МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Тульский государственный университет»

Институт прикладной математики и компьютерных наук
Кафедра «Прикладная математика и информатика»

Утверждено на заседании кафедры
«Прикладная математика и информатика»
24 января 2022 г., протокол № 5

Заведующий кафедрой
________________________М.В. Грязев

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению практических (семинарских) занятий
по дисциплине (модулю)
«Исследование операций»

основной профессиональной образовательной программы
высшего образования – программы бакалавриата

по направлению подготовки
01.03.02 Прикладная математика и информатика

с направленностью (профилем)
Прикладная математика и информатика

Форма обучения: очная

Идентификационный номер образовательной программы: 010302-01-22

Тула 2022 год

Разработчик методических указаний

Рудомазина Ю.Д., доцент каф. ПМиИ, к.ф.-м.н.

(ФИО, должность, ученая степень, ученое звание) *(подпись)*

На практических занятиях по курсу «Исследование операций» обсуждаются вопросы, закрепляющие и расширяющие материал, изложенный на лекциях. Рассматриваются некоторые расчетные задачи по решению практических задач из соответствующих разделов учебного курса.

Решение типовых задач курса базируется на учебных пособиях:

1. Воробьёв С.А. **Теория игр и исследование операций.**: Учеб. пособие. Тула, Изд–во ТулГУ. 2012. – 100 с.

2. Воробьёв С.А. **Модели и методы исследования операций**.: Учеб. пособие. Тула, Изд–во ТулГУ. 2007. – 148 с.

В эти пособиях приведены решения практически всех основных изучаемых вопросов. В конце каждой главы предлагаются индивидуальные задачи для каждого студента.

Дополнительно в качестве самостоятельных (аттестационных) предлагаются типовые задания с индивидуальными исходными данными.

Эти задания включают в себя следующие темы:

1. Построение моделей, приводящих к задачам линейного программирования
2. Построение моделей, приводящих к транспортным задачам
3. Решение задач целочисленного программирования методом отсечений
4. Решение задач целочисленного программирования методом ветвей и границ
5. Решение задач нелинейного программирования методом допустимых направлений
6. Построение Марковских сетевых моделей
7. Построение системы дифференциальных уравнений Колмогорова. Понятие предельных вероятностей состояний
8. Построение модели и определение основных характеристик СМО
9. Нахождение оптимальных решений матричных игр
10. Поиск точек равновесия по Нэшу, максиминных стратегий и стратегий угроз
11. Поиск оптимума по Парето
12. Нахождение решения арбитра и соответствующих ему оптимальных наборов частот

СОДЕРЖАНИЕ

Практическое занятие № 1.

«Построение моделей, приводящих к задачам линейного программирования»

Вот несколько типичных задач, приводящихся к задачам линейного программирования:

⇒ Есть предприятие, которое может выпускать продукцию нескольких различных типов. Для производства единицы каждого из них требуется определённое количество сырья и прочих ресурсов, запасы которых ограничены. Известна рыночная стоимость этих товаров и ограничения на возможный объём продажи. Требуется так составить план производства, чтобы получить максимально возможную прибыль.

⇒ Известно количество белков, жиров, углеводов и прочих веществ, необходимых человеку в течении суток. Дан список продуктов, для каждого из которых указано содержание всех этих веществ. Требуется составить суточный рацион минимальной массы (или стоимости).

⇒ Есть группа предприятий, размещенных в разных местах. Им требуется сырье, источники которого (склады) тоже находятся в разных местах. Есть система транспортного сообщения (тарифы на перевозку тоже разные). Требуется таким образом разместить заказы на сырьё, чтобы обеспечить заводы сырьем при минимуме расходов.

Эти задачи необходимо записать в виде математических моделей. Рассмотрим этот процесс на примере первой из них. Введём $\boldsymbol{n}$ переменных $\boldsymbol{x_1,x_2,...,x_n}$ – количество продукции каждого вида, что в совокупности образует план выпуска. Очевидно, эти переменные удовлетворяют условиям $\boldsymbol{x_1\geq 0}$, $\boldsymbol{x_2\geq 0}$, ..., $\boldsymbol{x_n\geq 0}$ – невозможно выпустить отрицательное количество продукции. Пусть имеется $\boldsymbol{m}$ ограничений по ресурсам. Обозначим максимальное доступное количество каждого из них как $\boldsymbol{b_1,b_2,...,b_m}$. Введём коэффициенты расхода этих ресурсов. $\boldsymbol{a_{ij}}$ показывает, какое количество ресурса вида $\boldsymbol{i}$ пойдёт на изготовление единицы продукции вида $\boldsymbol{j}$. Для определения наилучшего плана используем величины $\boldsymbol{c_1,c_2,...,c_n}$ – прибыль, получаемую от реализации единицы продукции каждого вида. Общая прибыль при реализации конкретного плана составит величину $\boldsymbol{L=c_1x_1+c_2x_2+...+c_nx_n}$. Очевидно, что оптимальным решением задачи будет набор переменных, доставляющих максимальное значение этой функции. Кроме того, этот набор должен удовлетворять ограничениям

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n \leq b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + ... + a_{mn}x_n \leq b_m \end{cases}$$

т.е. суммарный расход любого ресурса не должен превосходить его имеющихся запасов.

Для удобства методы, применяемые для решения какого-либо класса задач исследования операций, обычно разрабатываются для задачи некоторого стандартного вида, к которой в свою очередь можно свести любую задачу данного класса. Соответственно и для линейного программирования формулируется ос-

новная задача, сокращённо ОЗЛП. Её главная особенность заключается в том, что все ограничения записываются в виде равенств. Позже будет показано, что и задачи с ограничениями–неравенствами могут быть сведены к ОЗЛП.
Итак, имеется ряд переменных $\boldsymbol{x_1, x_2, \ldots, x_n}$. Требуется найти такие неотрицательные значения этих переменных, которые удовлетворяли бы системе линейных уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n = b_n \end{cases} \quad (1)$$

и, кроме того, обращали бы в минимум линейную функцию

$\boldsymbol{L = c_1x_1 + c_2x_2 + \ldots + c_nx_n}$. (2)
Условимся считать ***допустимым решением*** ОЗЛП любую совокупность переменных

$\boldsymbol{x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0}$. (3)
удовлетворяющую уравнениям (1). ***Оптимальным*** решением будем называть то из допустимых решений, при котором линейная функция (2) обращается в минимум.
Заметим, что далеко не все авторы приводят ОЗЛП именно в таком виде. Существуют различные взгляды на то, как оно должно выглядеть. Например, часто “стандартным” считают поиск максимума функции (2), бывают и иные отличия. Но, зная методы решения для любой из “стандартных форм” задачи, не представляет труда найти решение для любой другой формы.
ОЗЛП не обязательно имеет решение. Возможно, уравнения в системе (1) противоречат друг другу, или же они имеют решение только вне области неотрицательных значений переменных, что противоречит условию (3), или же решения существуют, но среди них нет оптимального. Последний случай возможен, если область допустимых решений (ОДР) не замкнута в сторону уменьшения целевой функции – в результате чего для любого решения задачи можно найти другое решение с ещё меньшим значением этой функции, т.е. решения, лучшего всех остальных, не может быть в принципе.

Индивидуальные задания по теме № 1.

«Построение моделей, приводящих к задачам линейного программирования»

Каждому студенту выдаются условия задачи: матрица расходов сырья на выпуск продукции, вектор запасов сырья и показатели дохода от его продажи.
Требуется:
1 – составить задачу линейного программирования по определению оптимального плана выпуска продукции;
2 – привести эту задачу к стандартному виду.

Практическое занятие № 2.

«Построение моделей, приводящих к транспортным задачам»

Как уже говорилось ранее, любая задача линейного программирования может быть приведена к стандартному виду и решена симплекс-методом. Однако не каждую задачу имеет смысл решать всегда именно таким способом. Некоторые задачи имеют специальный вид, позволяющий решить их более быстродействующими методами.

В частности, к таким относится транспортная задача. Она описана в начале второй главы – третий по порядку пример. Это безусловно задача линейного программирования, но она имеет достаточно специфический вид:

В $\boldsymbol{m}$ пунктах отправления имеются запасы некоторого груза в количестве $\boldsymbol{a_1}$, $\boldsymbol{a_2}$, ..., $\boldsymbol{a_m}$ единиц соответственно. Этот груз требуется доставить в $\boldsymbol{n}$ пунктов назначения, в каждый из пунктов – соответственно $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, ..., $\boldsymbol{b_n}$ единиц груза. $\boldsymbol{a_1}$, $\boldsymbol{a_2}$, ..., $\boldsymbol{a_m}$ называются запасами, а $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, ..., $\boldsymbol{b_n}$ – заявками. В классической постановке задачи сумма заявок равна сумме всех запасов

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \quad (1)$$

Если это равенство не выполняется, то применяются методы приведения задач к стандартной форме. Они рассмотрены ниже.

Также дана стоимость перевозки единицы товара с каждого склада каждому потребителю (физический смысл данной величины может быть не стоимостью, а расстоянием или временем перевозки, это для рассмотрения общих методов решения задачи непринципиально). Она задана матрицей

$$\left\{\begin{matrix} c_{11} & c_{12} & \ldots\ldots & c_{1n} \\ c_{21} & c_{22} & \ldots\ldots & c_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ c_{m1} & c_{m2} & \ldots\ldots & c_{mn} \end{matrix}\right\}$$

Необходимо найти такой план перевозок, чтобы выполнить все заявки и при этом минимизировать общую (суммарную) стоимость всех перевозок.

Решение задачи представляет собой набор переменных $\boldsymbol{x_{ij}}$ – количество груза, отправляемого из $\boldsymbol{i}$–ого пункта отправления в $\boldsymbol{j}$–ый пункт назначения – всего $\boldsymbol{m \times n}$ переменных, и все они *неотрицательны*.

На переменные $\boldsymbol{x_{ij}}$ наложены следующие условия:

Суммарное количество груза, выходящего из каждого пункта отправления, равно запасу груза в этом пункте. Всего получаем $\boldsymbol{m}$ ограничений–равенств:

$$\sum_{i=1}^{n} x_{ki} = a_k, \quad k = 1,\ldots,m \quad (2)$$

Суммарное количество груза, приходящего на каждый пункт назначения, равно заявленному этим пунктом количеству. Всего получаем $\boldsymbol{n}$ ограничений–равенств:

$$\sum_{j=1}^{m} x_{jl} = b_l, \quad l = 1,\ldots,n \quad (3)$$

Целевая функция, которую надо минимизировать – стоимость всех перевозок

$$L = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ji}x_{ji} \Rightarrow \min \quad (4)$$

Это типичная задача линейного программирования, поскольку все ограничения и целевая функция линейны.

Но у данной задачи есть ряд особенностей. Главное из них – это то, что коэффициенты при всех переменных в условиях (2) и (3) равны **1** (единице). Кроме того, одно из уравнений в этой системе не является независимым и может быть пропущено. Всего получается свободных переменных $(n\text{-}1)(m\text{-}1)$. Именно такое количество значений x_{ij} должно обращаться в нуль в оптимальном плане (или больше в случае вырожденного плана, но не меньше). Эти переменные будем называть перевозками, и по прежнему мы используем понятия *допустимого* плана, *опорного* и *оптимального*.

Для поиска оптимального плана не требуется применять симплекс-метод, все манипуляции выполняем непосредственно в таблице, называемой транспортной таблицей. В неё записаны все условия задачи: пункты отправления и назначения, запасы и заявки, стоимость перевозок.

	B_1	B_2	……	B_n	Запасы a_i
A_1	c_{11}	c_{12}	……	c_{1n}	a_1
A_2	c_{21}	c_{22}	……	c_{1n}	a_2
……	……	……	……	……	……
A_m	c_{m1}	c_{m1}	……	c_{mn}	a_i
Заявки b_j	b_1	b_2	……	b_n	$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$

Стоимость перевозки единицы товара помещаем в каждой клетке вверху справа. Те ячейки, в которые помещаем отличные от нуля значения перевозок, называем базисными, а остальные – свободными. При этом сумма этих элементов плана перевозок x_{ij} в каждой строке равна a_i, а в каждом столбце – b_j. Ячейку, содержащую x_{ij}, будем обозначать координатами (i,j). Заметим, что в отличие от ОЗЛП, решение ТЗ *всегда существует*.

В ячейках основной части транспортной таблицы используются все 4 угла – в каждый из них помещается какое-нибудь значение, используемое при нахождении оптимального плана перевозок.

Нахождение опорного плана. Самый простой способ построения опорного плана – “метод северо-западного угла”. Этот “научный термин” по сути означает, что двигаться мы будем слева направо и сверху вниз. Действительно, если представить основную часть транспортной таблицы в виде географической карты, то ячейка (1,1) будет её северо-западным углом, а ячейка (***m***,***n***) – её юго-восточным углом. Начинаем с ячейки 1,1 и удовлетворяем потребность первого потребителя за счёт первого склада. Если на складе ещё что-то после этого осталось, оно идёт на покрытие заявки второго потребителя, т.е. направо (ячейка 1,2), если же заявка полностью не удовлетворена, то остаток потребности удовлетворяется за счёт следующего склада, т.е. движение вниз к ячейке 2,1 – и так далее.

Важно заметить, что двигаемся мы на каждом шаге строго горизонтально или вертикально, никаких диагональных перемещений быть не должно. Если у нас одновременно заканчивается и запас на данном складе и полностью удовлетворяется заявка данного потребителя, то мы сначала делаем шаг вправо, к следующему потребителю, удовлетворяем его заявку с данного склада (перевозки равны нулю), и лишь после этого делаем шаг вниз, к следующему складу. Тем самым мы обеспечиваем постоянно одно и то же число базисных ячеек – пусть даже некоторые из них, вырожденные, содержат нулевые значения. Мы их будем принципиально отличать от оставшихся ячеек, называемых свободными и по определению всегда содержащих нулевые значения перевозок, так что в них ноль даже и не пишется.

Рассмотрим этот метод на примере – составим оптимальный план для таблицы:

	B_1	**B_2**	**B_3**	**B_4**	**B_5**	Запасы $\boldsymbol{a_i}$
A_1	10 **18**	8 **27**	5 **3**	6	9	48
A_2	6	7	8 **30**	6	5	30
A_3	8	7	10 **9**	8 **12**	7 **6**	27
A_4	7	5	4	6	8 **20**	20
Заявки $\boldsymbol{b_j}$	18	27	42	12	26	125

Мы сразу получили план перевозок, удовлетворяющий условиям (2) и (3). Он является опорным, т.к. базисных (не равных 0) переменных ровно $\boldsymbol{r=n+m-1=8}$. Остальные “пустые” клетки соответствуют нулевым перевозкам, т.е. свободным переменным. Таких у нас получилось $\boldsymbol{(n-1)(m-1)=12}$.

Этот план не является оптимальным. Действительно, его стоимость получается 1039. Если, к примеру, перенести 18 единиц из ячейки 1,1 в 1,2 – а для того, чтобы сохранить баланс, те же 18 единиц перенести из ячейки 2,3 в 1,3. – то получится план, имеющий стоимость уже 913.

Число ненулевых переменных может быть меньше, чем $\boldsymbol{r=n+m-1}$, т.е. в данном случае меньше 8, т.е. часть базисных переменных также равна 0. Это *вы-*

рожденный план. Он возникает, когда одновременно заканчивается и запас на очередном складе, и заявка от очередного потребителя, и следующая заполненная ячейка будет располагаться не ниже (“южнее”) и не правее (“восточнее”), а “юго-восточнее” последней заполненной. Но мы предполагаем, что сначала сделан шаг направо, заполнена клетка (величина перевозки равна нулю) и после этого сделан шаг вниз. В дальнейшем ***всегда*** будем считать, что в плане перевозок $\boldsymbol{r=n+m-1}$ базисных клеток, даже если в каких-то из них и помещаются “нулевые” значения перевозок. Это удобно для построения алгоритма.

Методы приведения к стандартному виду очевидно зависят от того, какое отличие было у приводимой задачи.

Транспортная задача с избытком запасов

Рассмотрим ситуацию, когда поданных заявок меньше, чем запасов на складах. Равенство (1) не выполняется.

$$\sum_{i=1}^{m} a_i > \sum_{j=1}^{n} b_j$$

В этом случае вводим ещё один (фиктивный) пункт назначения, “заявка” которого равна остатку груза, невостребованному потребителями:

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j$$

Реально *перевозки* из этого ПН означают, что такое количество груза останется на данном складе. Стоимость перевозок для данного ПН в общем случае будет равна нулю ($\boldsymbol{c_{i,n+1}=0,\ i=1,\ldots,m}$), хотя можно ввести цену и отличную от нуля: стоимость или убытки от хранения единицы груза. Последнее имеет смысл, если эта стоимость на разных складах (пунктах отправления) существенно различна.

Транспортная задача с избытком заявок (с недостатком запасов)

Это задача прямо противоположна ранее рассмотренной. Здесь запасов не хватает на всех потребителей, то есть

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j \,.$$

Есть несколько способов решения такой задачи.

Первый – вводится “фиктивный” $\boldsymbol{m+1}$–ый склад, запасы на котором равны

$$a_{m+1} = \sum_{j=1}^{n} b_j - \sum_{i=1}^{m} a_i$$

и стоимость перевозок с этого склада любому потребителю равна нулю. На самом деле цифра перевозок с этого ПО какому-то потребителю означает, что именно такое количество груза из заявленного *не будет доставлена* данному потребителю. Этот метод применим тогда, когда не важно, каким потребителям и в каком количестве недогрузить товар. Можно решить эту задачу и с учётом степени важности получателей груза. В этом случае вводятся различные стоимости перевозки единицы груза к каждому из потребителей, имеющие смысл штрафа за недопоставку единицы заказанного товара этому потребителю.

Второй способ – “исправляем” заявки, умножив каждую на коэффициент

$$k = \sum_{i=1}^{m} a_i \Big/ \sum_{j=1}^{n} b_j \,.$$

Тем самым мы в равной пропорции недогрузили потребителей. Возможны иные варианты, но все они сводятся к уменьшению заявок до выполнения равенства (1) и решения задач с “нормальным” балансом.

Индивидуальные задания по теме № 2.

«Построение моделей, приводящих к транспортным задачам»

Каждому студенту выдаются условия задачи: матрица стоимостей транспортировки, вектора заказов и заявок. Все задачи не сбалансированы.
Требуется:
1 – составить транспортную задачу линейного программирования по определению оптимального плана выпуска продукции;
2 – привести эту задачу к стандартному виду.

Практическое занятие № 3.

«Построение моделей, приводящих к задачам линейного программирования с дополнительными ограничениями»

В данном разделе рассматриваются методы, которые также могут быть отнесены к методам дискретной оптимизации. Характерной особенностью этих методов является наличие дополнительного требования о том, что все либо только часть переменных в оптимальном решении должны обладать свойством целочисленности, или как вариант – быть кратными некоторому заданному числу. Иногда встречаются и иные ограничения. Данные методы тесно связаны с задачами линейного программирования, однако дополнительные требования делают задачу нелинейной. Поэтому стандартные методы линейного программирования для решения этих задач непригодны, хотя в ряде случаев используются в качестве "вспомогательных" в составе более сложных алгоритмов решения целочисленных задач.

В качестве примера целочисленной задачи можно привести стандартную задачу составления оптимального по прибыли плана выпуска продукции предприятием в условиях ограниченных ресурсов – если вспомнить, что любой вид продукции мы можем выпустить лишь в целом количестве экземпляров. При этом округление до ближайшего целого числа далеко не всегда даёт наилучшее из целочисленных решений, а может вообще оказаться вне области допустимых решений – тогда как наилучшее из целочисленных решений может оказаться весьма далеко от нецелочисленного оптимума, найденного методами линейного программирования.

С другой стороны, стандартная транспортная задача не содержит в себе требования целочисленности, однако, если все граничные условия целочисленны, то полученный стандартными методами оптимальный план перевозок также оказывается целочисленным, несмотря на то, что для достижения этого не прилагалось никаких специальных усилий.

К транспортной задаче может быть приложено дополнительное требование не целочисленности, а дискретности, то есть кратности одному или одному из нескольких целых чисел. Представим себе транспортную сеть, связывающую склады и потребителей. Перевозки по этой сети осуществляет транспортная фирма, имеющая грузовики одного или нескольких номиналов грузоподъёмности. В таком случае стоимость перевозки от данного склада данному потребителю должно задаваться не для единицы груза, а для одного рейса грузовика (или для рейса по отдельности каждого номинала грузоподъёмности). Нам придётся по каждому маршруту отправлять количество груза, кратное грузоподъёмности. Возможен вариант отправки не полностью загруженного автомобиля, но стоимость рейса (аренды) будет точно такой же, как у полностью загруженного.

Есть ещё ряд целочисленных или дискретных транспортных задач, называемых задачей о рюкзаке (ранце), задачей о бомбардировщике и т.п. В наиболее сложных случаях "контейнер" может быть разбит на отсеки определённой ёмкости,

предназначенные для перевозки разных видов груза. Например, “бензовоз” с отсеками для перевозки горючего с различным октановым числом.
Можно встретить и несколько иной тип задач, также называемый в некоторых учебниках “задачей о рюкзаке”. Это задача, внешне сходная с задачей линейного программирования, имеющая линейную целевую функцию и линейные ограничения вида – найти максимум функции

$$\max(\min) Z = \sum_{j=1}^{n} c_j x_j$$

с линейными ограничениями

$$\sum_{j=1}^{n} a_{ij} x_i = b_i, \quad i = 1, \ldots, m$$

при этом на все переменные наложено стандартное для задач линейного программированмя условие неотрицательности

$$x_j \geq 0, \quad j = 1, \ldots, n$$

и, кроме того, на все переменные наложено дополнительное условие

$$x_j \in \{0;1\}, \quad j = 1, \ldots, n$$

таким образом, любая переменная может принимать только одно из двух значений – 0 или 1, т.е. мы как бы решаем, “класть в рюкзак” данный предмет или нет, но положить можем только одну штуку любого вида.
Теперь рассмотрим некоторую классификацию. Можно выделить:

1. задачи линейного программирования с дополнительным требованием целочисленности, причём они разделяются на полностью целочисленные и частично целочисленные (требование целочисленности касается не всех переменных, а только части);
2. транспортные задачи формально к целочисленным не относятся, но если исходные данные целочисленны, то получим и целочисленный оптимальный план;
3. Задачи с неделимостью (с кратностью некоторому числу, обычно целому);
4. Комбинаторные задачи (например, о коммивояжере) используются для решения задач, в которых число возможных решений заведомо конечно. Сюда же относятся задачи с булевыми переменными, т.е. принимающими только значения 0 и 1. Например, какое-то действие или назначение может либо состояться, либо нет. Хотя иногда комбинаторные и булевы задачи относят к разным группам и применяют методы решения отдельно – только комбинаторный или только булевый.

Любая классификация в некоторой мере условна, т.е. какую-либо задачу и метод можно без большой натяжки отнести к нескольким разным группам. Так, симплекс-метод ищет оптимальное решение не на всей ОДР – многограннике с бесконечным числом точек, а на конечном множестве вершин этого многогранника. Поэтому его иногда относят к комбинаторным методам и используют как составную часть комбинаторных алгоритмов.

Для решения подобных задач используются методы: отсечения, ветвей и границ, приближённые и ряд других. Далее рассмотрим некоторые наиболее употребительные из целочисленных и дискретных моделей и методов.

Индивидуальные задания по теме № 3.

«Построение моделей, приводящих к задачам линейного программирования с дополнительными ограничениями»

Рассматривается задача, полученная студентом при изучении темы №1. Её требуется дополнить условием целочисленности. Поскольку план может составлять лишь целое количество выпуска любого вида продукции.

Практическое занятие № 4.

«Многокритериальные задачи. Построение обобщённых критериев. Метод последовательных уступок. Метод идеальной точки.»

Бывают практические ситуации, когда вместо одного показателя W требуется оценивать эффективность операции по целой группе показателей $W_1, W_2, \ldots, W_k$. Некоторые из них надо увеличить, некоторые уменьшить. Решения обычно несовместимы, то есть если максимизируем один критерий, другой может сильно ухудшиться и т.п. Количественный анализ может здесь помочь выбросить явно нерациональные варианты. Эти ситуации также называются задачами многокритериальной оптимизации, также они рассматриваются в задачах выбора и принятия решений.

Рассмотрим геометрическую интерпретацию: пусть требуется составить суточный рацион, содержащий все необходимые для человека питательные вещества, из заданного набора продуктов. Пусть эта задача решена. Очевидно, что наборов продуктов, отвечающих этому требованию, будет несколько. По каким критериям выбрать из них наилучший? Для примера возьмём два критерия: стоимость рациона S и его вес m. Оба эти критерия желательно минимизировать. Но, как видим, экстремумы этих критериев не совпадают, самый лёгкий из них не является самым дешёвым. Таким образом окончательный выбор сделает человек, и этот выбор будет компромиссом между этими двумя критериями. Возможно, на выбор повлияют и другие соображения. Тем не менее очевидно, что при этом в любом случае будет выбран один из четырёх рационов, расположенных слева внизу. Они различаются и массой, и стоимостью, но по обоим этим критериям превосходят остальные 15, которые мы отбросим как заведомо невыгодные, поскольку для каждого из них при том же весе существуют более дешёвые, а при той же стоимости – более лёгкие.

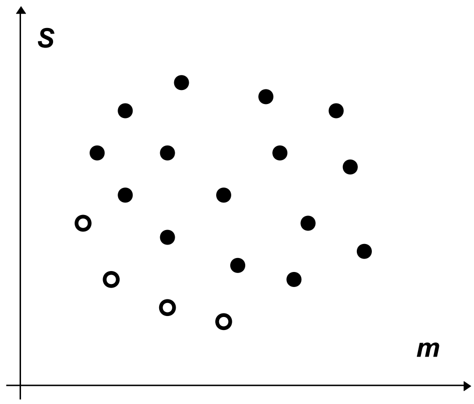


Рисунок 1.1. Операция с двумя показателями

Полученное подмножество решений называется **поверхностью Парето**. Члены этого множества удовлетворяют следующему условию: невозможно улучшить значение одного из показателей без того, чтобы при этом не ухудшилось значение хотя бы какого-то из оставшихся показателей.

Рассмотрим ещё один пример. Пусть есть 2 показателя – вероятность выполнения задачи P и стоимость S. Нас интересует максимум первого критерия при минимуме второго, но они явно не совпадают. Пусть есть набор критериев.

Изобразим их на плоскости, в осях координат ***W*** и ***P***. Очевидно, что предпочесть надо выделенные решения. Те, что лежат слева от них, имеют при той же стоимости более низкую вероятность выполнения задачи. Остаётся проанализировать эти 6 вариантов вместо 30. Нижний – самый дешевый, но с небольшой вероятностью. Верхний – более вероятный, но самый дорогой. Тут уже будет разбираться принимающий решения человек.

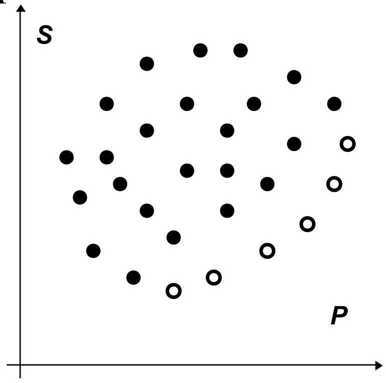


Рисунок 1.2. Варианты операций ко второму примеру

Разумеется, критериев может быть и больше двух. Принцип удаления заведомо неоптимальных решений при этом не изменится, но при этом не будет таких наглядных геометрических интерпретаций. Математических методов для этого существует достаточно много, они известны как “теория выбора” и “теория принятия решений”. Коротко рассмотрим наиболее часто применяемые методы окончательного выбора. При этом напомним, что достаточно часто задачи, принадлежащие к одной и той же группе, можно решать различными методами. Мы это увидим по мере изучения соответствующих разделов.

Построение обобщённых критериев

После того, как множество вариантов сокращено, надо выбрать из них один. Это можно сделать двумя способами – либо принимает решение человек, руководствуясь интуицией и рядом дополнительных соображений, не включённых в модель, либо строится *обобщённый критерий*, который включает в себя все частные критерии. Рассмотрим этот случай подробнее.

Пусть имеется $\boldsymbol{k}$ критериев $\boldsymbol{W_1, W_2, ..., W_k}$, часть из которых требуется максимизировать, а часть – минимизировать. Пусть максимизировать надо первые $\boldsymbol{m}$ критериев, а остальные минимизировать (иначе критерии можно просто перенумеровать. В качестве обобщённого критерия используем дробь

$$U = \frac{W_1 \times \ldots \times W_m}{W_{m+1} \times \ldots \times W_k}$$

Здесь в числителе те показатели, которые нужно увеличить, а в знаменателе – те, которые нужно уменьшить. Недостаток такого подхода – разнородные показатели могут взаимно компенсироваться, например, по обобщённому критерию будет выбран вариант с малой вероятностью осуществления, но зато очень дешёвый.

Существует и *другой способ* построения обобщённых показателей, более гибкий и позволяющий учесть больше особенностей задачи. Так, при выборе рациона кроме рассмотренных критериев – массы и стоимости – можно использовать и ряд других, например, длительность хранения, время и трудоёмкость приготовления, стойкость к транспортировке и погодным условиям и ряд других. Очевидно, эти критерии имеют неодинаковое значение, но их сравнительная значимость сильно зависит от ряда факторов. Например, если рационы на достаточно большой срок требуется нести на себе, то наибольшую важность имеют вес и стойкость к транспортировке. Если их используют не перемещаясь – на первый план выходят другие критерии.

В таких случаях обобщённый показатель строится как линейная комбинация: $\boldsymbol{U=\alpha_1 W_1+\alpha_2 W_2+...+\alpha_k W_k}$. Здесь коэффициенты $\alpha_1,\alpha_2,...,\alpha_k$ положительны при тех показателях, которые надо увеличить, и отрицательны при тех, которые надо уменьшить. Степень важности показателя определяется абсолютной величиной соответствующего ему коэффициента. Недостаток у этого способа – тот же, что и у ранее рассмотренного.

В ряде практических случаев задачу многокритериального выбора сводят к задаче с одним показателем. Выделяется главный показатель, по которому и проводится оптимизация, при этом остальные показатели превращают в ограничения, которые входят в комплекс заданных условий.

Метод последовательных уступок

Это способ построения компромиссного решения для многокритериальной задачи. Предполагается, что показатели $\boldsymbol{W_1, W_2, W_3,...,}$ расположены в убывающей последовательности по значимости. Сначала идет самый основной показатель $\boldsymbol{W_1}$, затем $\boldsymbol{W_2, W_3,...}$ по степени убывания значимости. Предположим, что все эти критерии надо максимизировать. Если это не так, то соответствующие критерии умножаем на –1.

На первом шаге решается задача с одним критерием эффективности $\boldsymbol{W_1}$, остальные показатели пока не рассматриваются. Находится решение, которое максимизирует главный показатель $\boldsymbol{W_1}$. Теперь, когда оно найдено, назначается величина *уступки* $\boldsymbol{\Delta W_1}$, то есть величина, на которую мы согласны отклониться от максимума в показателе $\boldsymbol{W_1}$, чтобы максимизировать показатель $\boldsymbol{W_2}$. Вводим это ограничение в качестве ещё одного условия операции.

На следующем шаге с добавленным ограничением решаем задачу оптимизации для находим решение, обеспечивающее максимум второго показателя, затем определяется величина *уступки* $\boldsymbol{\Delta W_2}$ в показателе $\boldsymbol{W_2}$, чтобы за счёт неё максимизировать показатель $\boldsymbol{W_3}$. Процесс продолжается до тех пор, пока не будут рассмотрены все критерии.

Напомним, что все эти способы остаются не до конца формализованными, то есть окончательный выбор все равно связан с волевым решением человека. Данные методы предоставляют лишь возможность оценить преимущества и недостатки каждого варианта выбора.

Метод идеальной точки

Этот метод выбора окончательного решения из всего множества поверхности Парето, он также называется методом точки утопии. Метод применим, если в пространстве критериев $\boldsymbol{W_1, W_2, W_3,...,W_n}$ все «оси координат» можно считать равноценными, то есть в этом пространстве можно определить (ввести) метрику, т.е. понятие **расстояния**. Таким образом, мы не только получаем для каждо-

го исхода операции чётко определённую точку в пространстве признаков, но ещё и задаём меру близости этих исходов.
В произвольном случае мы должны для каждого критерия выбрать соответствующий масштаб таким образом, чтобы одинаковое расстояние по любой из осей соответствовало бы равной степени значимости улучшения (или ухудшения) общей ситуации. Применительно к теории игр ситуация несколько упрощается. Здесь каждый из критериев соответствует одному и тому же понятию – выигрышу, просто относятся эти выигрыши к разным игрокам. Таким образом метрика в пространстве критериев определяется без каких-либо дополнительных действий.
Теперь рассмотрим сам метод. Мы знаем, что «Утопией» называли государство, имевшее множество достоинств и только один недостаток: в реальности оно не существовало. Также и «идеальная точка», она же «точка утопии» -- является идеальным, но недостижимым для данной системы операций исходом.
Пусть имеется многокритериальная задача с набором критериев $W_1, W_2, W_3,...,W_n$, причём на пространстве этих показателей определена метрика. Не ограничивая общности вопроса, будем считать, что все эти показатели следует максимизировать. Найдём по отдельности максимумы каждого из этих показателей, не обращая внимания на остальные: W_1 **max**, W_2 **max** и т.д. Очевидно, что эти значения несовместны.
Точку с координатами **(**W_1 **max,** W_2 **max,** W_3 **max,...,**W_n **max)** будем называть идеальной точкой и обозначать ***Ut***. Очевидно, что решением задачи такая точка не является, поскольку не принадлежит области допустимых решений – как и к поверхности Парето. Однако она позволяет легко сделать выбор из достижимых вариантов: окончательным решением будет та точка поверхности Парето, которая ближе всего к идеальной точке.
Рассмотрим двумерную задачу (рис. 1.3.):

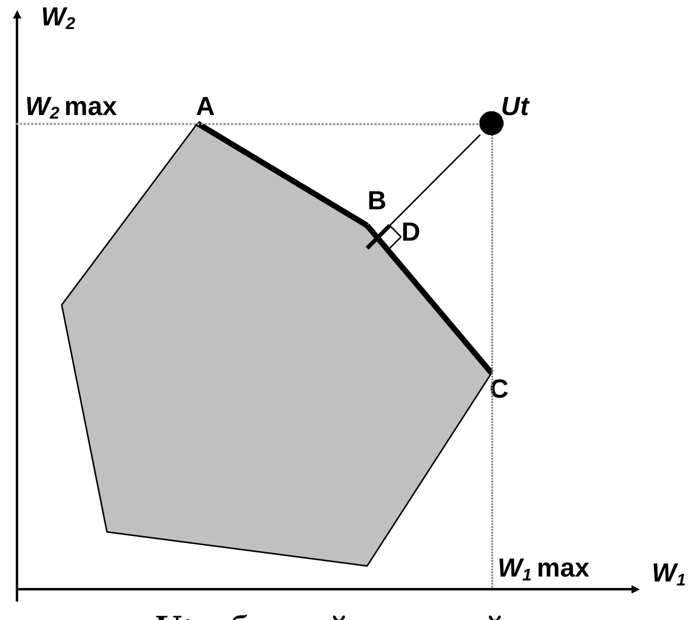


Рисунок 1.3. Точка утопии ***Ut*** и ближайшая к ней точка на поверхности Парето

Требуется максимизировать критерии W_1 и W_2. Геометрическая фигура на рисунке – область допустимых решений. Мы видим, что максимумам по первому и второму критериям W_1 **max** и W_2 **max** соответствуют различные точки -- **A** и

C. Поверхность Парето состоит из двух отрезков – **AB** и **BC**. Точка утопии ***Ut*** не принадлежит ОДР. Ищем ближайшую к ней точку поверхности Парето. Как видим, это точка **D** на отрезке **BC**. Она и будет окончательным решением.

Индивидуальные задания по теме № 4.

«Многокритериальные задачи. Построение обобщённых критериев. Метод последовательных уступок. Метод идеальной точки.»

Рассматривается множество решений на плоскости. Для него строится поверхность Парето и находится решение методом идеальной точки.

Практическое занятие № 5.

«Решение задач целочисленного программирования методом отсечений»

Пусть имеем стандартную задачу линейного программирования - поиск максимума функции

$\max(\min)Z = \sum_{j=1}^{n} c_j x_j$ (1)

с ограничениями (линейными)

$\sum_{j=1}^{n} a_{ij} x_i = b_i, \quad i = 1,\ldots,m$ (2)

при этом на все переменные наложено стандартное для задач линейного программированмя условие неотрицательности

$x_j \geq 0, \quad j = 1,\ldots,n$ (3)

и, кроме того, на все переменные наложено дополнительное условие

x_j – целые, ***j*=1,...,*n*** (4)

Для большинства "целочисленных" методов требуется, чтобы все элементы в системе уравнений (2) были целочисленны. Этого нетрудно добиться умножением соответствующих уравнений на подходящее целое число, так что в принципе любую систему линейных уравнений можно привести к соответствующему виду.

Решение этой задачи находится следующим образом:

1. Сначала решаем обычную задачу линейного программирования для условий (1)–(3), игнорируя требование целочисленности (4) – например, стандартным симплекс-методом.
2. Получив оптимальное решение (если оно существует), проверяем его на целочисленность (4). Возможно, полученное решение "само" оказалось целочисленным, и нам не требуется прибегать к каким-либо дополнительным мерам. Если для всех переменных оно выполняется, то получено оптимальное решение задачи. Если задача (1)–(3) не имеет решения, то его не имеет и задача (1)–(4).
3. Если хоть одна из переменных в полученном оптимальном решении – не целое число, то строится дополнительное ограничение, которое отсекает часть области, в которой содержится оптимальное решение задачи (1)–(3) и не содержится ни одного допустимого значения задачи (1)–(4), т.е. ни одного целого «вектора».
4. Возвращаемся к задаче (1)–(3) и решаем её с учётом появившегося дополнительного ограничения. Если опять получено дробное решение, то опять вводим дополнительное ограничение и так до тех пор, пока не будет получено целочисленное решение.

Это так называемый алгоритм Гомори. Он позволяет за конечное число шагов прийти к оптимальному целочисленному решению, если оно существует.

Основа этого метода – составление дополнительного ограничения, так называемого "правильного отсечения". Оно должно:

а – быть линейным;

б – отсекать от ОДР найденное оптимальное нецелочисленное решение задачи;

в – не затрагивать ни одной из целых “точек” исходной задачи.
Пусть задача без учёта условия целочисленности решена. Обозначим ***B*** множество индексов переменных, соответствующих *базисным* переменным оптимального решения. Соответственно обозначим ***R*** множество индексов ***j***, которые соответствуют *свободным* переменным. После каждой итерации система ограничений, каждое из которых соответствует строке таблицы симплекс-метода имеет вид

$$x_i = b_i - \sum_{j\in R} a_{ij} x_j, \quad i \in B \quad (5)$$

В случае, если выполняется условие оптимальности, находим оптимальное решение, в котором все свободные переменные равны нулю, а все базисные равны соответствующим свободным членам $x_j^* = 0, \quad j \in R; \quad x_i^* = \beta_i, \quad i \in B$. Если полученное решение целочисленно, то задача решена. Рассмотрим ситуацию, когда это не так, т.е. хотя бы один элемент решения принимает нецелочисленное значение.
Пусть какие-то b_i – нецелые. Например, b_{i0}.
Рассмотрим i_0 – ое равенство системы (5). При этом мы помним, что для данной системы выполнено условие оптимальности.

$$x_{i_0} = b_{i_0} - \sum_{j\in R} a_{i_0 j} x_j \quad (6)$$

Обозначим целую и дробную части числа ***a*** как **[*a*]** и **{*a*}**.
Напомним свойства целой и дробной части. Целая часть может быть какой угодно, но дробная всегда неотрицательна. Например **[7,3]=7** и **{7,3}=0,3**. Но **[-7,3]=– 8** и **{-7,3}=0,7**. **– 8 + 0,7 = – 7,3**.
Очевидно, что ***a*=[*a*]+{*a*}**. Слагаемые в уравнении (6) разделим на целые и дробные части и группируем. Получим

$$x_{i_0} = \left(\left[b_{i_0} \right] - \sum_{j\in R} \left[a_{i_0 j} \right] x_j \right) + \left(\left\{ b_{i_0} \right\} - \sum_{j\in R} \left\{ a_{i_0 j} \right\} x_j \right) \quad (7)$$

Выражение в левых круглых скобках – целое, а для того, чтобы x_{i0} было целым, необходимо, чтобы была целой и величина

$$L_{i_0} = \left\{ b_{i_0} \right\} - \sum_{j\in R} \left\{ a_{i_0 j} \right\} x_j$$

также была целой. А поскольку x_{i0} – координата *допустимого*, т.е. целочисленного решения задачи (1)–(4), то L_{i_0} – всегда целое число. Очевидно, что $L_{i_0} \le 0$.
В самом деле, величина $\sum_{j\in R} \left\{ a_{i_0 j} \right\} x_j$ не может быть отрицательной, $0 \le \left\{ b_{i_0} \right\} \le 1$. Так как L_{i_0} – целое, то из предположения $L_{i_0} > 0$ следует $\left\{ b_{i_0} \right\} > 1$, чего не может быть, т.к. это дробная часть числа. Итак, любое допустимое решение задачи (1)–(4) должно удовлетворять неравенству

$$\left\{ b_i \right\} - \sum_{j\in R} \left\{ a_{ij} \right\} x_j \le 0$$

Это соотношение и определяет правильное отсечение Гомори (его называют также *первым* правильным отсечением Гомори).

Признаком отсутствия целочисленного решения служит появление в таблице хотя бы одной строки с дробными свободными членами и целыми остальными коэффициентами, *так как в этом случае соответствующее уравнение не имеет решения в целых числах.*
Теоретически всё получается достаточно надёжно, но вот при практической реализации алгоритмов, реализующих данную схему метода, возникает ряд проблем.
Первая проблема состоит в том, что какова бы ни была точность вычислений, в любом случае количество разрядов под запись числа “с плавающей точкой” ограничено, вычисления выполняются с некоторой конечной точностью, поэтому дробное число может восприниматься как целое и наоборот. Способ борьбы с таким недостатком – представление коэффициентов в таблице симплекс-метода в виде простых дробей, т.е. каждое число записывается в виде двух целых чисел, отдельно числителя и знаменателя.
Вторая проблема состоит в том, что после решения нецелочисленной задачи мы можем получить не одно, а несколько нецелочисленных значений в столбце свободных членов. Однако на каждом шаге алгоритма мы должны ввести только одно повое отсечение, т.е. выбрать из всех строк таблицы с нецелым свободным членом только одну строку. Как лучше сделать такой выбор? На практике оказывается, что от такого выбора существенно зависит скорость сходимости алгоритма. Гомори предложил два способа выбора строки таблицы для построения отсечения.
Первый способ. Если оптимальный план содержит несколько дробных компонент, то дополнительное ограничение рекомендуется записывать для компоненты с наибольшей дробной частью (т.е. для строки с наибольшей дробной частью свободного члена).

$$\max\left\{b_i\right\}$$

Второй способ – выбирается та строка, в которой достигнут максимум дробной части отношения свободного члена к сумме всех коэффициентов данной строки.

$$\max\left\{b_i \Big/ \sum_{j\in R} a_{ij}\right\}$$

Второй способ обеспечивает более быструю сходимость.
Заметим также, что требование целочисленности в данном алгоритме распространяется на все переменные – как переменные исходной задачи, так и вновь добавляемые в процессе решения.
Пример. Рассмотрим применение этого метода на практическом примере, рассмотренном в разделе 2.4. – только дополним его требованием целочисленности. В итоге получим задачу:
Найти максимум целевой функции

$$z = 70x_1 + 30x_2 \to max$$

при ограничениях

$$\begin{cases} 10x_1 + 4x_2 \le 40 \\ 6x_1 + 4x_2 \le 36 \end{cases}$$
$$x_j \ge 0, \quad j = 1,2$$
$$x_j - \textit{целые}, \quad j = 1,2$$

Видно, что задача имеет нестандартный вид – целевая функция стремится к максимуму, ограничения записаны в виде неравенств. Приведём задачу к стандартному виду. Сначала преобразуем целевую функцию так, чтобы она стремилась к минимуму

$$L = -z = -(70x_1 + 30x_2) \to min$$

Теперь преобразуем ограничения из неравенств в равенства, для чего к правой части каждого из неравенств добавим по дополнительному неотрицательному переменному

$$\begin{cases} 10x_1 + 4x_2 + x_3 = 40 \\ 6x_1 + 4x_2 + x_4 = 36 \end{cases}$$
$$x_j \ge 0, \quad j = 1,\ldots,4$$

Теперь надо решить эту задачу симплекс-методом без учёта требования целочисленности. Эта задача была решена в разделе 2.4. В итоге была получена таблица вида

Таблица 5.1

Базисные переменные	Свободный член	Свободные переменные	
		x_3	x_4
L	**–295**	**–25/4**	**–5/4**
x_1	**1**	**1/4**	**–1/4**
x_2	**15/2**	**–3/8**	**5/8**

Найденное оптимальное решение имеет вид:
Свободные переменные равны нулю x_3=x_4=0, базисные переменные равны соответствующим свободным членам x_1=1, x_2=7,5. Целевая функция L = –295. Вспомним теперь требование целочисленности и проверим полученное решение. Требование не выполняется, так как переменная x_2=7,5 нецелочисленна. Следовательно, мы должны построить дополнительное отсечение. Выберем в таблице какую-нибудь строку с нецелочисленным свободным членом. Она там одна, самая нижняя (выделена), со свободным членом 15/2. Соответственно по рассмотренному правилу строим правильное отсечение Гомори

$$\frac{1}{2} - \left(\frac{5}{8}x_3 + \frac{5}{8}x_4\right) \le 0 \text{ или } \frac{5}{8}x_3 + \frac{5}{8}x_4 \ge \frac{1}{2}$$

Заметим, что хотя в строке при переменных x_3 и x_4 различные коэффициенты -- –3/8 и 5/8, но в отсечении получаем одинаковые коэффициенты 5/8. Это связано со свойством дробной части отрицательного числа – целая и дробная часть числа –3/8 соответственно равны

$$\left[-\frac{3}{8}\right] = -1 \quad \left\{-\frac{3}{8}\right\} = \frac{5}{8}$$

Полученное дополнительное ограничение записано в виде неравенства. Необходимо преобразовать его в стандартное ограничение - равенство. Для этого введём дополнительную неотрицательную переменную x_5. Получим

$$\frac{5}{8}x_3 + \frac{5}{8}x_4 - x_5 = \frac{1}{2}.$$

Выразим отсюда новую переменную , которую будем использовать в качестве базисной

$$x_5 = -\frac{1}{2} - \left(-\frac{5}{8}x_3 - \frac{5}{8}x_4\right)$$

Заметим, что эта переменная выражена через те же свободные переменные x_3 и x_4, что и остальные переменные в таблице 5.1. Таким образом нам не требуется возвращаться к самому началу решения, достаточно лишь добавить новую строку к таблице 5.1. В результате получим

Таблица 5.2

Базисные переменные	Свободный член	Свободные переменные	
		x_3	x_4
L	**–295**	**–25/4**	**–5/4**
x_1	**1**	**1/4**	**–1/4**
x_2	**15/2**	**–3/8**	**5/8**
x_5	**–1/2**	**–5/8**	**–5/8**

Рассмотрим полученную таблицу. Сразу заметим, что столбец свободных членов содержит отрицательный элемент –1/2, то есть это решение не является допустимым. Этого и следовало ожидать, поскольку дополнительное ограничение отсекает от ОДР найденное на предыдущем шаге оптимальное нецелочисленное решение.

Опять решим задачу, не обращая внимания на условие целочисленности. Выберем в таблице разрешающий элемент. Для этого рассмотрим строку с отрицательным свободным членом –1/2 и найдём в ней отрицательные коэффициенты. Их целых два, оба –5/8. Рассмотрим оба столбца, содержащих эти коэффициенты, в качестве кандидатов в разрешающие столбцы.

В каждом из столбцов найдём коэффициенты, имеющие одинаковый знак со своими свободными членами. Таких всего четыре (выделены в таблице 5.2). В строке x_1 это коэффициент 1/4, в строке x_2 это коэффициент 5/8, и наконец в строке x_5 это оба коэффициента –5/8. Найдём теперь отношение соответствующих свободных членов к каждому из этих коэффициентов. В первой строке получим 1/(1/4)=4, во второй (15/2)/(5/8)=12, в третьей для обоих коэффициентов (–1/2)/(–5/8)=4/5, то есть наименьшее отношение получено для обоих коэффициентов последней строки. Поскольку оба эти коэффициента “равноправны”, то произвольно выберем, например, второй из них – соответственно получим разрешающую строку и разрешающий столбец.

Таблица 5.3

Базисные переменные	Свободный член	Свободные переменные	
		x_3	$x_4 \to x_5$
$\boldsymbol{L}$	**–295**	**–25/4**	**–5/4**
$\boldsymbol{x_1}$	**1**	**1/4**	**–1/4**
$\boldsymbol{x_2}$	**15/2**	**–3/8**	**5/8**
$\boldsymbol{x_5 \to x_4}$	**–1/2**	**–5/8**	**–5/8**

Переменную $\boldsymbol{x_4}$ будем перемещать в базисные, а $\boldsymbol{x_5}$ наоборот – в свободные. Пересчитаем коэффициенты по известному нам (рассмотренному в разделе 2.4.) алгоритму, в результате получим

Таблица 5.4

Базисные переменные	Свободный член	Свободные переменные	
		x_3	$x_4 \to x_5$
$\boldsymbol{L}$	**–295**	**–25/4**	**–5/4**
$\boldsymbol{x_1}$	**1**	**1/4**	**–1/4**
$\boldsymbol{x_2}$	**15/2**	**–3/8**	**5/8**
$\boldsymbol{x_5 \to x_4}$	**–1/2**	**–5/8**	**–5/8**

Переменную $\boldsymbol{x_4}$ будем перемещать в базисные, а $\boldsymbol{x_5}$ наоборот – в свободные. Пересчитаем коэффициенты по известному нам (рассмотренному в разделе 2.4.) алгоритму, в результате получим

Таблица 5.5

Базисные переменные	Свободный член	Свободные переменные	
		x_3	x_5
$\boldsymbol{L}$	**–294**	**–5**	**–2**
$\boldsymbol{x_1}$	**6/5**	**1/2**	**–2/5**
$\boldsymbol{x_2}$	**7**	**–1**	**5/8**
$\boldsymbol{x_4}$	**4/5**	**1**	**–8/5**

Проанализируем результат. В столбце свободных членов (не считая целевой функции) все числа неотрицательны, значит решение осталось допустимым (опорным). В строке целевой функции (не считая свободного члена) все коэффициенты отрицательны, значит решение оптимально. Поскольку в этой строке нет нулевых элементов, делаем вывод – найденное оптимальное решение единственно.

Запишем это оптимальное решение. Свободные переменные равны нулю x_3=x_5=0, базисные переменные равны соответствующим свободным членам x_1=6/5, x_2=7, x_4=4/5. Целевая функция L = –294.

Опять проанализируем решение с точки зрения целочисленности. Имеем целых два нецелочисленных переменных, x_1=6/5 и x_4=4/5. Надо выбрать из этих двух строк одну, по которой провести отсечение. В первом случае дробная часть свободного члена равна 1/5, во втором 4/5, то есть больше. Выберем эту строку (в таблице 5.5. выделена). Построим отсечение. Условие будет иметь вид

$$\frac{4}{5}-\left(0x_3+\frac{2}{5}x_5\right)\le 0$$

При переменной x_3 в строке коэффициент 1, т.е. целое число, значит в неравенство x_3 входит с нулевым коэффициентом. Преобразуем ограничение-неравенство в равенство, для чего вводим дополнительно неотрицательную переменную x_6. В результате получим

$$\frac{4}{5}-\left(0x_3+\frac{2}{5}x_5\right)+x_6=0$$

или в преобразованном виде

$$x_6=-\frac{4}{5}-\left(0x_3-\frac{2}{5}x_5\right)$$

Добавляем к последней таблице ещё одну строку. Получаем

Таблица 5.6

Базисные переменные	Свобод-ный член	Свободные перемен-ные	
		x_3	x_5
L	**–294**	**–5**	**–2**
x_1	**6/5**	**1/2**	**–2/5**
x_2	**7**	**–1**	**5/8**
x_4	**4/5**	**1**	**–8/5**
x_6	**–4/5**	**0**	**–2/5**

Эту задачу опять надо решить симплекс-методом без учёта требования целочисленности. Рассмотрим полученную таблицу. Заметим, что столбец свободных членов содержит отрицательный элемент –4/5, то есть это решение не является допустимым, как и после добавления предыдущего отсечения.
Так же, как и в предыдущем случае, выберем разрешающий элемент и соответственно разрешающую строку и разрешающий столбец.

Таблица 5.7

Базисные переменные	Свобод-ный член	Свободные перемен-ные	
		x_3	$x_5 \to x_6$
L	**–294**	**–5**	**–2**
x_1	**6/5**	**1/2**	**–2/5**
x_2	**7**	**–1**	**5/8**
x_4	**4/5**	**1**	**–8/5**
$x_6 \to x_5$	**–4/5**	**0**	**–2/5**

Переменную x_5 будем перемещать в базисные, а x_6 наоборот – в свободные. Пересчитаем коэффициенты по табличному алгоритму преобразования коэффициентов, в результате получим

Таблица 5.8

Базисные переменные	Свобод-ный член	Свободные перемен-ные	
		x_3	x_6
L	**–290**	**–5**	**–5**
x_1	**2**	**1/2**	**–1**
x_2	**5**	**–1**	**5/2**
x_4	**4**	**1**	**–4**
x_5	**2**	**0**	**–5/2**

Проанализируем результат. В столбце свободных членов (не считая целевой функции) все числа неотрицательны, значит решение осталось допустимым (опорным). В строке целевой функции (не считая свободного члена) все коэффициенты отрицательны, значит решение оптимально. Поскольку в этой строке нет нулевых элементов, делаем вывод – найденное оптимальное решение единственно.

Запишем это оптимальное решение. Свободные переменные равны нулю $\boldsymbol{x_3}=\boldsymbol{x_6}=0$, базисные переменные равны соответствующим свободным членам $\boldsymbol{x_1}=2$, $\boldsymbol{x_2}=5$, $\boldsymbol{x_4}=4$, $\boldsymbol{x_2}=2$. Целевая функция $\boldsymbol{L} = -290$.

Рассмотрим полученное решение с учётом условия целочисленности. Все свободные члены целочисленны, так что мы наконец получили целочисленное решение исходной задачи. С учётом того, что исходная задача содержала всего два переменных и преобразованную нами целевую функцию, можно записать найденное решение в виде:

$\boldsymbol{x_1}=2$, $\boldsymbol{x_2}=5$, целевая функция $\boldsymbol{Z} = 290$.

Значение целевой функции на 5 единиц меньше, чем у оптимального нецелочисленного решения – это та цена, которую пришлось уплатить за выполнение условия целочисленности. Полученная точка (2; 5) достаточно далеко отстоит от точки оптимального целочисленного решения (1; 7,5), так что получить решение простым округлением до ближайшего целого числа явно не получилось бы.

Заметим также, что в данном примере второе отсечение мы строили по строке для переменной $\boldsymbol{x_5}$, которая в исходную задачу не входила, а была добавлена при преобразовании условия первого отсечения к стандартному виду. Тем не менее к этой переменной также предъявляются требования целочисленности.

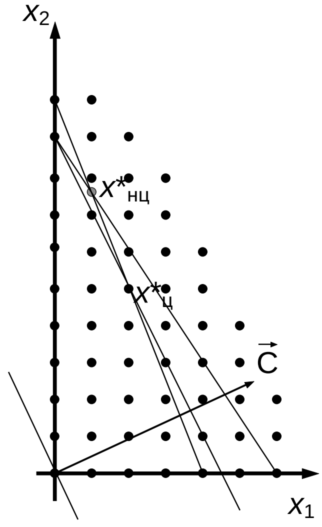


Рисунок 5.1. Геометрическая интерпретация рассмотренной задачи и её решения методом отсечений

На рисунке дана геометрическая интерпретация решения задачи. Показаны точка $x^*_{нц}$ нецелочисленного оптимального решения, линия, отсекающая её, и точка $x^*_{ц}$ оптимального целочисленного решения. Видно, что эта точка не могла быть получена округлением $x^*_{нц}$ до ближайшего целого значения.

Индивидуальные задания по теме № 5.

«Решение задач целочисленного программирования методом отсечений»

Задача, составленная в разделе 3, решается методом отсечений. Полученное решение сравнивается с оптимальным решением нецелочисленной задачи.

Практическое занятие № 6.

«Решение задач целочисленного программирования методом ветвей и границ»

Пусть имеется стандартная задача целочисленного линейного программирования – требуется найти экстремум (для определённости пусть будет максимум) целевой функции

$$\max \quad Z = \sum_{j=1}^{n} c_j x_j$$

с линейными ограничениями

$$\sum_{j=1}^{n} a_{ij} x_i = b_i, \quad i = 1, \ldots, m$$

при этом на все переменные наложено условие неотрицательности

$$x_j \geq 0, \quad j = 1, \ldots, n$$

и условие целочисленности

$$\boldsymbol{x_j} \text{ – целые, } \boldsymbol{j=1,\ldots,n}$$

Будем решать эту задачу с использованием общей схемы метода ветвей и границ. В основу метода положен тот уже замеченный нами при рассмотрении методов отсечений факт, что целевая функция для целочисленного решения принимала “худшее” значение, чем для нецелочисленного – это позволяет находить верхние границы на подмножествах. Как и при решении задач методами отсечений, сначала решаем задачу линейного программирования, игнорируя требование целочисленности, каким-нибудь стандартным методом решения – например, симплекс-методом.

Если в результате мы получили полностью целочисленное решение, то дальнейших шагов не требуется. Если задача не имеет оптимального решения, то и целочисленная задача также не имеет оптимального решения. Если же хотя бы одна из компонент плана имеет отличную от нуля дробную часть, то будем искать целочисленное решение.

Пусть в найденном оптимальном нецелочисленном плане X^* имеется некоторое нецелое число x^*_i, для которого $\{x^*_i\}>0$. Поскольку и само значение $x^*_i>0$, то очевидно, что этот нецелочисленный элемент решения заключён в промежутке между двумя целыми числами $K_i<x^*_i<K_i+1$, где $K_i=[x^*_i]$ – целая часть числа x^*_i Очевидно, что в целочисленном решении данной задачи эта переменная примет либо значение K_i и меньше, либо значение K_i+1 и больше. Других целочисленных значений просто не существует. *Разветвим* наше множество решений. Добавим к системе ограничений в одном случае условие $x^*_i<K_i$, а во втором – условие $x^*_i >K_i+1$. В двух полученных задачах линейного программирования не будет ни одного общего целочисленного решения, при этом ни одного целочисленного решения из ОДР исходной задачи мы не потеряем, т.е. основное условие разбиения на подмножества выполнено. При этом, правда, из ОДР исходной задачи будет удалена полоса $K_i<x^*_i <K_i+1$, но она не содержит ни одно-

го целочисленного решения. С этой полосой будет удалено также найденное на предыдущем шаге оптимальное нецелочисленное решение, таким образом наблюдаем некоторое сходство с методами отсечений. Однако наряду с отсечением нецелочисленного решения мы делим оставшуюся ОДР на две части. Но какое-нибудь из двух подмножеств может оказаться пустым.

Теперь решим уже две полученные задачи линейного программирования каким-либо стандартным методом, не учитывая условия целочисленности. Проанализируем полученные решения. Всего возможны шесть вариантов исхода:

1. Обе получившиеся задачи не имеют оптимального решения. Вывод: исходная задача имела оптимальное нецелочисленное решение, но оптимального целочисленного решения она не имеет.
2. Одна из задач не имеет решения, другая имеет целочисленное оптимальное решение. Последнее и является решением исходной задачи.
3. Одна из задач не имеет решения, другая имеет оптимальное, но опять нецелочисленное решение. Выбираем какую-либо из нецелочисленных компонент этого решения и опять разбиваем задачу на две задачи, аналогично описанному ранее. Решаем их без учёта условия целочисленности и анализируем результат.
4. Обе задачи имеют решение, и оба решения целочисленны. В этом случае сравниваем найденные значения целевой функции и выбираем то, у которого оно больше.
5. Обе задачи имеют решение, и оба решения имеют нецелочисленные компоненты. Сравниваем целевые функции полученных решений. Выбираем задачу, для которой она больше – *с наибольшей вероятностью* оптимальное решение находится там. Разветвляем её на две задачи и решаем их.
6. Обе задачи имеют решение, одно из которых целочисленно, а другое имеет нецелочисленные компоненты. Сравниваем целевые функции полученных решений. Если больше целевая функция целочисленного решения, оно и является решением исходной задачи. Если же больше целевая функция нецелочисленного решения, то *возможно* искомое решение находится там. Выбираем одну из нецелочисленных компонент плана и производим разветвление этой задачи.

Заметим, что в случае вариантов 3, 5 и 6 после получения целочисленного решения нам необходимо будет выполнить процедуру обратного хода, то есть сравнить значение целевой функции найденного целочисленного решения с целевыми функциями ранее отброшенных задач. Возможно, придётся вернуться к какой-либо из этих задач и разветвлять уже её.

Индивидуальные задания по теме № 6.

«Решение задач целочисленного программирования методом ветвей и границ»

Задачу из предыдущей темы решаем методом ветвей и границ. Сравниваем результаты и трудоёмкость решения.

Практическое занятие № 7.

«Методы градиентного поиска»

Ведущее место среди прямых методов решения экстремальных задач занимает *градиентный метод* (точнее, семейство градиентных методов) поиска стационарных точек дифференцируемой функции. Напомним, что *стационарной* называется точка, в которой $\nabla f(x) = 0$ и которая в соответствии с необходимым условием оптимальности является «подозрительной» на наличие локального экстремума. Таким образом, применяя градиентный метод, находят множество точек локальных максимумов (или минимумов), среди которых определяется максимум (или минимум) глобальный.

Идея данного метода основана на том, что градиент функции указывает направление ее *наиболее быстрого возрастания* в окрестности той точки, в которой он вычислен. Поэтому, если из некоторой текущей точки $x^{(1)}$ перемещаться в направлении вектора $\nabla f(x) = 0$, то функция f будет возрастать, по крайней мере, в некоторой окрестности $x^{(1)}$. Следовательно, для точки $x^{(2)} = x^{(1)} + \lambda \nabla f\left(x^{(1)}\right)$, $(\lambda > 0)$, лежащей в такой окрестности, справедливо неравенство $f\left(x^{1}\right) \le f\left(^{(2)}\right)$. Продолжая этот процесс, мы постепенно будем приближаться к точке некоторого локального максимума (см. *рис. 2.1).*

Однако как только определяется направление движения, сразу же встает вопрос о том, как далеко следует двигаться в этом направлении или, другими словами, возникает проблема выбора шага λ, в рекуррентной формуле

$$x^{(q+1)} = x^{(q)} + \lambda \nabla f\left(x^{(q)}\right) \qquad (2.8)$$

задающей последовательность точек, стремящихся к точке максимума.

В зависимости от способа ее решения различают различные варианты градиентного метода. Остановимся на наиболее известных из них.

Метод наискорейшего спуска

Название метода можно было бы понимать буквально, если бы речь шла о минимизации целевой функции. Тем не менее, по традиции такое название используется и при решении задачи на максимум.

Пусть $f(x) = f(x_l, x_l, \ldots x_n)$ — дифференцируемая функция, заданная на R^n, а $x^{(q)} = \left(x_1^{(q)}, x_2^{(q)}, \ldots, x_n^{(q)}\right)$ — некоторая текущая точка. Оговоримся, что каких-либо общих рекомендаций, касающихся выбора исходной точки (или, как еще говорят, начального приближения) $x^{(0)}$, не существует, однако по возможности она должна находиться близко от искомого оптимального плана x^*. Как уже говорилось выше, если $x^{(q)}$ — нестационарная точка (т. е. $\left|\nabla f\left(x^{(q)}\right)\right| > 0$), то при движении в направлении $\nabla f\left(x^{(q)}\right)$ функция $f(x)$ на некотором промежутке обязательно будет возрастать. Отсюда возникает естественная идея такого выбора шага, чтобы движение в указанном направлении продолжалось до тех пор, пока возрастание не прекратится. Для этого выразим зависимость значения $f(x)$ от шагового множителя $\lambda > 0$, полагая $x = x^{(q)} + \lambda \nabla f\left(x^{(q)}\right)$

$$f(x) = f\left(x^{(q)} + \lambda \nabla f\left(x^{(q)}\right)\right) = \varphi(\lambda) \qquad (2.9)$$

или, в координатной форме,

$$\varphi(\lambda)= f\left(x_1^{(q)} + \lambda \frac{\partial f\left(x^{(q)}\right)}{\partial x_1},..., x_n^{(q)} + \lambda \frac{\partial f\left(x^{(q)}\right)}{\partial x_n} \right) \tag{2.10}$$

Чтобы добиться наибольшего из возможных значений f при движении по направлению $\nabla f\left(x^{(q)}\right)$, нужно выбрать такое значение $\tilde{\lambda}$, которое максимизирует функцию $\varphi(\lambda)$ $\left(\varphi\left(\tilde{\lambda}\right)=\max\limits_{\lambda>0}\varphi(\lambda)\right)$. Для вычисления $\tilde{\lambda}$, используется необходимое условие экстремума $d\varphi\left(\tilde{\lambda}\right)/d\lambda = 0$. Заметим, что если для любого $\lambda >0$ $d\varphi\left(\tilde{\lambda}\right)/d\lambda > 0$, то функция $f(x)$ не ограничена сверху (т. е. не имеет максимума). В противном случае, на основе (2.10) получаем

$$\frac{d\varphi}{d\lambda} = \frac{\partial f\left(x\right)}{\partial x_1} \times \frac{dx_1}{d\lambda} + ... + \frac{\partial f\left(x\right)}{\partial x_n} \times \frac{dx_n}{d\lambda} \tag{2.11}$$

что, в свою очередь, дает

$$\frac{d\varphi}{d\lambda} = \frac{\partial f\left(x\right)}{\partial x_1} \times \frac{\partial f\left(^{(q)}\right)}{\partial x_1} + ... + \frac{\partial f\left(x\right)}{\partial x_n} \times \frac{\partial f\left(x^{(q)}\right)}{\partial x_n} = \nabla f\left(x\right) \nabla f\left(x^{(q)}\right) \tag{2.12}$$

Если считать, что следующая точка $x^{(q+1)}$ соответствует оптимальному значению $\lambda = \tilde{\lambda}$, то в ней должно выполняться условие $d\varphi\left(\tilde{\lambda}\right)/d\lambda = 0$, и $\tilde{\lambda}$ следует находить из условия $\nabla f\left(x^{(q+1)}\right) \nabla f\left(x^{(q)}\right) = 0$ или

$$\nabla f\left(x^{(q)} + \tilde{\lambda} \nabla f\left(x^{(q)}\right)\right) \nabla f\left(x^{(q)}\right) = 0 \tag{2.13}$$

Условие (2.13) означает равенство нулю скалярного произведения градиентов функции f точках $x^{(q+1)}$ и $x^{(q)}$ Геометрически оно может быть интерпретировано как перпендикулярность векторов градиентов функции f в указанных точках, что и показано на *рис. 2.2.* Продолжая геометрическую интерпретацию метода наискорейшего спуска, отметим, что в точке $x^{(q+1)}$ вектор $\nabla f\left(x^{(q+1)}\right)$, будучи градиентом, перпендикулярен линии уровня, проходящей через данную точку. Стало быть, вектор $\nabla f\left(x^{(q)}\right)$ является касательным к этой линии. Итак, движение в направлении градиента $\nabla f\left(x^{(q)}\right)$ следует продолжать до тех пор, пока он пересекает линии уровня оптимизируемой функции.

После того как точка $x^{(q+1)}$ найдена, она становится текущей для очередной итерации. На практике признаком достижения стационарной точки служит достаточно малое изменение координат точек, рассматриваемых на последовательных итерациях. Одновременно с этим координаты вектора $\nabla f\left(x^{(q)}\right)$ должны быть близки к нулю.

Метод дробления шага

Для нахождения шага λ, в методе наискорейшего спуска требуется решить уравнение (2.13), которое может оказаться достаточно сложным. Поэтому часто ограничиваются «подбором» такого значения λ, что $\varphi(\lambda) > \varphi(0)$. Для этого задаются некоторым начальным значением λ_1 (например, $\lambda_1 = 1$) и проверяют условие $\varphi(\lambda_1) > \varphi(0)$. Если оно не выполняется, то полагают

$$\lambda_2 = \frac{1}{2}\lambda_1$$

и т. д. до тех пор, пока не удается найти подходящий шаг, с которым переходят к следующей точке $x^{(q+1)}$. Критерий завершения алгоритма, очевидно, будет таким же, как и в методе наискорейшего спуска.

Индивидуальные задания по теме № 7.

«Методы градиентного поиска»

Студенту предлагается функция двух переменных, экстремум которой требуется найти. Это требуется проделать любым из изученных градиентных методов – на выбор студента. Сделанный выбор требуется обосновать.

Практическое занятие № 8.

«Решение задач нелинейного программирования методом допустимых направлений»

Метод допустимых направлений. Данный метод также называется методом возможных направлений или же по имени автора — методом Зойтендейка, см. [16]. Его основную идею будет удобно продемонстрировать на примере ЗИП с ограничениями в форме неравенств:

$$f(x) \to \max, D = \{x \in R^n | g_i(x_1, x_2, ..., x_n) \le 0, i \in 1:m\} \qquad (2.16)$$

В указанном методе так же, как и в градиентных методах, находится последовательность точек $x^{(0)}, x^{(1)}, ..., x^{(q)}, ...,$ таких, что $f(x^{(q+1)}) \ge f(x^{(q)})$[1]. При этом переход от точки $x^{(q)}$ к точке происходит по некоторому выбранному направлению $s^{(q)}$ с шаговым множителем λ_q:

$$x^{(q+1)} = x^{(q)} + \lambda_q s^{(q)} \qquad (2.17)$$

По отношению к векторам, задающим направления перемещения, вводятся два фундаментальных понятия.

►*Направление s называется* **допустимым** *(возможным) в точке* $x^{(q)} \in D$, *если существует такое* $\lambda > 0$, *что* $x^{(q+1)} = x^{(q)} + \lambda s \in D$.

►*Направление s называется* **прогрессивным** *в точке* $x^{(q)} \in D$*, если существует такое* $\lambda > 0$*, что* $f(x^{(q)} + \lambda s) > f(x^{(q)})$ *для задачи максимицзации и* $f(x^{(q)} + \lambda s) < f(x^{(q)})$ *для задачи минимизации.*

На основе данных определений достаточно просто сформулировать критерий проверки оптимальности точки (так называемый критерий оптимальности в терминах допустимых и прогрессивных направлений):

► *точка* x^* *является оптимальным планом задачи* (2.16)*, если в ней ни одно допустимое направление не является прогрессивным.*

В алгоритме метода допустимых направлений правила выбора точки $x^{(q+1)}$ к которой происходит очередной переход, различаются в зависимости от того, где находится текущая точка $x^{(q)}$. Принципиально возможны две ситуации.

1°. Точка $x^{(q)}$ лежит внутри области *D*, т. е. для всех $i \in 1:m$ $g_i(x^{(q)}) < 0$ (см. *рис. 2.4)*. Очевидно, что для внутренней точки любое направление будет допустимым (при выборе достаточно малого шага), поэтому естественным представляется движение в сторону «гарантированного» возрастания значения функции, а именно в направлении градиента. Значит, для внутренней точки $x^{(q)}$ целесообразно выбрать $s^{(q)} = \nabla f(^{(q)})$.

---

[1] Так как в данных методах при переходе к очередной рассматриваемой точке происходит улучшение значения целевой функции (в частности, для задачи минимизации — уменьшение), их также называют *релаксационными методами.*

Шаговый множитель λ_q выбирается так, чтобы, с одной стороны, новая точка $x^{(q+1)}$ принадлежала D, а с другой — значение целевой функции в ней $f(x^{(q+1)})$ было как можно большим.

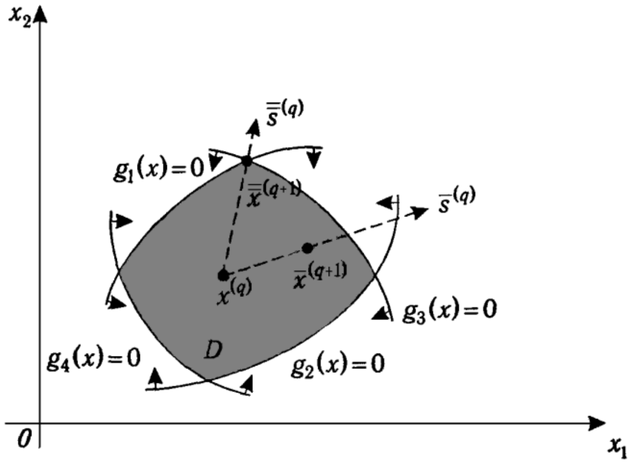


Рис. 2.4

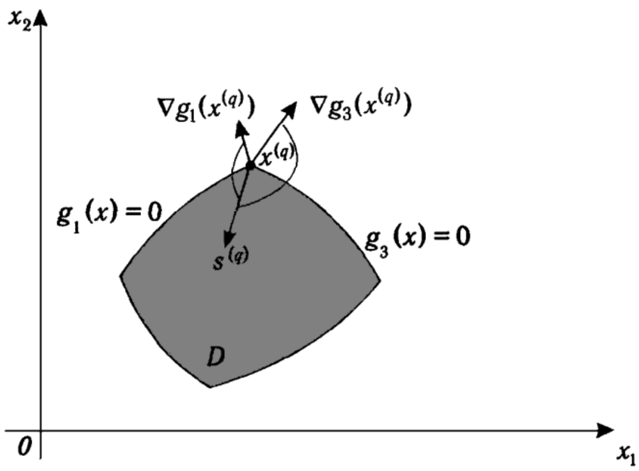


Рис. 2.5

С этой целью сначала найдем промежуток $[0,\Lambda]$ из условия $[0,\Lambda]=\left\{\lambda\middle|\left(x^{(q)}+\lambda s^{(q)}\right)\in D\right\}$, для чего необходимо решить систему неравенств:

$$g_i\left(x^{(q)}+\lambda s^{(q)}\right)\le 0, i\in 1:m \quad (2.18)$$

Зная промежуток $[0,\Lambda]$, определяем значение шагового множителя λ_q из условия максимизации значения функции в направлении $s^{(q)}$:

$$f\left(x^{(q)}+\lambda_q s^{(q)}\right)=\max_{\lambda\in[0,\Lambda]}\left\{f\left(x^{(q)}+\lambda s^{(q)}\right)\right\} \quad (2.19)$$

Вновь найденная точка $x^{(q+1)}$ может находиться или внутри области *D,* или на ее границе. В первом случае (на *рис. 2.4* ему соответствует точка $\overset{-}{x}^{(q+1)}$) переходим к началу данного пункта и повторяем вышеописанные действия, а во втором (точка $\overset{=}{x}^{(q+1)}$ на *рис. 2.4)* — действуем по рассматриваемой далее схеме.

2°. Точка $x^{(q)}$ находится на границе области (см. *рис. 2.5).* Это означает, что одно или несколько неравенств из системы ограничений задачи (2.16) выполняются как строгие равенства: $g_i\left(x^{(q)}\right)=0$. Например, на *рис. 2.5* $g_1\left(x^{(q)}\right)=0$ и $g_3\left(x^{(q)}\right)=0$.

Ограничение, которое в текущей точке выполняется как равенство, называют *активным.* Множество номеров активных ограничений в точке $x^{(q)}$ будем обозначать как $N(x^{(q)})$. В примере, изображенном на *рис. 2.5, $I(x^{(q)})$ = {l,* 3}. Также из рисунка видно, что все допустимые направления, исходящие из точки x^q\ должны образовывать тупые углы с векторами градиентов функций, задающих активные ограничения в данной точке. Последнее условие может быть выражено через задание ограничений на значения скалярных произведений вектора направления s на градиенты функции ограничений:

$$s\nabla g_i\left(x^{(q)}\right)\le 0, i\in I\left(x^{(q)}\right)\cap I_л \tag{2.20}$$

$$s\nabla g_i\left(x^{(q)}\right)< 0, i\in I\left(x^{(q)}\right)\cap I_н \tag{2.21}$$

где $I_л$ — множество номеров индексов линейных ограничений, $I_н$ — множество номеров индексов нелинейных ограничений. Соответственно, $I\left(x^{(q)}\right)\cap I_л$ — номера линейных активных ограничений, а $I\left(x^{(q)}\right)\cap I_н$ — номера нелинейных активных ограничений. Отличие условий (2.20) от условий (2.21) заключается в том, что в случае линейного ограничения направление, образующее прямой угол с градиентом ограничивающей функции (т. е. их скалярное произведение равно нулю), будет заведомо допустимым, а в случае нелинейного ограничения — возможно, нет.

Все возможные направления в точке $x^{(q)}$ образуют так называемый *конус допустимых направлений,* и из них для следующего перехода, очевидно, нужно выбрать прогрессивное. Если такового не существует, то согласно сформулированному выше критерию точка $x^{(q)}$ является оптимальной! Для ускорения максимизации функции желательно, чтобы угол между искомым допустимым прогрессивным направлением $s^{(q)}$ и градиентом целевой функции $\nabla f\left(x^{(q)}\right)$ был как можно меньше или, что то же самое, как можно большей была бы проекция s на $\nabla f\left(x^{(q)}\right)$ (при условиях нормировки вектора $s^{(q)}$*).* Иными словами, желательно, чтобы неравенство $s^{(q)}\nabla f\left(x^{(q)}\right)+\sigma\ge 0$ выполнялось при минимально возможном $\sigma\in R$. Тогда задачу отыскания наилучшего допустимого прогрессивного направления $s^{(q)}$ можно свести к задаче минимизации параметра σ:

$$\sigma\to\min \tag{2.22}$$

при условиях

$$S:\left\{ s\in R^n, \sigma\in R \left| \begin{array}{c} s\nabla f\left(x^{(q)}\right)+\sigma\ge 0, \\ s\nabla g_i\left(x^{(q)}\right)\le 0, i\in I\left(x^{(q)}\right)\cap I_{л,} \\ s\nabla g_i\left(x^{(g)}\right)+\tau\sigma\le 0, i\in I\left(x^{(q)}\right)\cap I_н, \\ s_1^2+s_2^2+\ldots+s_n^2\le 1 \end{array} \right. \right\} \tag{2.23}$$

где $s_1^2+s_2^2+\ldots+s_n^2\le 1$ — условие нормировки, обеспечивающее ограниченность решения;

τ — некоторое достаточно малое число, характеризующее «строгость» выполнения неравенств.

В отличие от всех остальных, последнее условие в системе (2.23) является нелинейным, что существенно усложняет процесс решения задачи (2.22)-(2.23). Поэтому на практике для определения направления $s^{(q)}$ (возможно, не лучшего) переходят от данной задачи к задаче линейного программирования путем замены указанных выше условий нормировки на ограничения вида $-1\le s_j\le 1$:

$$\sigma\to\min \tag{2.24}$$

$$\tilde{S}:\left\{ s\in R^n, \sigma\in R \left| \begin{array}{c} c_1s_1+c_2s_2+\ldots+c_ns_n+\sigma\ge 0, \\ a_{i,1}s_1+a_{i,2}s_2+\ldots+a_{i,n}s_n\le 0, i\in I\left(x^{(q)}\right)\cap I_n, \\ a_{i,1}s_1+a_{i,2}s_2+\ldots+a_{i,n}s_n+\tau\sigma\le 0, i\in I\left(x^{(q)}\right)\cap I_н, \\ -1\le s_j\le 1, j\in 1:n \end{array} \right. \right\} \tag{2.25}$$

где $(c_1,c_2,\ldots,c_n)=\nabla f\left(x^{(q)}\right)$, $\left(a_{i,1},a_{i,2}s_2,\ldots,a_{i,n}\right)=\nabla g_i\left(x^{(q)}\right)$.

После того как прогрессивное направление $s^{(q)}$ найдено, шаговый множитель определяется по методу, описанному в п. 1°.

В заключение отметим, что при практических расчетах алгоритм завершается при достижении такой точки x^*, в которой $\left|\nabla f\left(x^*\right)\right|<\varepsilon$, где ε — достаточно малое число.

Представляется полезным обратить внимание читателя и на | то, что применяемый для решения задач линейного программирования симплекс-метод может быть рассмотрен как частный случай метода допустимых направлений. В частности, этап выбора столбца, вводимого в очередной базис, соответствует определению допустимого прогрессивного направления. Более подробно о такой концепции симплекс-метода можно прочесть в [1].

2.1.6. Пример решения ЗНП методом допустимых направлений. Рассмотрим процесс применения метода допустимых направлений на конкретном примере. Пусть дана ЗНП:

$$f(x) = x_1 + x_2 \to \max \tag{2.26}$$

$$D = \left\{ x \in R^2, \middle| \begin{array}{c} g_1(x) = x_1^2 + x_2^2 - 25 \le 0, \\ g_2(x) = x_2 - 3 \le 0 \end{array} \right\} \tag{2.27}$$

Итерация 1. В качестве начального приближения возьмем точку $x^{(1)}$ = (0,0). Нетрудно заметить, что она удовлетворяет системе неравенств (2.27), т. е. $x^{(1)} \in D$. Для $x^{(1)}$ все неравенства выполняются как строгие, т. е. множество индексов активных ограничений $I\left(x^{(1)}\right) = \varnothing$. Следовательно, в $x^{(1)}$ любое направление является допустимым, и нам остается определить, с каким шагом λ_1, можно двигаться вдоль градиента целевой функции $s^{(1)} = \nabla f\left(x^{(1)}\right) = (1,1)$. Система неравенств типа (2.18), из решения которых определяется интервал допустимых значений для λ, для данной задачи примет вид:

$$\begin{cases} \lambda^2 + \lambda^2 - 25 \le 0 \\ \lambda - 3 \le 0 \\ \lambda \ge 0 \end{cases} \Leftrightarrow \begin{cases} \lambda \le \sqrt{12,5} \\ \lambda \le 3 \\ \lambda \ge 0 \end{cases} \Rightarrow \lambda \in [0;3]$$

Тогда

$$\max_{\lambda \in [0,3]} f\left(x^{(1)} + \lambda s^{(1)}\right) = \max_{\lambda \in [0,3]}\left(\left(x_1^{(1)} + \lambda s_1^{(1)}\right) + \left(x_2^{(1)} + \lambda s_0^{(1)}\right)\right) = \max_{\lambda \in [0,3]}\left((0 + \lambda \times 1) + (0 + \lambda \times 1)\right) = \max_{\lambda \in [0,3]}(2\lambda)$$

достигается при $\lambda_1 = 3$. Отсюда получаем следующую точку

$$x^{(2)} = x^{(1)} + \lambda_1 s^{(1)} = \begin{pmatrix} 0 + 3 \times 1 \\ 0 + 3 \times 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

Итерация 2. Путем подстановки координат точки $x^{(2)}$ в (2.27) определим множество активных ограничений в точке $x^{(2)} : I\left(x^{(2)}\right) = \{2\}$. Соответственно, задача (2.24) - (2.25), которую требуется решить для определения допустимого прогрессивного направления $s^{(2)}$ с учетом того, что $\nabla f\left(x^{(2)}\right) = (1,1)$ и $\nabla g_2\left(x^{(2)}\right) = (0,1)$, примет вид:

$$\sigma \to \min,$$

$$\tilde{S} : \left\{ s \in R^2, \sigma \in R \middle| \begin{array}{c} s\nabla f\left(x^{(2)}\right) + \sigma = s_1 + s_2 + \sigma \ge 0, \\ s\nabla g_2\left(x^{(2)}\right) = 0 \times s_1 + 1 \times s_2 \le 0, \\ -1 \le s_1 \le 1, \\ -1 \le s_2 \le 1 \end{array} \right\}$$

В данном случае оптимальный план ЗЛП находится довольно просто и равен $(\sigma, s_1, s_2)^* = (-1{,}1{,}0)$. Отбросив дополнительную переменную σ, получаем вектор $s^{(2)} = (1{,}0)$, т. е. очередная точка будет определяться как

$$x^{(3)} = x^{(2)} + \lambda s^{(2)} = \begin{pmatrix} 3+\lambda\times 1 \\ 3+\lambda\times 0 \end{pmatrix} = \begin{pmatrix} 3+\lambda \\ 3 \end{pmatrix}$$

Действуя по аналогии с предыдущей итерацией, для определения промежутка допустимых значений шагового множителя λ составляем систему неравенств (2.18):

$$\begin{cases} g_1\left(x^{(2)} + \lambda s^{(2)}\right) = (3+\lambda)^2 + 3^2 - 25 \le 0 \\ g_1\left(x^{(2)} + \lambda s^{(2)}\right) = 3 - 3 \le 0 \\ \lambda \ge 0 \end{cases} \Rightarrow \begin{cases} (3+\lambda)^2 \le 16 \\ \lambda \ge 0 \end{cases} \Rightarrow \begin{cases} -4 \le 3+\lambda \le 4 \\ \lambda \ge 0 \end{cases}$$

Окончательно имеем $\lambda \in [0;1]$.

Тогда

$$\max_{\lambda\in[0,1]} f\left(x^2 + \lambda s^2\right) = \max_{\lambda\in[0,1]}\left(\left(x_1^{(2)} + \lambda s_1^{(2)}\right) + \left(x_2^{(2)} + \lambda s_0^{(2)}\right)\right) = \max_{\lambda\in[0,1]}\left((3+\lambda) + (3)\right) = \max_{\lambda\in[0,1]}(\lambda + 6)$$

достигается при $\lambda_2 = 1$. Отсюда получаем следующую точку $x^{(3)} = (3{,}4)$.

Итерация 3. В точке $x^{(3)}$ множество активных ограничений будет иметь вид $I\left(x^{(3)}\right) = \{1{,}2\}$. Найдем значения градиентов: $\nabla f\left(x^{(3)}\right) = (1{,}1)$, $\nabla g_1\left(x^{(3)}\right) = \left(2x_1^{(3)}, 2x_2^{(3)}\right) = (8{,}6)$ и $\nabla g_2\left(x^{(2)}\right) = (0{,}1)$.

Задача определения допустимого прогрессивного направления (2.24)-(2.25) будет иметь вид:

$$\sigma \to \min,$$
$$\tilde{S} : \left\{ s \in R^2, \sigma \in R \;\middle|\; \begin{array}{c} s\nabla f\left(x^{(3)}\right) + \sigma = s_1 + s_2 + \sigma \ge 0, \\ s\nabla g_1\left(x^{(3)}\right) + \tau\sigma = 8s_1 + 6s_2 + \tau\sigma \le 0, \\ s\nabla g_2\left(x^{(3)}\right) = 0\times s_1 + 1\times s_2 \le 0, \\ -1 \le s_1 \le 1, -1 \le s_2 \le 1 \end{array} \right\}$$

Значение τ из практических соображений следует брать достаточно малым, например $\tau = 0{,}001$. Опуская решение данной задачи, приведем интересующие нас компоненты ее оптимального плана: $s^{(3)} = (0{,}0)$. Итак, не существует прогрессивного направления, исходящего из точки $x^{(3)}$. Таким образом, оптимальный план рассматриваемой задачи (2.26)-(2.27) $x^* = (4{,}3)$, а максимальное значение целевой функции $f^* = x_1^{(3)} + x_2^{(3)} = 7$.

Графическая иллюстрация проведенного процесса решения представлена графически на *рис. 2.6.*

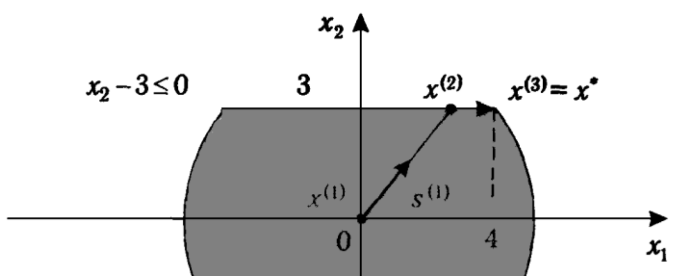


Рис. 2.6

Индивидуальные задания по теме № 8.

«Решение задач нелинейного программирования методом допустимых направлений»

Выдаётся задача нелинейного программирования. Целевая функция линейна, одно из ограничений линейно, второе нелинейно (квадратично). Требуется решить задачу методом допустимых направлений. При этом предполагается использовать компьютер с любым известным студенту пакетом для математического анализа.

Практическое занятие № 9.

«Построение Марковских сетевых моделей»

Переходы системы из состояния в состояние можно рассматривать как происходящие под влиянием некоторых потоков событий.

Поток событий называется **регулярным**, если события следуют одно за другим через строго определенные промежутки времени.

Поток событий называется **стационарным**, если вероятность появления того или иного числа событий на участке времени длиной Δt зависит только от длины участка и не зависит от того, где именно на оси t расположен этот участок. Если обозначить $P_n(t, t+\Delta t)$ вероятность появления n требований (заявок, событий) в промежуток времени $(t, t+\Delta t)$, то свойство стационарности можно записать так: $P_n(t,t+\Delta t) = P_n(\Delta t)$. Стационарность потока соответствует неизменности условий, постоянству работы системы. В этом случае плотность (интенсивность) потока (среднее число требований в единицу времени) постоянна: $\lambda(t) = const$.

Поток событий называется **потоком без последействия**, если вероятность появления на любом участке времени того или иного числа событий не зависит от того, какое число событий попало на другие участки, не пересекающиеся с данным. Это свойство состоит в том, что вероятность $P_n(t,t+\Delta t)$ не зависит от порядка и интенсивности поступления требований до момента t, т.е. в таком потоке текущие требования поступают в систему независимо от предшествующих.

Поток событий называется **ординарным**, если вероятность попадания на элементарный участок Δt двух или более событий пренебрежимо мала по сравнению с вероятностью попадания одного события. Это свойство состоит в том, что практически невозможно одновременное поступление на обслуживание двух или более требований. Если обозначить $P^*(\Delta t)$ вероятность поступления в систему за время Δt более одного требования, то это свойство можно выразить следующим образом: $\lim\limits_{\Delta t \to 0} \dfrac{P^*(\Delta t)}{\Delta t} = 0$.

Ординарный поток без последействия называется **пуассоновским**. Если события образуют пуассоновский поток, то число событий m, падающих на любой участок времени $(t,t+\Delta t)$, распределено по закону Пуассона: $P_m(\Delta t) = \dfrac{a^m}{m!} e^{-a}$, где a – математическое ожидание числа точек, попадающих на участок $(t,t+\Delta t)$: $a = \int\limits_{t}^{t+\Delta t} \lambda(t) dt$.

Если плотность потока $\lambda(t) = const$, то пуассоновский поток называется “стационарным пуассоновским” или **простейшим потоком**.

В **марковском случайном процессе** все потоки событий, переводящие систему из состояния в состояние, являются пуассоновскими.

Для простейшего потока число событий, попадающих на любой участок длины Δt, распределено по закону Пуассона с параметром $a = \lambda \Delta t$.

Расстояние T между двумя соседними событиями в простейшем потоке есть непрерывная случайная величина, распределенная по показательному закону с плотностью $f(t) = \lambda e^{-\lambda t}$ $(t \geq 0)$ и параметрами $m_t = \sigma_t = \dfrac{1}{\lambda}$.

При суммировании (взаимном наложении) большого числа ординарных стационарных потоков (при условии, что складываемые потоки оказывают на сумму приблизительно равномерно малое влияние) с практически любым последействием получается поток, сколь угодно близкий к простейшему.

Если промежуток времени T распределен по показательному закону, то любые сведения о том, сколько времени уже длился этот промежуток, не влияют на закон распределения оставшегося времени. Показательный закон – единственный, обладающий таким свойством.

Если поток событий нестационарен, то его основной характеристикой является мгновенная плотность $\lambda(t)$, являющаяся пределом отношения среднего числа событий, приходящегося на элементарный участок времени $(t, t + \Delta t)$, к длине этого участка, когда последняя стремится к нулю:

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{m(t + \Delta t) - m(t)}{\Delta t} = m'(t),$$

где $m(t)$ – математическое ожидание числа событий на участке $(0, t)$.

Ординарный поток однородных событий называется **потоком с ограниченным последействием** (или **потоком Пальма**), если промежутки времени между последовательностями событий $T_1, T_2, \ldots$ представляют собой независимые случайные величины. Очевидно, простейший поток является частным случаем потока Пальма, в нем расстояния T_i представляют собой независимые случайные величины, распределенные по показательному закону. Нестационарный пуассоновский поток не является потоком Пальма.

Для нестационарного потока (ординарного, без последействия, с переменной плотностью $\lambda(t)$) закон распределения промежутка времени T между соседними событиями выражается плотностью распределения в виде

$$f_{t_0}(t) = \lambda(t_0 + t) \exp\left[-\int_{t_0}^{t_0 + t} \lambda(t) dt \right] \quad (t \geq 0).$$

Потоком Эрланга k-го порядка называется поток, получаемый из простейшего путем “разрежения”, т.е. если сохранить каждую $(k+1)$-ю точку, а остальные удалить. Очевидно, простейший поток можно рассматривать как поток Эрланга нулевого порядка.

При увеличении порядка k $(k = 0, 1, 2, \ldots)$ (и одновременном уменьшении масштаба по оси t, делением на $k+1$) поток Эрланга приближается к регулярному потоку с постоянными интервалами, равными $\dfrac{1}{\lambda}$.

Если $P_{ij}^{(k)}$ – вероятность перехода системы из состояния x_i в состояние x_j на k-м шаге, то вероятность того, что система X будет находиться после k шагов в состоянии x_i выражается формулой:

$$P_i(k) = \sum_{j=1}^{n} P_i(k-1) P_{ij}^{(k)}; \quad i = \overline{1,n}.$$

Индивидуальные задания по теме № 9.

«Построение Марковских сетевых моделей»

По описанию системы (магазин, заправочная станция, ремонтное предприятие) и данным по среднему времени тех или иных процессов требуется стставить Марковскую модель СМО.

Практическое занятие № 10.

«Построение системы дифференциальных уравнений Колмогорова. Понятие предельных вероятностей состояний»

Для описания случайного процесса, протекающего в системе с дискретными состояниями $Z_0, \ldots, Z_n$, часто пользуются вероятностями состояний $P_0(t), P_1(t), \ldots, P_n(t)$, где $P_k(t), \ (k = \overline{0,n})$ – вероятность того, что в момент t система будет находиться в состоянии Z_k.

Очевидно, что $\sum\limits_{k=0}^{n} P_k(t) = 1.$

Если процесс, протекаемый в системе с дискретными состояниями и непрерывным временем, является **марковским**, то для вероятностей состояний $P_k(t), \ (k = \overline{0,n})$ можно составить систему линейных дифференциальных уравнений Колмогорова.

Если имеется размеченный граф состояний (рис.4.3) (здесь над каждой стрелкой, ведущей из состояния в состояние, проставлена интенсивность потока событий, переводящего систему из состояния в состояние по данной стрелке), то систему дифференциальных уравнений для вероятностей $P_k(t)$ можно сразу написать, пользуясь следующим простым **правилом**.

В левой части каждого уравнения стоит производная $\dfrac{dP_k(t)}{dt}$, а в правой части – столько членов, сколько стрелок связано непосредственно с данным состоянием; если стрелка ведет **в** данное состояние, член имеет знак плюс, если ведет **из** данного состояния, член имеет знак минус.

Каждый член равен плотности потока событий, переводящего систему по данной стрелке, умноженной на вероятность того состояния, из которого исходит стрелка.

Начальные условия для интегрирования отражают состояние системы в начальный момент. Если, например, система при $t = 0$ была в состоянии Z_i, то полагают

$$P_i(0) = 1; \ \ P_k(0) = 0 \ \ (k \neq i).$$

Предельным режимом для системы X называется случайный процесс, устанавливающийся в системе при $t \to \infty$. Если в числе состояний системы имеются состояния без выхода, то при $t \to \infty$ система с практической достоверностью окажется в одном из них.

Если все потоки событий, переводящие систему из состояния в состояние, стационарны $(\lambda_{ik} = const)$, общее число состояний конечно и состояний без выхода нет, то предельный режим существует и характеризуется **предельными вероятностями** $P_k \ \ (k = \overline{0,n})$.

Индивидуальные задания по теме № 10.

«Построение системы дифференциальных уравнений Колмогорова. Понятие предельных вероятностей состояний»

Для полученной в предыдущей теме модели СМО составить систему дифференциальных уравнений Колмогорова. Найти предельные вероятности состояний.

Практическое занятие № 11.

«Построение модели и определение основных характеристик СМО с отказами»

Классификацию широкого разнообразия СМО можно провести по различным признакам (табл.4.1).

Системы массового обслуживания с ожиданием бывают разомкнутые и замкнутые. **Разомкнутая (открытая) система** – это система с неограниченным источником потока требований (например, покупатели в магазинах, пассажиры в метро и т.д.).

Замкнутая – это система, в которой поток требований ограничен. Замкнутые системы обслуживают конечное число требований. Как только требования обслужены, они возвращаются в источник (рис.4.2).

Задачи такого типа часто встречаются при эксплуатации машин (оборудования), когда отремонтированные машины возвращаются в цех и становятся источником новых требований.

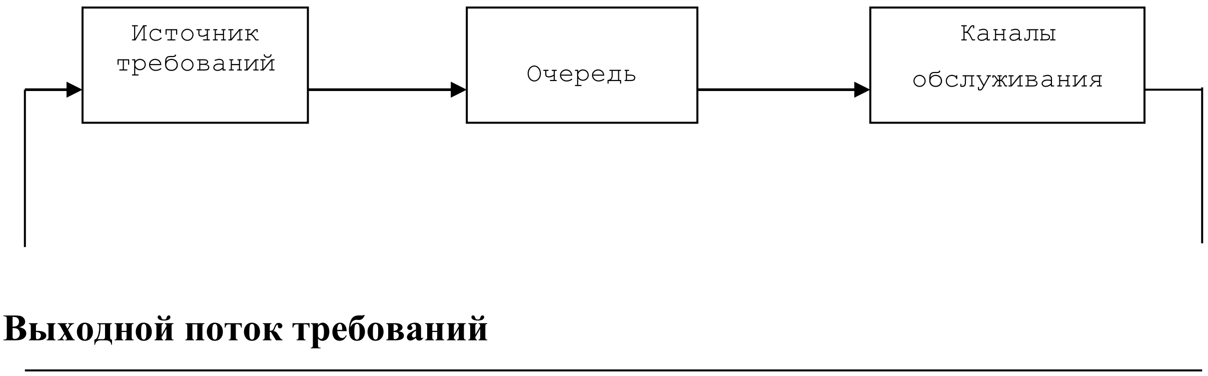


Рис. 4.2. Схема работы замкнутых систем

СМО называется **системой с ожиданием**, если заявка, заставшая все каналы занятыми, становится в очередь и ждет, пока не освободится какой-нибудь канал.

Если ожидание заявки в очереди ничем не ограничено, то система называется "**чистой системой с ожиданием**" или с неограниченным ожиданием. Если ожидание ограничено какими-то условиями, то система называется "**системой смешанного типа**". Это промежуточный случай между чистой системой с отказами и чистой системой с ожиданием.

Ограничения, наложенные на ожидание, могут быть различного типа. Часто бывает, что ограничение накладывается на время ожидания заявки в очереди, т.е. оно ограничено сверху каким-то сроком $T_{ож}$, который может быть как строго определенным, так и случайным. При этом ограничивается только срок ожидания в очереди, а начатое обслуживание доводится до конца, независимо от того, сколько времени продолжалось ожидание (например, клиент в парикмахерской, сев в кресло, обычно уже не уходит до конца обслуживания).

В других задачах возможно ограничение не на время ожидания в очереди, а на общее время пребывания в системе (например, воздушная цель может пробыть в зоне стрельбы лишь ограниченное время и покидает ее независимо от того, кончился обстрел или нет). Часто встречаются ограничения на длину очереди, т.е. заявка становится в очередь только в том случае, если длина очереди не слишком велика (торговые предприятия, ремонтные мастерские с ограниченными площадями и т.п.).

Индивидуальные задания по теме № 11.

«Построение модели и определение основных характеристик СМО с отказами»

Для построенной в разделе 9 модели найти характеристики СМО с отказами.

Практическое занятие № 12.

«Построение модели и определение основных характеристик СМО с конечными и бесконечными очередями»

Если обслуживание производится поэтапно некоторой последовательностью каналов, то такую СМО называют **многофазной**.

В **СМО со «взаимопомощью»** между каналами одна и та же заявка может одновременно обслуживаться двумя и более каналами. Например, один и тот же вышедший из строя станок могут обслуживать два рабочих сразу. Такая «взаимопомощь» между каналами может иметь место как в открытых, так и в замкнутых СМО.

В **СМО с ошибками** заявка, принятая к обслуживанию в системе, обслуживается не с полной вероятностью, а с некоторой вероятностью $P \neq 1$; другими словами, могут иметь место ошибки в обслуживании, результатом которых является то, что некоторые заявки, пошедшие СМО и якобы «обслуженные», в действительности остаются не обслуженными из-за «брака» в работе СМО.

Примерами таких систем могут быть: справочные бюро, иногда выдающие неправильные справки и указания; корректор, могущий пропустить ошибку или неверно ее исправить; телефонная станция, иногда соединяющая абонента не с тем номером; торгово-посреднические фирмы, не всегда качественно и в срок выполняющие свои обязательства, и т.д.

Для анализа процесса, протекающего в СМО, существенно знать **основные параметры системы**: число каналов n, интенсивность потока заявок λ, производительность каждого канала μ (среднее число заявок, обслуживаемое в единицу времени каналом), условия образования очереди, интенсивность ухода заявок из очереди или системы.

Отношение $K_з = \dfrac{\lambda}{\mu n} = \dfrac{\alpha}{n}$ называют **коэффициентом загрузки системы**. Часто рассматриваются только такие системы, в которых $K_з < 1$.

Время обслуживания в СМО может быть как случайной, так и не случайной величиной. На практике это время чаще всего принимается распределенным по показательному закону $f(t) = \mu e^{-\mu t} \quad (t \geq 0)$, $\quad \mu = \dfrac{1}{\bar{t}_{обс}}$.

Основные характеристики СМО сравнительно мало зависят от вида закона распределения времени обслуживания, а зависят главным образом от среднего значения $\bar{t}_{обс}$. Поэтому часто пользуются допущением, что время обслуживания распределено по показательному закону.

Допущения о пуассоновском характере потока заявок и показательном распределении времени обслуживания (которые мы будем предполагать впредь) ценны тем, что позволяют применить в теории массового обслуживания аппарат так называемых марковских случайных процессов.

Эффективность систем обслуживания в зависимости от условий задач и целей исследования можно характеризовать большим числом разных количественных показателей.

Наиболее часто применяются следующие **показатели**:

1. Вероятность того, что обслуживанием заняты k каналов – P_k. Частным случаем является P_0 – вероятность того, что все каналы свободны.

2. Вероятность отказа заявки в обслуживании $P_{отк}$.

3. Среднее число занятых каналов $N_з = \sum_{k=1}^{n} kP_k$ характеризует степень загрузки системы.

4. Среднее число каналов, свободных от обслуживания:

$$N_0 = \sum_{k=0}^{n-1}(n-k)P_k = n - N_з .$$

5. Коэффициент (вероятность) простоя каналов $P_{пр} = \dfrac{N_0}{n}$.

6. Коэффициент загрузки оборудования (вероятность занятости каналов)

$$P_з = \frac{N_з}{n}.$$

7. Относительная пропускная способность q – средняя доля поступивших заявок, обслуживаемая системой, т.е. отношение среднего числа заявок, обслуживаемых системой в единицу времени, к среднему числу поступающих за это время заявок.

8. Абсолютная пропускная способность A, т.е. число заявок (требований), которое может обслужить система за единицу времени:

$$A = \lambda(1 - P_{отк}) = \lambda q .$$

9. Среднее время простоя канала

$$\bar{t}_{пр} = \frac{1 - P_з}{\mu P_з}.$$

Для систем **с ожиданием** используют дополнительно характеристики:

10. Среднее время ожидания требований в очереди $\bar{t}_{ож}$.

11. Среднее время пребывания заявки в СМО $\bar{t}_c = \bar{t}_{ож} + \bar{t}_{обс}$.

12. Средняя длина очереди $N_{оч} = \lambda \bar{t}_{ож}$.

13. Среднее число заявок в сфере обслуживания (в СМО)

$$N_c = \sum_{k=1}^{\infty} kP_k = N_{оч} + N_з = \lambda \bar{t}_c .$$

14. Вероятность того, что время пребывания заявки в очереди не продлится больше определенного времени.

15. Вероятность того, что число требований в очереди, ожидающих начала обслуживания, больше некоторого числа.

Кроме перечисленных критериев при оценке эффективности систем могут быть использованы **стоимостные показатели**:

$C_{об}$ – стоимость обслуживания каждого требования в системе;

$C_{ож}$ – стоимость потерь, связанных с ожиданием в единицу времени;

$C_у$ – стоимость убытков, связанных с уходом требований из системы;

$C_к$ – стоимость эксплуатации канала системы в единицу времени;

$C_{пк}$ – стоимость единицы простоя канала.

При выборе оптимальных параметров системы по экономическим показателям можно использовать следующую **функцию стоимости потерь**:

а) для систем с неограниченным ожиданием

$C_п = (C_{ож}N_{оч} + C_{пк}N_0 + C_кN_з)T$, где T – интервал времени;

б) для систем с отказами $C_п = (C_кN_з + C_{пк}N_0 + C_уP_{отк}\lambda)T$;

в) для смешанных систем $C_п = (C_{пк}N_0 + C_{ож}N_{оч} + C_уN_{отк}\lambda + C_кN_з)T$.

Варианты, в которых предусматривается строительство (ввод) новых элементов системы (например, каналов обслуживания), обычно сравниваются по приведенным затратам $C_{пр}$.

Приведенные затраты по каждому варианту есть сумма текущих затрат (себестоимости) и капитальных вложений, приведенных к одинаковой размерности в соответствии с нормативом эффективности, например:

$C_{пр} = C_п + E_нC_{кап}$ (приведенные затраты за год);

$C_{пр} = C_пT_н + C_{кап}$ (приведенные затраты за срок окупаемости),

где $C_п$ – текущие затраты (себестоимость) по каждому варианту, р.;

$E_н$ – отраслевой нормативный коэффициент экономической эффективности капитальных вложений (обычно $E_н$ = 0,15 - 0,25);

$C_{кап}$ – капитальные вложения по каждому варианту, р.;

$T_н$ – нормативный срок окупаемости капитальных вложений, лет.

Выражение $C_{кап} + C_пT_н$ есть сумма текущих и капитальных затрат за определенный период. Их называют **приведенными**, так как они относятся к фиксированному отрезку времени (в данном случае к нормативному сроку окупаемости).

Показатели $C_п$ и $C_{пр}$ могут применяться как в виде суммы капитальных вложений и себестоимости готовой продукции, так и в виде **удельных капитальных вложений** на единицу продукции и себестоимости единицы продукции.

Индивидуальные задания по теме № 12.

«Построение модели и определение основных характеристик СМО с конечными и бесконечнымиочередями»

Для построенной в разделе 9 модели найти характеристики СМО с очередями.

Практическое занятие № 13.

«Построение модели и определение основных характеристик замкнутых СМО, решение оптимизационных задач»

Пример На станцию текущего ремонта автомашин поступает простейший поток заявок с интенсивностью $\lambda = 0{,}5$ машины в час. Имеется одно помещение для ремонта. Во дворе станции могут одновременно находиться, ожидая в очереди, не более трех машин. Среднее время ремонта одной машины 2 ч. Для улучшения обслуживания были сделаны два предложения:

1) дополнительно построить одно помещение для ремонта;

2) дополнительно построить два помещения для ремонта.

Строительство одного помещения для ремонта автомашин стоит 200 тыс.р. Потери от отказа в своевременном обслуживании одной машины составляют 400 р./год. Потери от простоя одного помещения (канала) составляют 20 р./ч. Из трех возможных вариантов системы (одно, два, три помещения) необходимо выбрать лучший по критерию минимальных приведенных затрат ($E_H = 0{,}2$).

Решение. Все величины приведем к годовому периоду времени. Имеем 0,5 поступлений (заявок) в час, следовательно, за сутки их будет 24·0,5=12, а за год 12·365=4380. Имеем СМО смешанного типа с ограничением по длине очереди. Ее характеристики: $\lambda = 0{,}5;\quad \mu = 0{,}5;\quad \alpha = \lambda/\mu = 1;\quad m = 3;\quad n = 1.$

1. Определим вероятности отказов в рассматриваемых случаях:

а) $n=1$: $P_{отк} = P_{n+m} = \dfrac{\alpha^n}{n!}\left(\dfrac{\alpha}{n}\right)^m \left[\sum\limits_{k=0}^{n}\dfrac{\alpha^k}{k!} + \dfrac{\alpha^n}{n!}\sum\limits_{s=1}^{m}\left(\dfrac{\alpha}{n}\right)^s\right]^{-1} = \dfrac{1}{1+1+3} = 0{,}2;$

б) $n=2$: $P_{отк} = P_{2+3} = \dfrac{1/16}{1+1+1/2+1/4+1/8+1/16} = \dfrac{1}{47} = 0{,}021;$

в) $n=3$: $P_{отк} = P_{3+3} = \dfrac{1/162}{1+1+1/2+1/6+1/6(1/3+1/9+1/27)} = \dfrac{1}{445} = 0{,}0022.$

2. Определим простои системы:

а) $n=1$: $P_{пр} = P_0 = \left[\sum\limits_{k=0}^{n}\dfrac{\alpha^k}{k!} + \dfrac{\alpha^n}{n!}\sum\limits_{s=1}^{m}\left(\dfrac{\alpha}{n}\right)^s\right]^{-1} = 0{,}2.$

Годовой фонд рабочего времени составит $T_ч = 365 \cdot 24 = 8760$.

Простои за год составят 8760·0,2 = 1752 ч.

б) $n=2$. Здесь простои помещений за год будут складываться следующим образом: $T_{пр} = (2P_0 + P_1)T_ч$.

Имеем $P_0 = \frac{16}{47} = 0{,}34; \quad P_1 = \frac{\alpha}{1!} P_0 = 0{,}34.$

Потери за год составят $8760 \cdot (2 \cdot 0{,}34 + 0{,}34) = 8760 \cdot 1{,}02 = 8935$ ч.

в) $n = 3.$ Простои за год в этом случае будут равны

$$T_{пр} = (3P_0 + 2P_1 + P_2)8760 \text{ ч.}$$

Имеем $P_0 = \frac{162}{445}; \quad P_1 = \frac{\alpha}{1!} P_0 = \frac{162}{445};$

$$P_2 = \frac{\alpha^2}{2!} P_0 = \frac{81}{445}; \quad T_{пр} = 8760 \cdot \left(\frac{3 \cdot 162}{445} + \frac{2 \cdot 162}{445} + \frac{81}{445} \right) = \frac{891 \cdot 8760}{445} = 17540.$$

Конечные результаты расчетов сведем в табл.4.10.

Таблица 4.10

Результаты решения задачи

Кол-во помещений для ремонта	Вероятность отказа в обслуживании	Простои помещений за год, ч.	Количество отказов за год	Потери от отказов за год, руб.	Потери от простоев помещений, руб.	Дополнительные затраты, руб.	Приведенные затраты, руб.
1	0,20	1752	876	350400	35040	0	385440
2	0,021	8935	93	37200	178700	200000	255900
3	0,0022	17540	9,64	3856	350800	400000	434656

Вывод: целесообразно иметь два помещения для ремонта.

Индивидуальные задания по теме № 13.

«Построение модели и определение основных характеристик замкнутых СМО, решение оптимизационных задач»

Для ранее рассмотренной задачи выбрать оптимальное количество каналов обслуживания

Практическое занятие № 14.

«Построение моделей матричных игр, поиск оптимальных стратегий»

Каждый из игроков имеет свой набор стратегий. Рассмотрим ситуацию в общем виде. Пусть игроку **A** доступно m стратегий, которые обозначим как $A_1, A_2, \ldots, A_m$. Соответственно игроку **B** доступно n стратегий, которые обозначим как $B_1, B_2, \ldots, B_n$. Таким образом первый игрок может выбрать одну из своих стратегий A_i, $1 \le i \le m$, а второй – одну из своих стратегий B_j, $1 \le j \le n$. Исход игры таким образом будет зависеть от пары стратегий, выбранных игроками – (A_i, B_j).
Рассмотрим все возможные пары стратегий (A_i, B_j), $i=1,\ldots,m$, $j=1,\ldots,n$. Каждой из них поставим в соответствие выигрыш первого игрока a_{ij}, Выигрыш второго всегда можно найти как $b_{ij}=-a_{ij}$, Набор $m \times n$ таких значений и образует матрицу игры, которую также называют платёжной матрицей.
Матрица игры может быть представлена либо в виде таблицы, либо в виде матрицы. Таблица используется в основном тогда, когда игру для решения преобразуют и упрощают, т.е. номера стратегий могут идти не по порядку. Она выглядит следующим образом

Таблица 2.1. Общий вид матрицы игры

A_i \ B_j	B_1	B_2	…	B_n
A_1	a_{11}	a_{12}	…	a_{1n}
A_2	a_{21}	a_{22}	…	a_{2n}
…	…	…	…	…
A_m	a_{m1}	a_{m2}	…	a_{mn}

Для упрощения, например при записи условий задачи, та же игра может быть записана в виде обычной матрицы, например:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Рассмотрим несколько примеров игр.

Пример 1. «В какой руке?». Игрок **A** берёт монету и прячет её в правом или левом кулаке. Игрок **B** должен угадать, в какой руке монета. Если он угадал, то выиграл 1 рубль (а игрок **A** этот рубль проиграл), если не угадал, то 1 рубль выиграл игрок **A**. Строим платежную матрицу. Игра состоит всего из двух личных ходов. У игрока **A** две стратегии: A_1 – взять монету в правую руку; A_2 – в левую. У игрока **B** тоже две стратегии: B_1 – показать на правую руку; B_2 – на левую. Получаем игру 2×2, её матрица имеет вид:

A_i \ B_j	B_1	B_2
A_1	-1	1
A_2	1	-1

Пример 2. «Камень, ножницы, бумага». Каждый из игроков **A** и **B** одновременно и независимо от другого показывает руку, сложенную в одну из упомянутых выше фигур. Выигрыш определяется по следующей схеме: камень тупит ножницы, ножницы режут бумагу, бумага заворачивает камень. Выигрыш даёт игроку +1, проигрыш –1. Если оба игрока показали одинаковые фигуры, то «ничья», то есть 0. Таким образом каждый игрок имеет по три стратегии. Обозначим их одинаково: A_1 и B_1 – камень, A_2 и B_2 – ножницы, A_3 и B_3 – бумага. Матрица игры в итоге примет вид:

B_j / A_i	B_1	B_2	B_3
A_1	0	1	–1
A_2	–1	0	1
A_3	1	–1	0

Пример 3. «Пять пальцев». Каждый из игроков А и В одновременно и независимо от другого показывает от одного до пяти пальцев. Подсчитывается их сумма, показанная игроками. Это число от 2 до 10. Если общее число пальцев чётное, то выигрывает игрок **A**, если нечётное, то **B**. Численно выигрыш равен суммарному числу показанных пальцев. Построим матрицу игры. У каждого игрока по пять стратегий – показывать 1, 2, 3, 4 или 5 пальцев. Пусть номер стратегии равняется количеству пальцев, показываемому данным игроком. Получается игра 5×5, матрица которой имеет вид:

B_j / A_i	B_1	B_2	B_3	B_4	B_5
A_1	2	–3	4	–5	6
A_2	–3	4	–5	6	–7
A_3	4	–5	6	–7	8
A_4	–5	6	–7	8	-9
A_5	6	–7	8	-9	10

Теперь рассмотрим более сложную ситуацию, когда кроме личных имеются и случайные ходы. При этом исход операции для каждой пары стратегий не может быть определён однозначно. Для построения модели потребуется вспомнить теорию вероятностей. Пример взят из [1].

Пример 4. Игра «два бомбардировщик и истребитель».

Сторона **A** посылает в район расположения некоторого объекта противника **B** два бомбардировщика – I и II. I летит спереди, II сзади. Один из бомбардировщиков (заранее неизвестно какой) несёт бомбу; другой выполняет только функцию сопровождения. При подлёте к объекту бомбардировщики встречает истребитель противника. Если истребитель атакует задний бомбардировщик, то по нему ведут огонь пушки только этого бомбардировщика, поражающие истребитель с вероятностью 0,3. Если же истребитель атакует передний бомбардировщик, по нему ведут огонь пушки как переднего, так и заднего бомбардировщика; совместно они поражают его с вероятностью $1\text{-}(1\text{-}0{,}3)^2\text{=}0{,}51$.

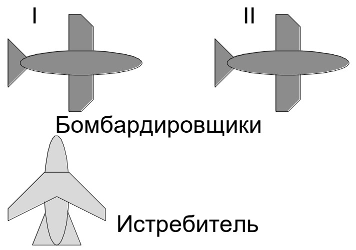


Рисунок 2.1.1.
к примеру 4

Если истребитель не сбит ответным огнем бомбардировщика то он поражает выбранную им цель с вероятностью 0,8. У истребителя есть время только на одну атаку. Задача стороны **A** – донести бомбу до объекта. Задача стороны **B** – воспрепятствовать этому. Требуется найти оптимальные стратегии сторон. Рассмотрим и обозначим чистые стратегии игроков. для стороны **A** два варианта – какой бомбардировщик сделать носителем. A_1 – бомбу везёт первый самолёт; A_2 – второй. для стороны **B** – какой бомбардировщик атаковать. B_1 – атаковать первый; B_2 – второй.

Составим матрицу игры, для чего найдем средний выигрыш первого игрока при каждой из возможных комбинаций стратегий. Выигрышем в данном случае будет являться вероятность того, что бомба будет доставлена к объекту, т.е. объект будет уничтожен.

1. (A_1,B_1) – носитель I, атакуется I.

Вероятность успеха складывается из двух независимых событий. Либо истребитель сбит – тогда бомба доставляется к объекту. Либо не сбит, но при этом в свою очередь промахнулся. Вероятность первого события, того, что оба бомбардировщика вместе поразят истребитель равна 0,51. Вероятность, что ни один не попадёт, равна +(1-0,51). Вероятность, что истребитель промахнётся, равна (1-0,8). В итоге

$$\mathbf{a}_{11}=0{,}51+(1-0{,}51)(1-0{,}8)=0{,}608$$

2. (A_2,B_2) – носитель II, атакуется II. Рассуждая аналогично, получим

$$\mathbf{a}_{22}=0{,}3+0{,}7\text{x}0{,}2=0{,}44$$

3. Рассмотрим сразу обе ситуации (A_1,B_2) и (A_2,B_1) – носитель I, атакуется II и носитель II, атакуется I. Т.е. истребитель атакует не тот бомбардировщик, который везёт бомбу. Этом случае для успеха операции не принципиально, сбит ли истребитель и сбит ли тот бомбардировщик, который он атаковал. Бомба в любом случае доставляется к объекту.

$$\mathbf{a}_{21}=1 \text{ и } \mathbf{a}_{12}=1$$

Матрица игрытаким образом имеет вид

$\mathbf{B_j}$ / $\mathbf{A_i}$	B_1	B_2
A_1	0,608	1
A_2	1	0,44

Напомним, что элементы данной матрицы – вероятности. В каждом конкретном случае исход будет точно определён. Бомба либо будет доставлена к объекту, либо не будет.

Индивидуальные задания по теме № 14.

«Построение моделей матричных игр, поиск оптимальных стратегий»

По имеющемуся описанию требуется составить модель матричной игры и её оптимальное решение.

Практическое занятие № 15.

«Построение моделей биматричных игр, поиск точек равновесия по Нэшу, максиминных стратегий и стратегий угроз»

Достаточно часто на практике возникают ситуации, когда выигрыш одного игрока не равен в точности проигрышу другого. Например: 2 фирмы извлекают прибыль из нескольких рынков, и если одна фирма своими действиями помешает другой получить какой-то доход, то это вовсе не означает, что она сама этот доход получит. Такие игры называют также неантагонистическими или играми с ненулевой суммой.

Пусть игроку **A** доступно $\boldsymbol{m}$ стратегий, которые обозначим как $\boldsymbol{A_1, A_2, \ldots, A_m}$. Соответственно игроку **B** доступно $\boldsymbol{n}$ стратегий, которые обозначим как $\boldsymbol{B_1, B_2, \ldots, B_n}$. Если игрок **A** выберет стратегию $\boldsymbol{A_i}$, а $\boldsymbol{B}$ –стратегию $\boldsymbol{B_j}$, то эта пара стратегий ($\boldsymbol{A_i, B_j}$) определяет выигрыш каждого из игроков.

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

Матрица **A** определяет выигрыш игрока **A** для каждой пары стратегий.

$$B = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{pmatrix}$$

Матрица $\boldsymbol{B}$ определяет выигрыш игрока $\boldsymbol{B}$ при той же паре стратегий. В ряде игр эти матрицы записываются в виде общей матрицы:

$$\begin{pmatrix} (a_{11}, b_{11}) & (a_{12}, b_{12}) & \ldots & (a_{1n}, b_{1n}) \\ (a_{21}, b_{21}) & (a_{22}, b_{22}) & \ldots & (a_{2n}, b_{2n}) \\ \ldots & \ldots & \ldots & \ldots \\ (a_{m1}, b_{m1}) & (a_{m2}, b_{m2}) & \ldots & (a_{mn}, b_{mn}) \end{pmatrix}$$

Очевидно, что все биматричные игры можно рассматривать как частный случай матричных, где $\boldsymbol{b_{ij}=-a_{ij}}$. Соответственно если хотя бы для одной пары стратегий ($\boldsymbol{A_i, B_j}$) это условие не выполняется, то игру можно рассматривать только как биматричную.

Особенность биматричных игр – в отличие от матричных, нет «непримиримого противоречия» между игроками. Поэтому в принципе возможны их согласованные действия. В связи с этим игры делят на кооперативные, в которых игроки могут договариваться о совместных действиях и осуществлять их, и некооперативные – когда возможность договариваться не предусмотрена. Для начала рассмотрим некооперативные игры.

Второй важной особенностью является то, что для биматричных игр нет единого понимания того, что именно считать «оптимальным решением». Есть несколько разных подходов, которые мы и рассмотрим.
Третья особенность заключается в том, что наличие решения в чистых стратегиях не отрицает возможности решения и в смешанных стратегиях. Поэтому не выделяется отдельным этапом поиск решения в чистых стратегиях. Сразу ищем в смешанных, то есть в виде $S_A^* = (p_1,\dots,p_m)$, $S_B^* = (q_1,\dots,q_n)$. Чистые стратегии таким образом будем искать только как частный случай смешанных. Паре стратегий будет соответствовать также пара выигрышей. Мы отдельно рассматриваем выигрыши первого и второго игроков:

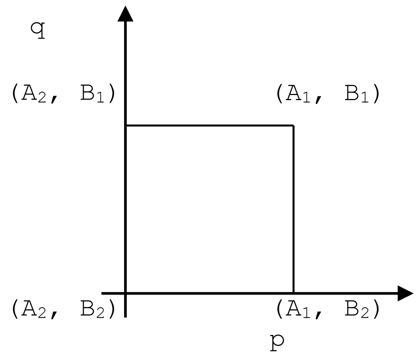
$$H_A = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} p_i q_j, \qquad H_B = \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} p_i q_j. \tag{1}$$

Определение. Точкой равновесия по Нэшу называется пара стратегий S_A^* и S_B^*, обладаюшие следующим свойством: если игрок отступает от своей равновесной стратегии в ситуации, когда другой игрок придерживается своей равновесной стратегии, то этот игрок не получает никаких преимуществ, отклоняясь от своей равновесной стратегии, выигрыш уменьшается или в лучшем случае остаётся тем же.
Такой критерий называют ещё критерием «чистого эгоизма». Игрок, отступая от точки равновесия, теряет в своём выигрыше. Что при этом происходит с выигрышем другого игрока, вообще никак не учитывается. Он может уменьшиться, остаться прежним или вообще увеличиться. В данном случае это не имеет значения.
Для поиска точек равновесия сначала рассмотрим наиболее простой вариант – игры, в которых у каждого игрока только две стратегии.

Биматричные игры 2x2

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

В данном случае любую смешанную стратегию игрока можно однозначно задать одним числовым показателем. Для первого игрока это будет $A: 0 \le p \le 1$. Соответственно вероятности будут равны $p_1 = p,$ и $p_2 = 1 - p$. Крайние значения дадут нам $p = 0 \Rightarrow A_1$ (чистая стратегия), $p = 1 \Rightarrow A_2$ (чистая стратегия). Если же $0 < p < 1 \Rightarrow$ смешанная стратегия.
Аналогично для второго игрока это будет $B: 0 \le q \le 1$. Здесь получаем вероятности $q_1 = q,$ и $q_2 = 1 - q$. При значениях $q = 0 \Rightarrow B_1$ (чистая стратегия), $q = 1 \Rightarrow B_2$ (чистая стратегия), и $0 < q < 1 \Rightarrow$ смешанная стратегия.
Любая пара стратегий игроков задаётся таким образом парой координат точки (p, q). Всё множество допустимых стратегий составляет единичный квадрат. Его вершины – 4 пары чистых стратегий. Внутренние точки – пары смешанных стратегий. Рёбрам соответствует ситуация, когда у одного из игроков чистая стратегия, у другого смешанная…



Точки равновесия будем искать в виде (p*, q*). Средний выигрыш игроков можно вычислить по формулам:

$$H_A(p,q) = a_{11}pq + a_{12}p(1-q) + a_{21}(1-p)q + a_{22}(1-q)(1-p)$$
$$H_B(p,q) = b_{11}qp + b_{12}p(1-q) + b_{21}(1-p)q + b_{22}(1-p)(1-q) \quad (2)$$

Запишем в приведённых терминах требование равновесия по Нэшу.

Определение. Будем говорить, что пара чисел (p^*, q^*), $0 \le q^* \le 1$, $0 \le p^* \le 1$, определяет равновесную стратегию, если $\forall (p, q)$ подчинённых условиям $0 \le p \le 1$, $0 \le q \le 1$ одновременно выполняются следующие равенства:

$$H_A(p,q^*) \le H_A(p^*,q^*) \quad (3)$$
$$H_B(p^*,q) \le H_B(p^*,q^*) \quad (4)$$

Неравенства (3)-(4) очень слабо пригодны для построения алгоритма поиска решения, поскольку слишком много надо сравнивать.

Однако, если рассмотреть формулы (1), (2), можно Заметить, что они зависят от p и q «почти линейно». Имеется ещё произведение p q. Отсюда можно вывести более простые условия.

Теорема. требования

$H_A(p,q^*) \le H_A(p^*,q^*)$, $H_B(p^*,q) \le H_B(p^*,q^*)$,

равносильны выполнению неравенств:

$$\begin{cases} H_A(0,q^*) \le H_A(p^*,q^*) \\ H_A(1,q^*) \le H_A(p^*,q^*) \end{cases} \quad (5)$$

$$\begin{cases} H_B(p^*,0) \le H_B(p^*,q^*) \\ H_B(p^*,1) \le H_B(p^*,q^*) \end{cases} \quad (6)$$

Эти неравенства позволяют построить уже вполне работоспособный алгоритм поиска решения, поскольку требуется всего четыре сравнения.

Алгоритм поиска решения Средний выигрыш игроков 1и 2 запишем в удобной форме:

$$H_A(p,q) = (a_{11} - a_{12} - a_{21} + a_{22})pq + (a_{12} - a_{22})p + (a_{21} - a_{22})q + a_{22} \quad (7)$$

$$H_B(p,q) = (b_{11} - b_{12} - b_{21} + b_{22})pq + (b_{12} - b_{22})p + (b_{21} - b_{22})q + b_{22} \quad (8)$$

Если в неравенствах (5)-(6) из правой части вычесть левую, то полученные разности должны быть больше нуля.

$H_A(p,q)=H_A(1,q)=(a_{11}-a_{12}-a_{21}+a_{22})pq+(a_{12}-a_{22})p+(a_{21}-a_{22})q+a_{22}$ Обозначим
$-(a_{11}-a_{12}-a_{21}+a_{22})q+a_{22}-a_{12}-(a_{21}-a_{22})q-a_{22}$
$\alpha=a_{22}-a_{12}$
$C=a_{11}-a_{12}-a_{21}+a_{22}$
С учётом этих обозначений получим
$H_A(p,q)-H_A(1,q)=(p-1)(Cq-\alpha)\geq 0$
$H_A(p,q)-H_A(0,q)=p(Cq-\alpha)\geq 0$
Если аналогичный вывод поделать для H_B, введя обозначения:
$D=b_{11}-b_{12}-b_{21}+b_{22}$,
$\beta=b_{22}-b_{21}$
то получим:
$H_B(p,q)-H_B(p,1)=(q-1)(Dp-\beta)\geq 0$
$H_B(p,q)-H_B(p,0)=q(Dp-\beta)\geq 0$
Таким образом, для того, чтобы в биматричной игре пара стратегий (p,q) определяла равновесие необходимо и достаточно, чтобы одновременно выполнялись следующие неравенства:

$$\begin{cases}(p-1)(Cq-\alpha)\geq 0\\ p(Cq-\alpha)\geq 0\\ (q-1)(Dp-\beta)\geq 0,\\ q(Dp-\beta)\geq 0\\ 0\leq p\leq 1\\ 0\leq q\leq 1\end{cases}\qquad (9)$$

Где $\begin{cases}C=a_{11}-a_{12}-a_{21}+a_{22}\\ \alpha=a_{22}-a_{12}\\ D=b_{11}-b_{12}-b_{21}+b_{22}\\ \beta=b_{22}-b_{21}\end{cases}$

Вывод: если C и D не равны нулю, то легко видеть, что точки равновесия определяются формулами:

$$p=\frac{\beta}{D},\quad q=\frac{\alpha}{C}\qquad p=\frac{b_{22}-b_{21}}{b_{11}-b_{12}-b_{21}+b_{22}},\quad q=\frac{a_{22}-a_{12}}{a_{11}-a_{12}-a_{21}+a_{22}}$$

Таким образом, равновесие каждого игрока определяется элементами матрицы его противника. Заметим, что такого требования мы не вводили, но в итоге получилось, что равновесие игроков определяется не желанием максимизировать свой выигрыш, а в первую очередь желанием удержать под контролем выигрыш противника.

Как видим, стратегии, соответствующие точкам равновесия по Нэшу, определяются по «чужим» матрицам и решают вопрос максимального ограничения выигрыша противника. Однако у каждого игрока есть и «свои» матрицы. Что если найти стратегии игроков по ним? Оказывается, это будут стратегии, дающие максимально возможные выигрыши игрокам. Почему же мы их не нашли, когда искали точки равновесия? На самом деле эти стратегии не обладают свойством устойчивости. Т.е. если игрок использует свою «выигрышную» стратегию, то противник может отклониться от своей и получить дополнительный выигрыш. А игрок получит меньший выигрыш.

Тем не менее хочется каким то образом максимизировать свои выигрыши. При этом мы помним, что игра некооперативная, т.е. договариваться игрокам нельзя. Что же делать. Предлагается следующий подход: каждый из игроков имеет не одну, а две стратегии. Первая пара стратегий – стратегии угроз $\left(S_A^*, S_B^*\right)$. Они позволяют игрокам максимально ограничивать выигрыш соперника величинами $\left(H_A^*, H_B^*\right)$. Применяются, как это ясно из названия, для наказания противника за то, что он не хочет соблюдать обоюдную выгоду. Вторая пара – максиминные стратегии $\left(S_A^0, S_B^0\right)$ – как раз таки и позволяет игрокам получать тот самый максимальный выигрыш $\left(H_A^0, H_B^0\right)$.

Стратегии угроз соответствуют точкам равновесия по Нэшу и находятся по «чужим» матрицам. Максиминные – находятся по собственным матрицам игроков. Рассмотрим способы нахождения этих стратегий в общем виде. Пусть имеем игру размерности **m**x**n**. Как и в случае с матричными играми, предварительно преобразуем матрицы таким образом, чтобы они не содержали ни одного отрицательного элемента.

Перед нами четыре задачи. Будем решать их последовательно.

1. Поиск максиминной стратегии игрока **A**. Требуется найти стратегию $S_A^0 = \left(p_1, p_2, ..., p_m\right)$ таким образом, чтобы $V = H_A^0 \to \max$. Пусть игрок ***B*** применяет чистую стратегию стратегию $\boldsymbol{B_j}$. В этом случае выигрыш **A** должен быть больше или равен $V = H_A^0$.

Получаем систему неравенств

$a_{1j} p_1 + a_{2j} p_2 + ... + a_{mj} p_m \geq V \quad j = 1, ..., n$

Введём новые переменные $x_j = p_j / V \quad j = 1, ..., n$. Разделив все неравенства системы на неотрицательную величину **V**, получим

$$a_{1j} x_1 + a_{2j} x_2 + ... + a_{mj} x_m \geq 1 \quad j = 1, ..., n \tag{1}$$

Поскольку все элементы матрицы **A** неотрицательны, **V** тоже неотрицательна. Вероятности неотрицательны по определению. Таким образом введённые переменные должны удовлетворять условию

$$x_j \geq 0 \quad j = 1, ..., n \tag{2}$$

Из $p_1 + p_2 + ... + p_m = 1$ следует, что $x_1 + x_2 + ... + x_m = 1/V$. $V = H_A^0 \to \max$. Таким образом получаем целевую функцию

$$L = x_1 + x_2 + ... + x_m = 1/V \to \min \tag{3}$$

Мы получили задачу линейного программирования (1), (2), (3). Решив её, получим $V = 1/L$. Найдём стратегию $S_A^0 = \left(p_1, p_2, ..., p_m\right)$, $p_j = x_j \times V \quad j = 1, ..., n$. $H_A^0 = V$.

2. Поиск стратегии угроз игрока **A**. Требуется найти стратегию $S_A^* = \left(p_1, p_2, ..., p_m\right)$ таким образом, чтобы $W = H_B^0 \to \min$. Пусть игрок ***B*** применяет чистую стратегию стратегию $\boldsymbol{B_j}$. В этом случае проигрыш ***B*** не должен быть меньше, чем $W = H_B^*$.

Получаем систему неравенств

$b_{1j} p_1 + b_{2j} p_2 + ... + b_{mj} p_m \leq W \quad j = 1, ..., n$

Введём новые переменные $x_j = p_j / W \quad j = 1, ..., n$. Разделив все неравенства системы на неотрицательную величину **W**, получим

$$b_{1j}x_1 + b_{2j}x_2 + ... + b_{mj}x_m \le 1 \quad j = 1,...,n \tag{4}$$

Поскольку все элементы матрицы **B** неотрицательны, **W** тоже неотрицательна. Вероятности неотрицательны по определению. Таким образом введённые переменные должны удовлетворять условию

$$x_j \ge 0 \quad j = 1,...,n \tag{5}$$

Из $p_1 + p_2 + ... + p_m = 1$ следует, что $x_1 + x_2 + ... + x_m = 1/W$. $W = H_B^0 \to \min$. Таким образом получаем целевую функцию

$$L = x_1 + x_2 + ... + x_m = 1/W \to \max \tag{6}$$

Мы получили задачу линейного программирования (4), (5), (6). Решив её, получим $W = 1/L$. Найдём стратегию $S_A^* = (p_1, p_2,..., p_m)$, $p_j = x_j \times W \quad j = 1,...,n$. $H_B^* = W$.

Как видим, полученные задачи линейного программирования совершенно независимы. Теперь найдём стратегии второго игрока.

3. Поиск максиминной стратегии игрока **B**. Требуется найти стратегию $S_B^0 = (q_1, q_2,..., q_n)$ таким образом, чтобы $W = H_B^0 \to \max$. Пусть игрок ***A*** применяет чистую стратегию стратегию ***A***$_i$. В этом случае выигрыш **B** должен быть больше или равен $W = H_B^0$.

Получаем систему неравенств

$b_{i1}q_1 + b_{i2}q_2 + ... + b_{in}q_n \ge W \quad i = 1,...,m$

Введём новые переменные $y_i = q_i / W \quad i = 1,...,n$. Разделив все неравенства системы на неотрицательную величину **W**, получим

$$b_{i1}y_1 + b_{i2}y_2 + ... + b_{in}y_n \ge 1 \quad i = 1,...,m \tag{7}$$

Поскольку все элементы матрицы **B** неотрицательны, **W** тоже неотрицательна. Вероятности неотрицательны по определению. Таким образом введённые переменные должны удовлетворять условию

$$y_i \ge 0 \quad i = 1,...,m \tag{8}$$

Из $q_1 + q_2 + ... + q_n = 1$ следует, что $y_1 + y_2 + ... + y_m = 1/W$. $W = H_B^0 \to \max$. Таким образом получаем целевую функцию

$$F = y_1 + y_2 + ... + y_n = 1/W \to \min \tag{9}$$

Мы получили задачу линейного программирования (7), (8), (9). Решив её, получим $W = 1/F$. Найдём стратегию $S_B^0 = (q_1, q_2,..., q_n)$, $q_j = y_j \times W \quad j = 1,...,n$. $H_B^0 = W$.

Последняя задача (7), (8), (9) симметрично-двойственна задаче (4), (5), (6). Составим задачу поиска стратегии угроз второго игрока. Она будет симметрично-двойственна задаче поиска максиминной стратегии первого игрока (1), (2), (3).

Индивидуальные задания по теме № 15.

«Построение моделей биматричных игр, поиск точек равновесия по Нэшу, максиминных стратегий и стратегий угроз»

Дана биматричная игра размерности 2х2. Требуется найти для неё точки равновесия по Нэшу, максиминные и минимаксные стратегии и соответствующие им выигрыши игроков. Сравнить полученные результаты.

Практическое занятие № 16.

«Поиск оптимума по Парето, понятие кооперативных игр»

Критерий Парето принципиально отличается от критерия Нэша. Если там каждый игрок, действуя сам по себе, добивался максимально выгодного для себя исхода, то здесь игроки, действуя совместно, стараются найти решение, максимально устраивающее их обоих.

В данном разделе мы по прежнему рассматриваем ситуацию, когда переговоры невозможны, т.е. если игрок применяет смешанную стратегию, то это действительно каждый раз ход неожиданный для противника.

Отличие ситуации оптимальности по Парето вот в чём: если в ситуации равновесия по Нэшу ни один из игроков действуя в одиночку не мог увеличить свой собственный выигрыш, то в ситуации оптимальной по Парето игроки действуя совместно не могут увеличить выигрыш хотя бы одного из них без того, чтобы выигрыш другого не уменьшился. Применительно к биматричной игре 2x2 ситуацию (p^*, q^*) назовём оптимальной по Парето, если из того, что для некоторой точки (p, q). одновременно выполняются условия $H_A(p^*, q^*) \le H_A(p, q)$ и $H_B(p^*, q^*) \le H_B(p, q)$ следует, что $(p, q) = (p^*, q^*)$.

Для нахождения решения используется координатная плоскость V0W, где $V = H_A$, $W = H_B$. То есть по координатным осям откладываются выигрыши первого и второго игрока соответственно.

Очевидно, что любая пара стратегий игроков (S_A, S_B) однозначно проектирутся в некоторую точку плоскости с координатами (V,W). Соответствие не является взаимно однозначным, то есть произвольной точке (V,W) этой плоскости может соответствовать одна пара стратегий (S_A, S_B), несколько пар – или вообще не соответствовать ни одной из допустимых пар стратегий.

Теперь спроектируем на плоскость всё множество допустимых стратегий. Если рассматривать игры размерности 2x2, то проектируется ранее рассмотренный единичный квадрат. Для игр большей размерности это более сложное множество. В итоге получаем на плоскости некоторую связанную геометрическую фигуру. Характерно, что для некооперативных игр эта фигура вовсе не обязательно является выпуклой.

На этой фигуре выделяем поверхность Парето так, как это описано в первой главе. Если она содержит более чем одну точку, то окончательное решение принимаем, например, методом идеальной точки.

В итоге получаем точку с координатами (V*,W*). На последнем этапе нам требуется найти пару стратегий (S_A^*, S_B^*), проекцией которых и является эта точка. Решение может быть единственным или не единственным.

Причина, по которой в некооперативных играх получаются невыпуклые проекции области допустимых стратегий, заключаются в том, что при смешанных стратегиях игроки случайным образом выбирают каждый свою стратегию. Поэтому реализуются все возможные пары активных стратегий, в т.ч. и те, которые не выгодны ни одному из игроков.

В случае кооперативных игр такого не происходит. Игроки по взаимной договорённости в назначенное время реализуют согласованные пары стратегий. Не выгодные им пары никогда не реализуются.

набор частот

В кооперативных играх вместо отдельных стратегий игроков находится согласованный набор частот. Пусть игроку **A** доступно ***m*** стратегий $\boldsymbol{A_1, A_2, \ldots, A_m}$. Игроку **B** – ***n*** стратегий, $\boldsymbol{B_1, B_2, \ldots, B_n}$.

Набором частот называем матрицу $R = \{r_{ij}\}$ размером mxn, показывающую, как часто используется каждая пара стратегий ($\boldsymbol{A_i}$, $\boldsymbol{B_j}$). Очевидно, что сумма всех частот равна единице, и любая из этих частот неотрицательна.

$$\forall i = 1,\ldots,m,\ j = 1,\ldots,n \quad r_{ij} \geq 0 \quad \sum_{i=1}^{m}\sum_{j=1}^{n} r_{ij} = 1$$

По прежнему используем проекцию на плоскость V0W. Но теперь мы получаем на плоскости выпуклую линейную оболочку, натянутую как на вектора базиса на проекции всех возможных пар чистых стратегий, т.е. на точки с координатами (a_{ij}, b_{ij}) $i = 1,\ldots,m,\ j = 1,\ldots,n$.

Таким образом проекция всегда является выпуклым многоугольником. Некоторые из точек (a_{ij}, b_{ij}) могут оказаться внутренними точками этого многоугольника.

Следующий этап – выделить поверхность Парето. Очевидно, что она состоит из одного или нескольких отрезков, соединяющих вершины (a_{ij}, b_{ij}). Каждой такой вершине соответствует «вырожденный» набор частот, в котором частота данной пары стратегий $r_{ij} = 1$, а все остальные частоты равны 0.

Если отрезок соединяет две вершины, соответствующие ($\boldsymbol{A_i}$, $\boldsymbol{B_j}$) и ($\boldsymbol{A_k}$, $\boldsymbol{B_l}$), то им будет соответствовать матрица, в которой отличны от нуля только два элемента – r_{ij} и r_{kl}. Их сумма равна 1, следовательно весь отрезок можно описать как $r_{ij} = \alpha$ и $r_{kl} = 1 - \alpha$, $0 \leq \alpha \leq 1$.

Поверхность Парето состоит из одного или нескольких таких отрезков. Их все надо определить. Очевидно, что принятое решение должно быть элементом данного множества.

Индивидуальные задания по теме № 16.

«Поиск оптимума по Парето, понятие кооперативных игр»

Для ранее рассмотренной биматричной игры найти решение по Парето. Сравнить результаты.
Рассмотреть эту игру как кооперативную. Построить ОДР и поверхность Парето. Сравнить с результатом для некооперативной игры. Объяснить разницу.

Практическое занятие № 17.

«Нахождение решения арбитра и соответствующих ему оптимальных наборов частот**»**

На следующем этапе нам надо выяснить, все ли точки поверхности Парето могут быть кандидатами на окончательное решение. Оказывается, что не все и не всегда.

Вводится понятие точки Status quo (на рисунке обозначается как Sq). Это точка с координатами с координатами (V*,W*), обычно соответствующими точке равновесия по Нэшу. Т.е. те выигрыши, которые каждый из игроков может гарантировать себе при независимых действиях. Очевидно, что каждый из игроков пойдёт а совместные действия лишь в том случае, когда это принесёт ему выигрыш больший, чем при самостоятельных действиях. Это условия для первого игрока $V \geq V^*$, а для второго $W \geq W^*$. Выберем ту часть поверхности Парето, которая удовлетворяет данным условиям. Это может быть вся поверхность Парето или какая то её часть. В последнем случае вместо $0 \leq \alpha \leq 1$ будет изменение в каких то других, меньших пределах. И вырожденный случай, когда точка Status quo лежит на поверхности Парето – тогда переговорное множество вырождается до одной этой точки.

В принципе решение на этом окончено. От математики больше ничего не зависит. Игроки начинают «торговаться» и в итоге останавливаются на какой то из точек выделенного множества.

Решение арбитра

Рассмотрим ситуацию, когда игроки не могут остановиться на каком то одном решении, т.к. оба хотят «справедливости». Для этой цели приглашают третье лицо – «арбитра», который должен быть совершенно независимым. Арбитр принимает справедливое и беспристрастное решение, которое должно устраивать обоих игроков. Здесь мы попытаемся ввести математическое определение «справедливости» применительно к математической модели теории игр. К арбитру должны быть предъявлены следующие требования:

1. Арбитражное решение должно быть элементом переговорного множества.
2. Арбитражная схема должна быть независимой от имён или обозначений игроков.
3. Если две игры близки между собой в каком-то смысле, то и арбитражные решения должны быть близки.
4. Арбитражное решение должно отражать действенность угроз игроков.

В теории игр подобные требования пытаются формализовать в виде математических аксиом. Одну из таких систем аксиом предложил Дж. Нэш.

Пусть игроку **A** доступно $\boldsymbol{m}$ стратегий $\boldsymbol{A_1, A_2, \dots, A_m}$. Игроку **B** – $\boldsymbol{n}$ стратегий, $\boldsymbol{B_1, B_2, \dots, B_n}$.

Через S мы будем обозначать выпуклую оболочку точек (a_{ij}, b_{ij}), $\overline{S}$ - переговорное множество, (V^*, W^*)- точка Status quo, (V^0, W^0) – решение арбитра.

Аксиома 1. (*Оптимальность по Парето*). Точка (V^0, W^0) должна быть элементом переговорного множества, то есть

1. $(V^0, W^0 \in S)$
2. $V^0 \geq V^*, \quad W^0 \geq W^*$
3. в S нет точки (V, W), отличной от точки (V^0, W^0), такой, что $V \geq V^0, \quad W \geq W^0$

Аксиома 2. (*Симметрия*). Пусть игра обладает следую0щими свойствами:

1. $V^* = W^*$;
2. если точка $(V, W \in S)$, то и точка $(W, V \in S)$.

Тогда должно выполняться условие $V^0 = W^0$.

То если игроки находятся в совершенно одинаковой ситуации, то и арбитражное решение должно давать им одинаковые выигрыши.

Следующие две аксиомы менее очевидны, как предыдущие.

Аксиома 3. *(Инвариантность относительно линейного преобразования)*. Пусть имеются две игры одинаковой размерности и с платёжными матрицами, связанными соотношениями

$$\overline{a}_{ij} = \alpha_1 a_{ij} + \beta_1 \quad \overline{b}_{ij} = \alpha_2 b_{ij} + \beta_2 \quad i = 1, \ldots, m \quad j = 1, \ldots, n$$

Тогда арбитражные решения для них также должны быть связаны соотношениями

$$\overline{V}^0 = \alpha_1 V^0 + \beta_1 \quad \overline{W}^0 = \alpha_2 W^0 + \beta_2$$

Аксиома 4. (*Независимость несвязанных альтернатив*). Если к игре добавить новые ходы для игроков с добавлением новых элементов платёжных матриц таким образом, что точка status quo не меняется, то либо арбитражное решение также не меняется, либо оно совпадает с одной из добавленных сделок.

Дж. Нэш показал, что существует единственная арбитражная схема, удовлетворяющая этим четырём аксиомам. Арбитражное решение должно выносится из условия

$$\max_{(V, W) \in \overline{S}} (V - V^*)(W - W^*)$$

то есть решение арбитра (V^0, W^0)должно удовлетворять условию

$$(V^0 - V^*)(W^0 - W^*) \geq (V - V^*)(W - W^*)$$

для всех точек (V, W), принадлежащих переговорному множеству.

Индивидуальные задания по теме № 17.

«Нахождение решения арбитра и соответствующих ему оптимальных наборов частот»

Для рассмотренной в предыдущем разделе кооперативной игры выделить переговорное множество, найти решение арбитра и соответствующий ему набор оптимальных частот. Сравнить результаты с ранее полученными.