $x_1, x_2 \ldots, x_n \geq 0$.

Ответ $F_{max} = 12.04$. $x_1 = 1.12$; $x_2 = 1.34$; $x_3 = 0$; $x_4 = 0.26$; $x_5 = 0$; $x_6 = 0$

Задача №2. Решить транспортную задачу в матричной постановке

Мощности	**55**	**35**	**35**	**35**	**35**	**55**
55	10	8	6	8	6	7
60	6	6	3	4	6	9
50	8	5	9	8	5	6
60	5	5	5	4	9	9
25	6	3	5	4	6	3

Ответ:

0	0	5.000000	0	20.000000	29.999999
0	0	29.999999	30.000000	0	0
0	35.000000	0	0	14.999999	0
55.000000	0	0	4.999999	0	0
0	0	0	0	0	25.000000

Минимальные транспортные расходы – 1190.

Решение задач оптимизации в MS Excel является самым оптимальным, так как затрачиваемое время на нахождение решения минимальное, алгоритм реализации достаточно просто, по сравнению с другими программами, что только усложняет исходную задачу. Учащимся рамного ближе пакет MS Excel, так как с его основами они знакомятся ещё в средней школе. Решения в других программах больше подходит для университетского курса, так как теория необходимая для их решения расширяется, появляются новые задачи, которые имеют свой специфические условия.

ГЛАВА 4 ОБЗОР РЕШЕНИЯ ЗАДАЧИ СИМПЛЕКС-МЕТОДОМ

4.1 Решение задачи № 1 в MS Excel

Табличное представление представлено на рисунке 5.1.

			Продукция					
	x1	x2	x3	x4	x5	x6		
Объем продукции							**Прибыль**	
Прибыль от реализации	6	3	6	5	2	3	0	
			Ограничения					
								Запас
Ресурсы	A	B	C	E	D	F		
Трудовые	5	-1	2	-1	5	0	0	4
Материальные	4	0	6	2	0	4	0	5
Финансовые	0	5	4	5	3	1	0	8
Нижняя граница выпуска	0	0	0	0	0	0		

Рисунок 5.1 – Табличное представление задачи

После определения модели задачи необходимо настроить параметры Поиска решения (рисунок 5.2). Результаты решения представлены на рисунке 5.3.

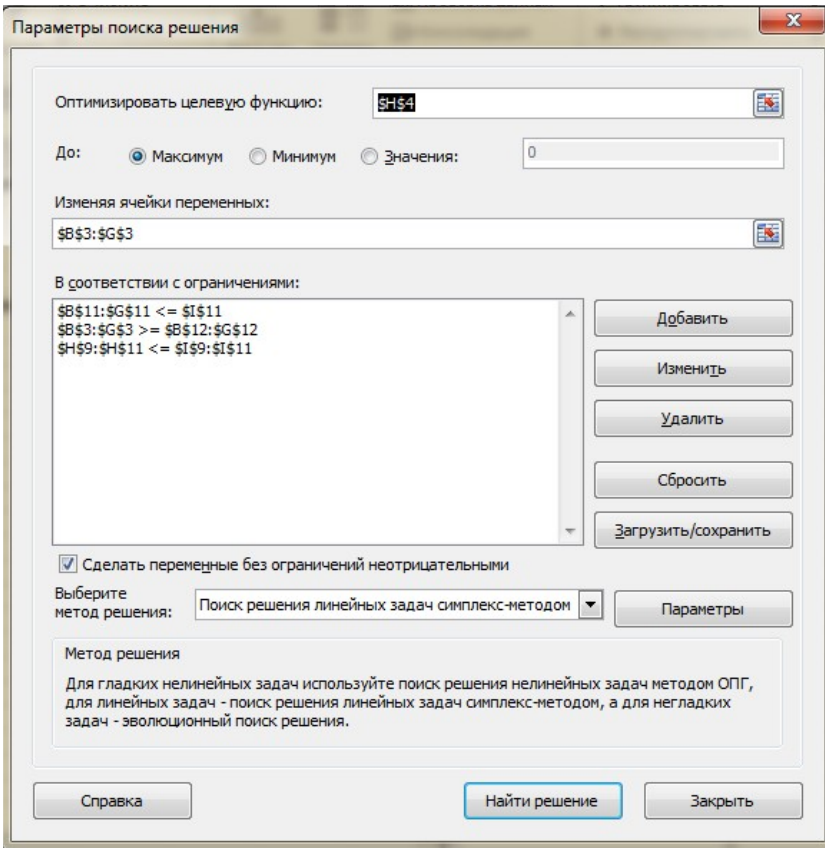


Рисунок 5.2 – Диалоговое окно Параметры поиска решения

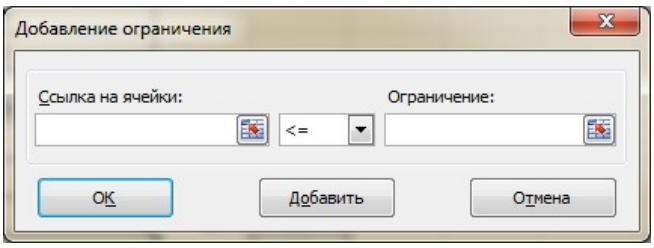


Рисунок 5.3 – Диалоговое окно Добавление ограничений

После нажатия кнопки Найти решение будет найдено оптимальное решение для модели выбора ассортимента продукции.

	Продукция							
	x1	x2	x3	x4	x5	x6		
Объем продукции	1.12	1.34	0	0.26	0	0	**Прибыль**	
Прибыль от реализации	6	3	6	5	2	3	12.04	
	Ограничения							
								Запас
Ресурсы	A	B	C	E	D	F		
Трудовые	5	-1	2	-1	5	0	4	4
Материальные	4	0	6	2	0	4	5	5
Финансовые	0	5	4	5	3	1	8	8
Нижняя граница выпуска	0	0	0	0	0	0		

Рисунок 5.4 – Результат решения

Модель поиска решения не всегда имеет решение, например, когда ограничения на ресурсы не позволяют удовлетворить потребность во всех продуктах. Для удовлетворения потребности необходимо добавить рабочее время или сырье.

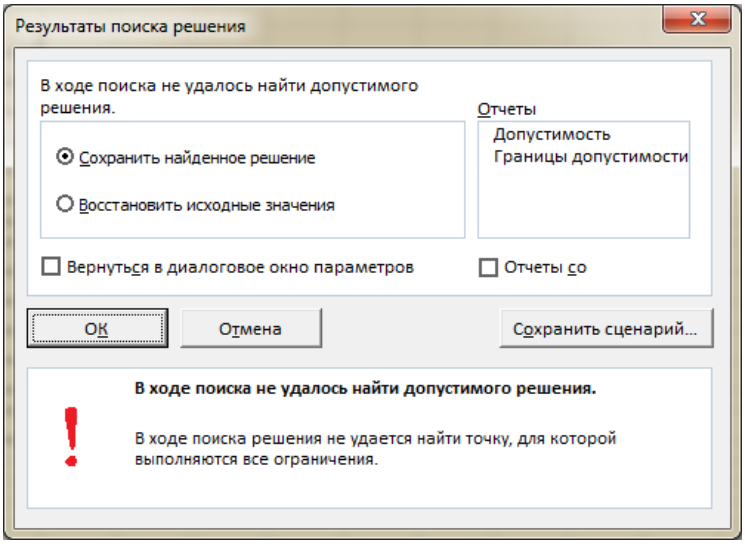


Рисунок 5.5 – Результат решения

Решение задачи происходит с помощью надстройки «Поиск решения» в которой уже есть все встроенные алгоритмы, только нужно вводить значения ячеек, в которых присутствуют данные задачи.

4.2 Решение задачи № 1 в MathCAD

Составляем целевую функцию:

$ORIGIN := 1$

$C := (6 \;\; 3 \;\; 6 \;\; 5 \;\; 2 \;\; 3)^T$

$F(x) := (x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5 \;\; x_6) \cdot C \qquad F(x) \rightarrow 6 \cdot x_1 + 3 \cdot x_2 + 6 \cdot x_3 + 5 \cdot x_4 + 2 \cdot x_5 + 3 \cdot x_6$

Матрица коэффициентов системы неравенств и столбец правых частей:

Начальное приближение:

Решаем с помощью блока Given, записываем систему ограничений $Ax \le b$:

Given

$A \cdot X \le b$

$X \ge 0$

$X := Maximize(F, X) \qquad X = \begin{pmatrix} 1.12 \\ 1.34 \\ 0 \\ 0.26 \\ 0 \\ 0 \end{pmatrix}$

$F(X) = 12.04$

Таким образом, получено решение

$$x_1 = 1.12;\; x_2 = 1.34;\; x_3 = 0;\; x_4 = 0.26;\; x_5 = 0;\; x_6 = 0$$

Максимальное значение целевой функции $F_{max} = 12.04$.

При решении задачи в Mathcad используется следующий алгоритм:

1) Специальной переменной ORIGIN присваивают значение 1. Значением ORIGIN является номер первого элемента строки или столбца в матрице. По умолчанию ORIGIN=0. В меню Math выбрать строку Options или ORIGIN.

2) Вводят исходные данные задачи в матричной форме.

3) Вводят линейную целевую функцию.

4) Задают начальные значения переменным задачи.

5) Вводят ограничения задачи в матричной форме.

6) Определяют оптимальное решение задачи с помощью встроенной функции Maximize (в случае поиска максимума функции) или Minimize (в случае поиска минимума функции).

4.3 Решение задачи № 1 в Python

Задачу на поиск оптимального решения будем искать симплекс-методом

```
#подключение библиотек
importnumpy as np
#симплекс-таблица 5*11.
st=np.array([[0,0,1,2,3,4,5,6,7,8,9],
        [7,4.,5.,-1.,2.,-1.,5.,0.,1.,0.,0.],
        [8,5.,4.,0.,6.,2.,0.,4.,0.,1.,0.],
        [9,8.,0.,5.,4.,5.,3.,1.,0.,0.,1.],
        [0,0.,-6.,-3.,-6.,-5.,-2.,-3.,0.,0.,0.]])

defsimplex_method(st):
l=0
```

```
whilemin(st[4])<0:
#Ищем минимальный элемент в последней строке
        n=np.argmin(st[4])
        u=np.array([-1.0,-1.0,-1.0,-1.0,-1.0])
        for i in range(1,4):
ifst[i,n]>0:
u[i]=st[i,1]/st[i,n]
minba=max(u)
iba=0
  #номер строки, в которой находится наше классное ai, которое удовлетворяет нужным условиям
for i in range(1,4):
if u[i]>0 and u[i]<=minba:
minba=u[i]
iba=i
#нашли нужную строчку, теперь будем строить новую симплекс-таблицу
#Делим ведущую строку на ведущий элемент
a=st[iba,n]
for j in range(1,11):
st[iba,j]=st[iba,j]/a
for i in range(1,5):
a=st[i,n]
for j in range(1,11):
if i<iba or i>iba:
st[i,j]=st[i,j]-st[iba,j]*a
#Заменяем номер старого базисного элемента на новый в нулевом столбце
st[iba,0]=st[0,n]
#Запишемтеперьответ
```

```
x=[0,0,0,0,0,0]
for i in range(1,4):
x[int(st[i,0])]=st[i,1]
print('Производственный план:',x)
print('Максимальное значение целевой функции:',st[4,1])
return 0
simplex_method(st)
```

4.4 Решение задачи № 1 в Maple

На рисунке 5.7 представлено решение задачи № 1 в Maple.

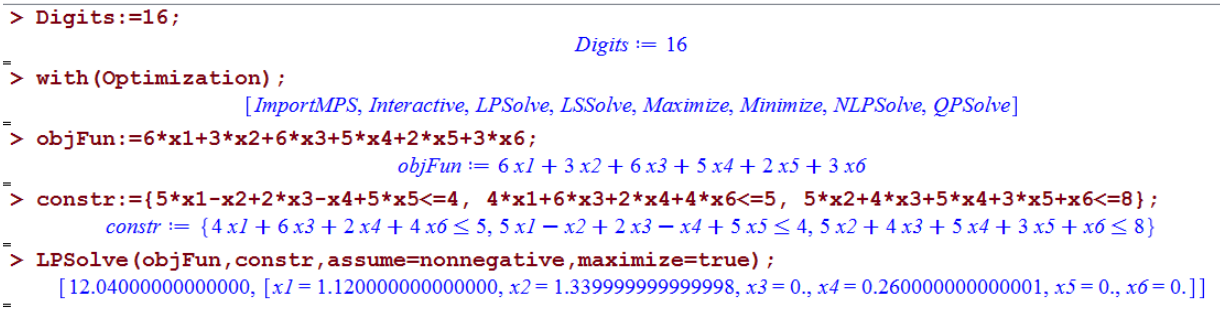


Рисунок 5.7 Решение задачи № 2

Как можно заметить, снова имеют преимущества пакеты MathCad и Maple, так как алгоритмы строятся достаточно компактно и быстро, шанс ошибки достаточно мал, по сравнению с тем Python, в котором одна ошибка приводит к отсутствию запуска программы. Такая же ситуация с MS Excel.

4.5 Решение задачи № 1 в IBM ILOG CPLEX STUDIO

CPLEX> enter example

Enter new problem ['end' on a separate line terminates]:

maximize 6x1 + 3x2 + 6x3 + 5x4 + 2x5 + 3x6

subject to 5x1 - x2 + 2x3 – x4 + 5x5 <= 4

4x1 + 6x3 + 2x4 + 4x6 <=5

5x2 + 4x3 + 5x4 + 3x5 + x6 <=8

bounds

0 <= x1

0 <= x2

0 <= x3

end

CPLEX> optimize

Tried aggregator 1 time.

No LP presolve or aggregator reductions.

Presolve time = 0.00 sec. (0.00 ticks)

Iteration log . . .

Iteration: 1 Dual infeasibility = 0.000000

Iteration: 2 Dual objective = 202.500000

Dual simplex - Optimal: Objective = 2.0250000000e+002

Solution time = 0.01 sec. Iterations = 2 (1)

Deterministic time = 0.00 ticks (3.38 ticks/sec)

CPLEX> display solution variables x1-x6

Variable Name	Solution Value
x1	1.120000
x2	1.340000
x3	0.000000
x4	0.260000
x5	0.000000
x6	0.000000
F	12.07000

```
CPLEX> quit
```

Стоит отметить, что задача линейного программирования достаточно просто решается с помощью MS Excel, MathCad, Maple, это обусловлено тем, что эти программы обладают набором необходимых команд и встроенных функций, которые помогают оптимизировать процесс, в отличии от Python и Cplex, так как они требуют написания кода программы, что уже может являться препятствием для решения задач большей размерности, поэтому целесообразно использовать на выбор, либо MS Excel, либо MathCad, либо Maple.

ГЛАВА 5 ОБЗОР РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ

5.1 Решение задачи № 2 в MS Excel

Рассмотрим пример решения транспортной задачи в матричной постановки в MSExcel. Исходные данные представлены в таблице 6.1

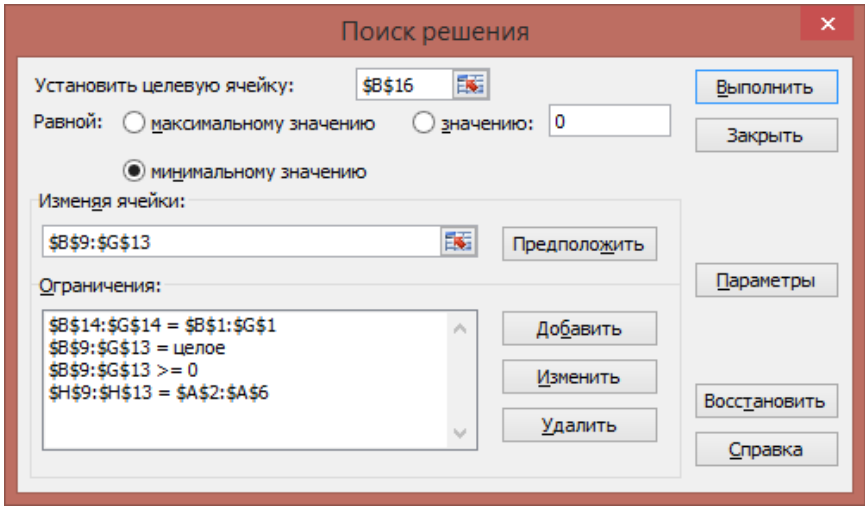
	A	B	C	D	E	F	G
1	Мощности	**55**	**35**	**35**	**35**	**35**	**55**
2	**55**	10	8	6	8	6	7
3	**60**	6	6	3	4	6	9
4	**50**	8	5	9	8	5	6
5	**60**	5	5	5	4	9	9
6	**25**	6	3	5	4	6	3

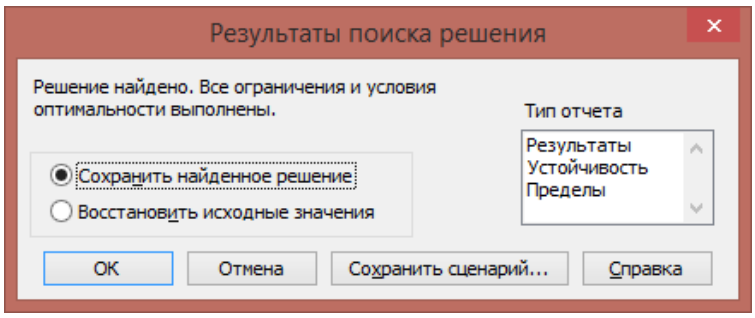
Рисунок 6.1 Исходные данные задачи

Вводим формулы, необходимые для решения:

	A	B	C	D	E	F	G	H
1	Мощности	**55**	**35**	**35**	**35**	**35**	**55**	**=СУММ(B1:G1)**
2	**55**	10	8	6	8	6	7	
3	**60**	6	6	3	4	6	9	
4	**50**	8	5	9	8	5	6	
5	**60**	5	5	5	4	9	9	
6	**25**	6	3	5	4	6	3	
7	**=СУММ(A2:A6)**							
8								
9	**неизвестные**							=СУММ(B9:G9)
10								=СУММ(B10:G10)
11								=СУММ(B11:G11)
12								=СУММ(B12:G12)
13								=СУММ(B13:G13)
14		=СУММ(B9:B13)	=СУММ(C9:C13)	=СУММ(D9:D13)	=СУММ(E9:E13)	=СУММ(F9:F13)	=СУММ(G9:G13)	
15								
16	**целевая функция**	=СУММПРОИЗВ(B2:G6;B9:G13)						

Заполняем «Поиск решения»:





Полученный результат:

	A	B	C	D	E	F	G	H
1	Мощности	**55**	**35**	**35**	**35**	**35**	**55**	**250**
2	**55**	10	8	6	8	6	7	
3	**60**	6	6	3	4	6	9	
4	**50**	8	5	9	8	5	6	
5	**60**	5	5	5	4	9	9	
6	**25**	6	3	5	4	6	3	
7	**250**							
8								
9	**неизвестные**	0	0	5	0	34	16	55
10		0	0	30	30	0	0	60
11		0	35	0	0	1	14	50
12		55	0	0	5	0	0	60
13		0	0	0	0	0	25	25
14		55	35	35	35	35	55	
15								
16	**целевая функция**	1190						

MS Excel дает возможность решать задачи поиска корней с заданной точностью, но для поиска всех корней необходимы дополнительные исследования, связанные с построением графика функции. При этом может возникнуть ситуация при которой график функции построить не представляет возможности. Оптимизационные задачи решаются при помощи надстройки Поиск решения, но при этом требуют дополнительных построений и вычислений для создания необходимых исходных данных.

5.2 Решение задачи № 2 в MathCAD

На рисунке 6.6 представлено решение задачи № 2 в MathCAD.

$$C := \begin{pmatrix} 10 & 8 & 6 & 8 & 6 & 7 \\ 6 & 6 & 3 & 4 & 6 & 9 \\ 8 & 5 & 9 & 8 & 5 & 6 \\ 5 & 5 & 5 & 4 & 9 & 9 \\ 6 & 3 & 5 & 4 & 6 & 3 \end{pmatrix}$$ - матрица тарифов

$a := (55 \quad 60 \quad 50 \quad 60 \quad 25)^T$ - запасы поставщиков

$b := (55 \quad 35 \quad 35 \quad 35 \quad 35 \quad 55)^T$ - потребности потребителей

$i := 1..5 \qquad j := 1..6$

$\sum_i a_i = 250 \qquad \sum_j b_j = 250$ - задача закрытая (суммарные потребности равны суммарным затратам)

$Z(x) := \sum_{i=1}^{5} \sum_{j=1}^{6} \left(x_{i,j} \cdot C_{i,j}\right)$ - затраты на перевозку (целевая функция)

$x_{i,j} := 0$ - начальное приближение

$t_i := 1 \qquad r_j := 1$

Given

$x \cdot r = a$ - все грузы вывезены

$x^T \cdot t = b$ - все потребности удовлетворены

$x \geq 0$

$x := \text{Minimize}(Z, x) \qquad x = \begin{pmatrix} 0 & 0 & 5 & 0 & 20 & 30 \\ 0 & 0 & 30 & 30 & 0 & 0 \\ 0 & 35 & 0 & 0 & 15 & 0 \\ 55 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 \end{pmatrix}$ - оптимальный план перевозок

$Z(x) = 1190$ - минимальные транспортные расходы

Рисунок 6.6 Решение задачи № 2

Таким образом, получен оптимальный план перевозок:

- Из пункта 1 надо доставить 5 ед. груза потребителю 3, 20 ед. потребителю 5 и 30 ед. потребителю 6.
- Из пункта 2 надо доставить 30 ед. груза потребителю 3 и 30 ед. потребителю 4.

- Из пункта 3 надо доставить 35 ед. груза потребителю 2 и 15 ед. потребителю 5.
- Из пункта 4 надо доставить 55 ед. груза потребителю 1 и 5 ед. потребителю 4.
- Из пункта 5 надо доставить 25 ед. груза потребителю 6.

Минимальные расходы на перевозку составили 1190 ден. ед.

5.3 Решение задачи №2 на Python

```
import numpy as np
t1=np.array([[0,55,35,35,35,35,55],
             [55,10,8,6,8,6,7],
             [60,6,6,3,4,6,9],
             [50,8,5,9,8,5,6],
             [60,5,5,5,4,9,9],
             [25,6,3,5,4,6,3]])
t2=np.array([[0,5,15,15,10],
              [15,11,0,20,11],
              [25,12,7,9,20],
              [5,0,14,16,18]])

def Transport(t,m,n):
    #На всякий случай проверяем условие баланса
    sum_a=0
    sum_b=0
    for j in range(1,n):
        sum_a=sum_a+t[0,j]
    for i in range(1,m):
        sum_b=sum_b+t[i,0]
```

```
if sum_a==sum_b:
    x=np.zeros((m,n)) #наш будущий план перевозок
#Найдём сначала допустимое решение с помощью метода северо-западного угла
    i=1
    j=1
    while i<m:
        while j<n:
            x[i,j]=min(t[0,j],t[i,0])
            t[0,j]=t[0,j]-x[i,j]
            t[i,0]=t[i,0]-x[i,j]
            if t[0,j]==0:
                for k in range(i+1,m):
                    x[k,j]=0
                j=j+1
            if t[i,0]==0:
                for l in range(j+1,n):
                    x[i,l]=0
                i=i+1
#Теперь будем искать оптимальный план перевозок с помощью метода потенциалов

    #Построим систему потенциалов, сначала найдём кол-во уравнений системы
    q=0
    for i in range(1,m):
        for j in range(1,n):
            if x[i,j]>0:
                q=q+1
```

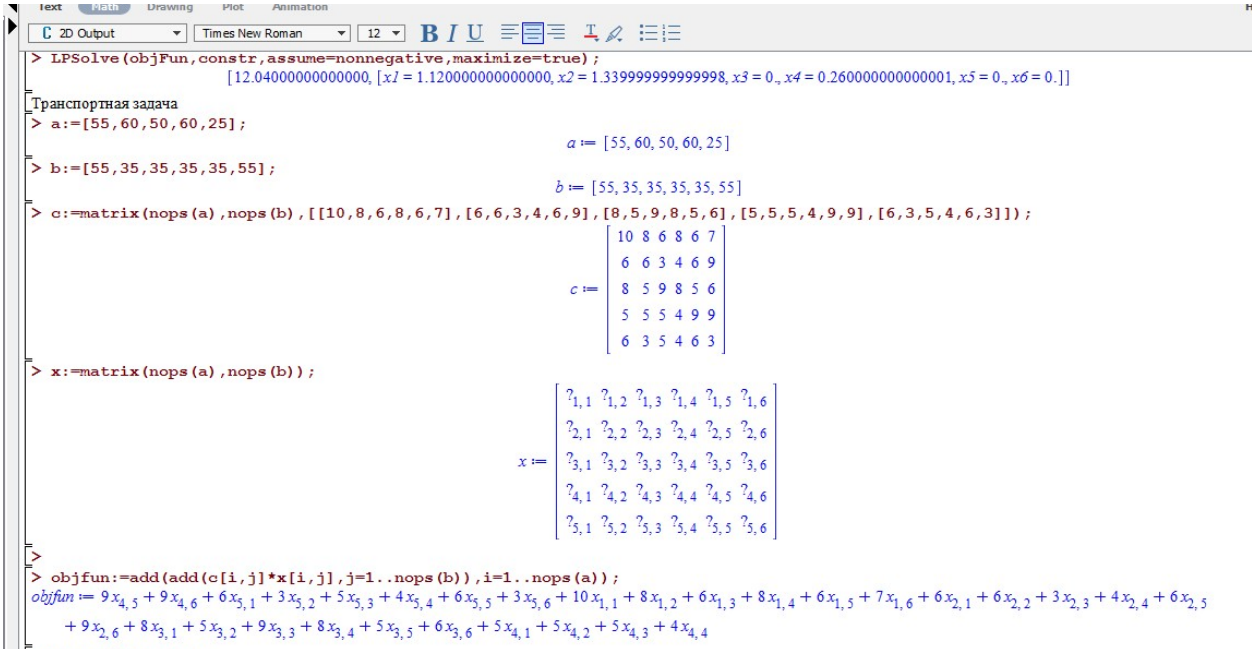
```
#Создадим матрицу, в которую позже внесем коэффициенты системы
#Размер матрицы q*(m+n)
sp=np.zeros((q,m+n-2))
spb=np.zeros(q) #вектор свободных членов
#Теперь вносим коэффициенты
k=0
for i in range(1,m):
    for j in range(1,n):
        if x[i,j]>0:
            spb[k]=x[i,j]
            sp[k,m-2+j]= 1
            sp[k,i-1]=-1
            k=k+1
#Надо сделать матрицу квадратной, чтобы у системы было единственное решение
#Положим первые две переменные равными нулю, просто уберем первые два столбца
sp1=sp[:,2:m+n-2]

#Решим систему линейных уравнений
uv=np.linalg.solve(sp1,spb)
print(uv)
u=[0,0,uv[0],uv[1],uv[2]]
v=uv[m-3:m+n-1]
print(u,v)
print(spb)

#Строим матрицу невязок
c=np.zeros((m,n))
```

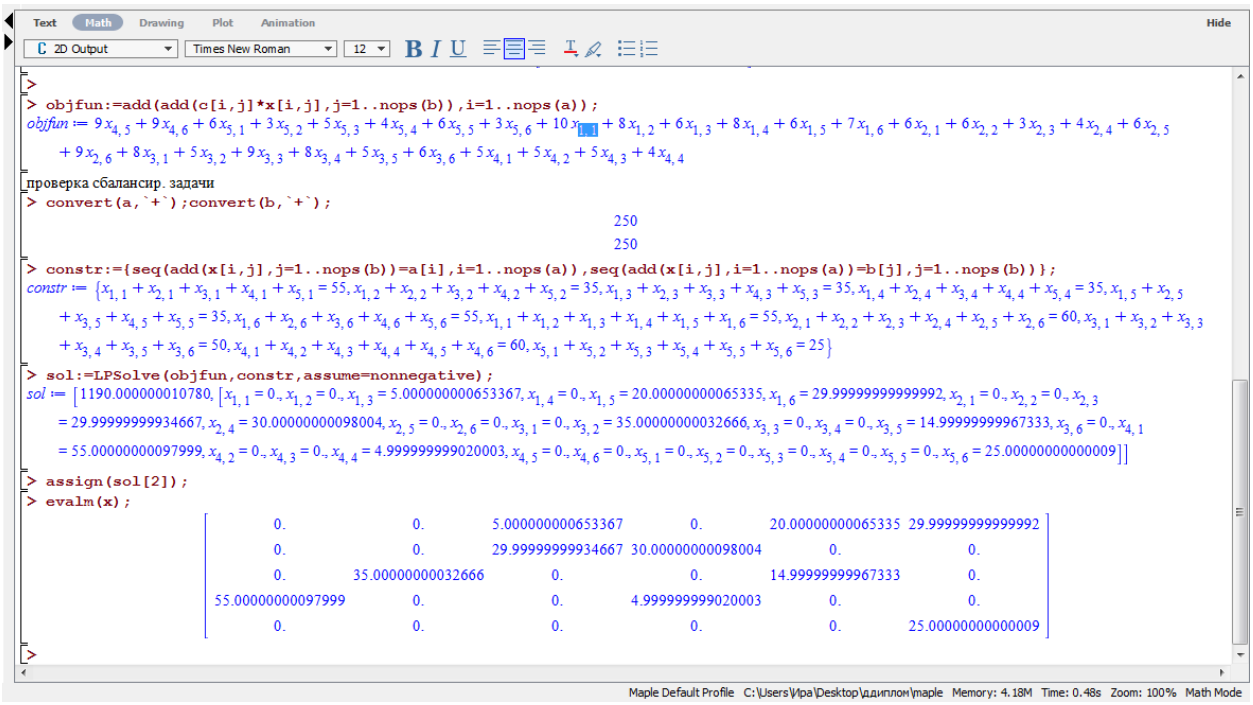
```
            for i in range(1,m):
                for j in range(1,n):
                    c[i,j]=t[i,j]-(v[j-1]-u[i-1])
            print(x)
        return 0
    Transport(t1,6,7)
```

Результат:

```
[ 15. -10.  -5.  55.  35.  25.  50.  20.  20.]
[0, 0, 15.0, -10.0, -5.0] [55. 35. 25. 50. 20. 20.]
[55. 35. 25. 10. 35.  5. 30. 30. 25.]
[[ 0.  0.  0.  0.  0.  0.  0.]
 [ 0. 55.  0.  0.  0.  0.  0.]
 [ 0.  0. 35. 25.  0.  0.  0.]
 [ 0.  0.  0. 10. 35.  5.  0.]
 [ 0.  0.  0.  0.  0. 30. 30.]
 [ 0.  0.  0.  0.  0.  0. 25.]]
```

5.4 Решение задачи № 2 в Maple





1. Решить первую транспортную задачу, найдя цену перевозки и план перевозки.

2. Решить вторую транспортную задачу (с другими ценами), найдя цену перевозки и план перевозки.

f1x/(f1x+f2x) f2x/(f1x+f2x)
f1y/(f1y+f2y) f2y/(f1y+f2y)

3. Составляем матрицу M=, где f1x - это план перевозки первой задачи, поставленный в функцию цели первой задачи; f2x - это план перевозки первой задачи поставленный, в функцию цели второй задачи; f1y - это план перевозки второй задачи, поставленный в функцию цели первой задачи; f2y - это план перевозки второй задачи, поставленный в функцию цели второй задачи.

4. Матрицу M транспонируем. Полученную матрицу обозначаем A.

5. Находим собственные значения и собственные векторы матрицы A.

6. Берем один из столбцов матрицы собственных векторов, в

зависимости конечно от собственных значений (если собственное значение равное единице стоит первым, то берем первый столбец, иначе второй) и нормируем этот столбец.

7. Первую координату нормированного вектора умножаем на первую матрицу цен и складываем с второй матрицей цен умноженное на вторую координату полученного нормированного вектора. Получили новую матрицу цен.

8. Решаем новую транспортную задачу с таким же объемом поставок и потребностей, но с новыми ценами, полученными в пункте 7.

9. Новый план перевозок ставим в функцию цели первой и второй транспортной задачи и таким образом получаем УПРАВЛЕНИЯ. На содержательном уровне нашли оптимальное управление, что одновременно минимизирует стоимость и время перевозки.

Наиболее простой реализацией транспортной задачи является реализация в Excel так как она происходит в надстройке «Поиск решения» которая не является сложной для усвоения, так же время затрачиваемое на решение является достаточно малым, в отличии от реализации на Python или Maple, так как количество элементов кода или программы достаточно велико и нужно быть очень внимательным при создании. Решение транспортной задачи в Mathcad является промежуточным вариантом между MS Excel и Maple, так как Mathcad более прост в использовании.

Заключение

В данной работе было рассмотрено большое количество оптимизационных задач, которые можно решать даже ученикам в школе, а также показана их программная реализация. В первой главе были рассмотрены основные теоретические сведения об оптимизационных задач, приведены основные методы их решения, а также показана их работа на конкретных примерах.

Во второй главе проведен сравнительных анализ различных программ и пакетов для реализации решения оптимизационных задач, рассмотрены основные преимущества и недостатки каждой программы.

В третьей главе рассмотрены примеры решения задач в различных программах, показаны отличительные особенности решения одной задачи в различными программными средствами.

Для учащихся в школе подойдут такие программы как MS Excel, MathCad и Maple, так как в них достаточно нетрудно разобраться, тем более что программа MS Excel входит в школьный курс информатики и ученики уже имеют представление о том, как с ней работать. Плюсом данных пакетов является сравнительно простая реализация методов решения оптимизационных задач, что позволяет в максимально короткие сроки решить поставленную задачу, не прибегая к сложным математическим вычислениям.