**Лекция 12-13**

**Линейное программирование**

3: Сущность ЛП состоит в нахождении точек наибольшего или наименьшего значения некоторой функции при определенном наборе ограничений, налагаемых на аргументы. Т.о. ЛП ­– это метод математического моделирования, разработанный для оптимизации использования ограниченных ресурсов. При этом целевые функции и ограничения строго линейны.

Следует с самого начала предупредить: предпосылка линейности, когда в реальности подавляющее большинство зависимостей носит более сложный нелинейный характер, есть огрубление, упрощение действительности. В некоторых случаях оно достаточно реалистично, в других же выводы, получаемые с помощью решения задачи ЛП, оказываются весьма несовершенными.

Математическая модель любой задачи ЛП включает в себя:

* переменные, которые следует определить;
* целевую функцию, подлежащую оптимизации;
* систему ограничений в форме линейных уравнений и неравенств.

4: **Пример**:

Фирма производит типографскую краску двух цветов: черную и синюю из сырья двух типов М1 и М2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Расход сырья (в тоннах) на тонну краски | | Максимально возможный ежедневный расход сырья |
| Черная | Синяя |
| Сырье М1 | 6 | 4 | 24 |
| Сырье М2 | 1 | 2 | 6 |
| Доход (в 1000$) на тонну краски | 5 | 4 |  |

Отдел маркетинга фирмы ограничил ежедневное производство синей краски до 2 т. (из-за отсутствия спроса), а также поставил условие, чтобы ежедневное производство синей краски не превышало более чем на тонну аналогичный показатель производства черной краски. Фирме требуется определить оптимальное (наилучшее) соотношение между видами выпускаемой продукции для максимизации общего ежедневного дохода.

5: ***Математическая модель***

*х*1 – ежедневный объем производства черной краски;

*х*2 – ежедневный объем производства синей краски.



Ежедневный расход сырья М1 и М2 ограничен соответственно 24 и 6 тоннами, поэтому получаем следующие ограничения:



Записываем ограничения по спросу: 1) *х*2 ≤ 2; 2) разность между ежедневными объемами производства красок черного и синего цветов не должна превышать 1 тонну, т.е. *х*2 – *х*1 ≤ 1.



Еще одно неявное ограничение состоит в том, что переменные *х*1и *х*2 должны быть неотрицательными. Т.о. имеем:



Любое решение, удовлетворяющее ограничениям модели, является допустимым. Например, решение *х*1 = 3 и *х*2 = 1 будет допустимым, так как не нарушает ни одного ограничения, включая условие неотрицательности. Значение целевой функции при этом решении будет равно z = 5\*3 + 4\*1 = 19 (тыс. $).

Итак, задача сформулирована, теперь встает вопрос о поиске оптимального допустимого решения, определяющего максимум целевой функции. Задача имеет много (фактически бесконечно много) допустимых решений. По этой причине невозможна подстановка значений переменных для поиска оптимума, т.е. нельзя применить простой перебор всех допустимых решений. Следовательно, необходима эффективная процедура отбора допустимых решений для поиска оптимального.

6:

1. **Графическое решение ЗАДАЧИ ЛП**

Использование графического метода удобно при решении задач ЛП с двумя переменными. При большем их числе необходимо применение алгебраического аппарата.

Графический метод решения задачи ЛП состоит из 2 этапов:

1. Построение пространства допустимых решений, удовлетворяющих всем ограничениям модели.
2. Нахождение оптимального решения среди всех точек пространства допустимых решений.

7:

 Пространство допустимых решений ограничено отрезками прямых, которые соединяются в угловых точках А, В, С, D, Е и F. Любая точка, расположенная внутри или на границе области, ограниченной ломаной ABCDEF, является допустимым решением, т.е. удовлетворяет всем ограничениям. Поскольку пространство допустимых решений содержит бесконечное число точек, необходима некая процедура поиска оптимального решения.

8: Для того чтобы найти оптимальное решение, необходимо определить направление возрастания целевой функции *z* = 5*х*1 + 4*х*2. Мы можем приравнять z к нескольким возрастающим значениям, например 10 и 15. Эти значения, подставленные вместо *z* в выражение целевой функции, порождают уравнения прямых; для значений 10 и 15 получаем уравнения прямых 5*х*1 + 4*х*2 = 10 и 5*х*1 + 4*х*2 =15. Целевая функция может возрастать до тех пор, пока прямые, соответствующие возрастающим значениям этой функции, пересекают область допустимых решений. Точка пересечения области допустимых решений и прямой, соответствующей максимально возможному значению целевой функции, и будет точкой оптимума.

На рис. видно, что оптимальное решение соответствует точке С. Эта точка является местом пересечения прямых (1) и (2), поэтому ее координаты *х*1 и *х*2 находятся как решение системы уравнений, задающих эти прямые:



Решением этой системы будет *х*1 = 3 и *х*2 = 1.5, при этом значение целевой функции равно *z* = 5*х*1 + 4*х*2 = 21. Полученное решение означает, что для фирмы оптимальным выбором будет ежедневное производство 3 т черной краски и 1.5 т – синей с ежедневным доходом в 21000 $.

9:

1. **СИМПЛЕКС-МЕТОД**

Переход от геометрического способа решения задачи ЛП к симплекс-методу лежит через алгебраическое описание крайних точек пространства решений. Для реализации этого перехода сначала надо привести задачу ЛП к стандартной форме, преобразовав неравенства ограничений в равенства путем введения дополнительных переменных.

Основное свойство симплекс-метода заключается в том, что решение задачи ЛП осуществляется итерационно. На каждой итерации алгоритм переходит к новой угловой точке, которая потенциально может улучшить значение целевой функции. Этот процесс перехода от одной угловой точки к следующей заканчивается, когда дальнейшее улучшение значений целевой функции невозможно.

Последовательность действий, выполняемых в симплекс-методе,

1. Преобразование в стандартную форму:

2. Находится начальное допустимое базисное решение.

3. На основе условия оптимальности определяется вводимая переменная. Если вводимых переменных нет, вычисления заканчиваются.

4. На основе условия допустимости выбирается исключаемая переменная.

1. Методом Гаусса-Жордана вычисляется новое базисное решение. Переход к п. 2.

10: **Преобразование задачи в стандартную форму**

1. Преобразовать неравенства в равенства;
2. Преобразовать свободные переменные в неотрицательные;
3. Целевая функция должна минимизироваться или максимизироваться.

**Пример**: 



 - свободная переменная (без ограничений).

Свободную переменную можно представить как разность двух неотрицательных переменных: 

11: Далее выполним следующие действия:

1. Вычтем из левой части первого неравенства дополнительную переменную *х*4 и затем умножим все неравенство на -1, для того, чтобы правая часть неравенства стала положительной.
2. Добавим дополнительную переменную *х*5 к левой части второго неравенства.
3. Т.к. третье ограничение изначально записано в виде равенства, то оставляем его без изменений.
4. Выполняем замену, где во всех ограничениях и целевой функции.



12:



13:

**Определение базисных решений**

Пусть ограничение задачи линейного программирования представлено в виде  равенств с  переменными и .

Положим значения  переменных равным нулю, а значения оставшихся  переменных найдем, как решение системы  уравнений.

Если полученные решение ( переменных) получится единственным, тогда эти  переменных называются ***базисными переменными***, а оставшиеся  ***небазисными переменными***. Значение базисных переменных называется ***базисным решением***.

Если значения базисных переменных не отрицательны, то это базисное решение называется ***допустимым решением***.

Количество базисных решений не превосходит .

14: **Пример**





Приведем к стандартной форме







15: ЗЛП в стандартной форме можно представить в виде след. таблицы.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  | Решение |
|  | 1 | -5 | -4 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 6 | 4 | 1 | 0 | 0 | 0 | 24 |
|  | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 6 |
|  | 0 | -1 | 1 | 0 | 0 | 1 | 0 | 1 |
|  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |

Допустимое решение , , 

Базисное решение , , , 

***Какая переменная дает наибольший рост функции  ?***

***Именно ее следует ввести в состав базисных переменных***

***При***  ***значение целевой функции возрастет на ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| базис |  |  | Точка пересечения | Комментарий |
|  | 6 | 24 | 24/6 = 4 > 0 | минимум |
|  | 1 | 6 | 6/1 = 6 > 0 |  |
|  | -1 | 1 | 1/(-1) = -1 | не подходит |
|  | 0 | 2 | 2/0 = ∞ | не подходит |

***Исключаемой (из базисных) переменных является .***

***Ведущий столбец, ведущая строка и ведущий элемент***.

16:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | 1 | -5 | -4 | 0 | 0 | 0 | 0 | 0 |
|  | 0 | 6 | 4 | 1 | 0 | 0 | 0 | 24 |
|  | 0 | 1 | 2 | 0 | 1 | 0 | 0 | 6 |
|  | 0 | -1 | 1 | 0 | 0 | 1 | 0 | 1 |
|  | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 |

**Вычисление нового базисного решения**

1. Вычисление элементов ведущей строки

- заменяем на ;

- все элементы ведущей строки (теперь это х1) делим на ведущий элемент (6)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | **0/6** | **6/6** | **4/6** | **1/6** | **0/6** | **0/6** | **0/6** | **24/6** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

17: 2. Вычисление элементов остальных строк (включая )

* вычисление -строки

новая -строка = текущая -строка - (пересечение -строки с ведущим столбцом) × (новая ведущая строка)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **-2/3** | **5/6** | **0** | **0** | **0** | **20** |
|  | **0/6** | **1** | **2/3** | **1/6** | **0** | **0** | **0** | **4** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

18:

* вычисление -строки

новая -строка = текущая -строка - (пересечение -строки с ведущим столбцом) × (новая ведущая строка)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **-2/3** | **5/6** | **0** | **0** | **0** | **20** |
|  | **0/6** | **1** | **2/3** | **1/6** | **0** | **0** | **0** | **4** |
|  | **0** | **0** | **4/3** | **-1/6** | **1** | **0** | **0** | **2** |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

19:

* вычисление -строки

новая -строка = текущая -строка - (пересечение -строки с ведущим столбцом) × (новая ведущая строка)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **-2/3** | **5/6** | **0** | **0** | **0** | **20** |
|  | **0/6** | **1** | **2/3** | **1/6** | **0** | **0** | **0** | **4** |
|  | **0** | **0** | **4/3** | **-1/6** | **1** | **0** | **0** | **2** |
|  | **0** | **0** | **5/3** | **1/6** | **0** | **1** | **0** | **5** |
|  |  |  |  |  |  |  |  |  |

20:

* вычисление -строки

новая -строка = текущая -строка - (пересечение -строки с ведущим столбцом) × (новая ведущая строка)



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **-2/3** | **5/6** | **0** | **0** | **0** | **20** |
|  | **0/6** | **1** | **2/3** | **1/6** | **0** | **0** | **0** | **4** |
|  | **0** | **0** | **4/3** | **-1/6** | **1** | **0** | **0** | **2** |
|  | **0** | **0** | **5/3** | **1/6** | **0** | **1** | **0** | **5** |
|  | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **2** |

Новое базисное решение , , , 

Новое уравнение



Если сделать базисной переменную , то мы можем увеличить 

21: Определим исключаемую переменную

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| базис |  |  | Точка пересечения | Комментарий |
|  | 2/3 | 4 | 4/(2/3) = 6 > 0 |  |
|  | 4/3 | 2 | 2/(4/3) = 3/2 > 0 | минимум |
|  | 5/3 | 5 | 5/(5/3) = 3 |  |
|  | 1 | 2 | 2/1 = 2 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **-2/3** | **5/6** | **0** | **0** | **0** | **20** |
|  | **0** | **1** | **2/3** | **1/6** | **0** | **0** | **0** | **4** |
|  | **0** | **0** | **4/3** | **-1/6** | **1** | **0** | **0** | **2** |
|  | **0** | **0** | **5/3** | **1/6** | **0** | **1** | **0** | **5** |
|  | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **2** |

22: Перерасчет таблицы

**s2-x2) **

**Z) x1)**

****

**s3)**

****

**S4)**

****

23:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| базис |  |  |  |  |  |  |  |  |
|  | **1** | **0** | **0** | **3/4** | **1/2** | **0** | **0** | **21** |
|  | **0** | **1** | **0** | **1/4** | **-1/2** | **0** | **0** | **3** |
|  | **0** | **0** | **1** | **-1/8** | **3/4** | **0** | **0** | **3/2** |
|  | **0** | **0** | **0** | **3/8** | **-5/4** | **0** | **1** | **5/2** |
|  | **0** | **0** | **0** | **1/8** | **-3/4** | **0** | **1** | **1/2** |

***Отрицательных коэффициентов в строке нет функция  достигла максимума***

***Оптимальное решение , ***



**Лекция 7**

**Задачи нелинейного программирования**

3: Нелинейное программирование – раздел математического программирования, изучающий методы решения экстремальных задач с нелинейной целевой функцией и (или) областью допустимых решений, определенной нелинейными ограничениями. К нелинейному программированию относят квадратичное, дробное, выпуклое, дискретное, целочисленное и геометрическое программирование.

В общем виде задачу нелинейного программирования можно сформулировать так:

*F*(*х*)→min (max) (1)

при условии

*g*(*x*)≤ 0, (2)

где *х* – вектор искомых переменных; *F*(*х*) - целевая числовая функция; *g*(*x*) – вектор-функция системы ограничений.

4: При этом могут быть разные случаи:

1. целевая функция – нелинейная, а ограничения – линейны;
2. целевая функция – линейная, а ограничения (хотя бы одно из них) – нелинейные;
3. целевая функция и ограничения нелинейные.

Задачи условной оптимизации нелинейного программирования бывают двух типов: когда в ограничениях (2) имеют место:

а) знаки равенства;

б) знаки неравенства.

Решение задачи нелинейного программирования (поиск глобального минимума или максимума) состоит в отыскании таких значений переменных, подчиненных системе ограничений, при которых достигает минимума или максимума данная целевая функция.

При решении некоторых нелинейных задач иногда удается использовать линейную теорию. Для этого вводят допущение, что на том или ином участке целевая функция возрастает или убывает пропорционально изменению переменных. Такой подход называется методом кусочно-линейных приближений.

5:

Среди большого числа вычислительных алгоритмов нелинейного программирования значительное место занимают:

* различные варианты градиентных методов (метод проекции градиента, метод условного градиента и т. п.);
* методы штрафных функций;
* методы барьерных функций;
* метод модифицированных функций Лагранжа и др.

6:

Универсального метода, позволяющего находить наиболее эффективным способом решение любой нелинейной задачи, не существует. Поэтому для каждой конкретной задачи, учитывая ее специфику, подбирают тот или иной наиболее подходящий метод (и алгоритм) решения.

Задачи нелинейного программирования на практике возникают довольно часто, например, когда затраты растут непропорционально количеству закупленных или произведенных товаров. Хорошо известно, что чем больше партия закупаемого товара, тем меньше стоимость единицы продукта. Каждому знакомо понятие розничных и оптовых цен.

Рассмотрим частный случай общей задачи нелинейного программирования (1), предполагая, что система ограничений (2) содержит только уравнения, отсутствуют условия неотрицательности переменных, *F*(*х*) и *g*(*x*) – функции, непрерывные вместе со своими частными производными. Ограничения в задаче заданы уравнениями, поэтому для ее решения можно воспользоваться классическим методом отыскания условного экстремума функций нескольких переменных. Вводят набор переменных, называемых множителями Лагранжа, и составляют функцию Лагранжа:



где *L*(*x*, λ) — лагранжиан; φ(*x*) — целевая функция; λ*i* (*i* = 1, 2, ..., *k*) – множители Лагранжа; *k* — число ограничений *gi*(*x*).

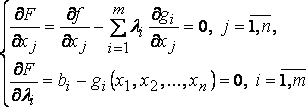
7:



находят частные производные



и рассматривают систему *n* + *m* уравнений:

 (3)

с *n* + *m* неизвестными  , .

Решив систему уравнений (3), получают все точки, в которых функция (1) может иметь экстремальные значения. Метод множителей Лагранжа имеет ограниченное применение, так как система (3), как правило, имеет несколько решений.

8: Пример. Найти точку условного экстремума функции  при ограничениях:

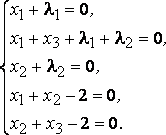


9: Составим функцию Лагранжа:

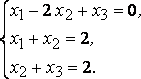


Продифференцируем ее по переменным . Приравнивая полученные выражения к нулю, получим следующую систему уравнений:

*x*2



10: Из первого и третьего уравнений следует, что ; тогда



Решив данную систему, получим:

 и 

**Лекция 15**

**Сетевые модели**

1. **Основные понятия**

1-2: Будем называть ***проектом*** деятельность, имеющую начало и конец во времени и направленную на достижение определенного результата. Результатом проекта может быть определенный продукт (построенное здание, разработанное программное обеспечение, собранное оборудование и т.д.) или услуга (транспортировка грузов, обучение персонала, медицинское обслуживание и т.п.). Часто успешное выполнение проекта называют ***реализацией проекта*** или просто говорят – ***процесс реализован***.

Как правило, проект представляется в виде ряда элементарных работ, которые называют ***операциями***. Некоторые операции могут быть выполнены только после завершения одной или нескольких других операций. В этом случае говорят о ***зависимости*** операций. Совокупность операций проекта и их зависимостей называется ***комплексом операций***.

Каждой операции комплекса соответствует два момента времени: начала и окончания операции. Эти моменты называются ***событиями***. Событие в отличие от операции не имеет протяженности по времени и является просто фиксацией факта начала или окончания операции. Если операции  предшествует операция  (или по другому – операция  ***опирается*** на ), а операция  имеет только одну последующую операцию, то событие, соответствующее окончанию операции *b*, будет являться одновременно событием, соответствующим началу операции .

Как правило, для обозначения событий используют целые положительные числа. При таком обозначении каждой операции соответствует пара , где  – номер ***начального***, а  ***– конечного события*** операции. При этом говорят, что ***операция инцидентна событиям***  и , и наоборот, ***события***  и  ***инциденты операции***. Часто для обозначения операции используют пару  номеров инцидентных событий.

Различают три вида событий комплекса операций: исходное, промежуточное и завершающее. ***Исходным событием*** называется событие, которое не является конечным ни для одной операции комплекса. ***Завершающим событием*** называется событие, которое не является начальным ни для одной операции комплекса. Все остальные события комплекса операций называются ***промежуточными***.

***Моментом свершения события*** считается момент окончания всех операций, для которых это событие является завершающим. Любая операция не может быть начата, если не свершилось событие, являющееся начальным для этой операции.

В общем случае комплекс операции может иметь несколько исходных и несколько завершающих событий (в этом случае говорят о ***многоцелевом комплексе***). В обоих случаях добавление одной общей предшествующей операции для всех исходных событий или одной последующей для всех завершающих сводит комплекс операций к виду с одним исходным и одним завершающим событием. В дальнейшем будут рассматриваться только такие комплексы.

В табл. представлен пример комплекса операций проекта разработки web-приложения с известной продолжительностью каждой операции. Для определенности назовем этот проект WSP.

3-5: **Комплекс операций проекта разработки web-приложения WSP**

|  |  |  |  |
| --- | --- | --- | --- |
| Код  операции | Наименование операции | Предшествующие операции | Продолжи-тельность  операции (дни) |
| I. АНАЛИЗ | | | |
| Z1 | Системный анализ |  | 15 |
| Z2 | Анализ требований | Z1 | 20 |
| II. ПРОЕКТИРОВАНИЕ | | | |
| Z3 | Проектирование базы данных | Z2, Z15, Z17 | 10 |
| Z4 | Проектирование классов | Z2, Z17 | 20 |
| Z5 | Проектирование интерфейсов пользователей | Z15, Z17 | 5 |
| III. КОДИРОВАНИЕ | | | |
| Z6 | Кодирование интерфейсов пользователей | Z4, Z5, Z16, Z17 | 15 |
| Z7 | Кодирование процедур СУБД | Z3, Z4, Z15, Z17 | 15 |
| Z8 | Кодирование классов | Z3, Z4, Z15, Z17 | 30 |
| IV. ТЕСТИРОВАНИЕ | | | |
| Z9 | Функциональное тестирование | Z6, Z7, Z8, Z18 | 30 |
| Z10 | Структурное тестирование | Z6, Z7, Z8, Z18 | 25 |
| V. ВНЕДРЕНИЕ | | | |
| Z11 | Разработка документации | Z6, Z7, Z8, Z9 | 10 |
| Z12 | Обучение пользователей | Z9, Z11 | 20 |
| Z13 | Испытание | Z9, Z10, Z11, Z12 | 60 |
| Z14 | Завершение работ | Z13 | 5 |
| VI. ДОПОЛНИТЕЛЬНЫЕ РАБОТЫ | | | |
| Z15 | Установка СУБД | Z1 | 3 |
| Z16 | Установка web-сервера | Z1 | 3 |
| Z17 | Установка инструментария | Z1 | 3 |
| Z18 | Подготовка полигона | Z1 | 4 |

Все работы проекта в приведенном примере разбиты на 6 групп. Каждой операции присвоен код, указаны продолжительности их выполнения и список предшествующих операций.

Проанализировав приведенный в табл. комплекс операций, не сложно выявить следующее:

1. у операции Z1 нет предшествующих операций;
2. у операции Z14 нет последующих операций;
3. все операции кроме Z5, Z12, Z13 и Z14 предшествуют нескольким операциям.

В соответствии с определением начальное событие операции Z1 является исходным, а завершающее событие операции Z14 – завершающим событием комплекса операций.

В табл. 2 вводится нумерация событий комплекса операций проекта WSP

6-7: **Нумерация событий комплекса операций проекта WSP**

|  |  |  |  |
| --- | --- | --- | --- |
| Начальное  событие | Код  операции | Предшествующие  операции | Конечное событие |
| 0 | Z1 |  | 1 |
| 2 | Z2 | Z1 | 3 |
| 4 | Z3 | Z2, Z15, Z17 | 5 |
| 6 | Z4 | Z2, Z17 | 7 |
| 8 | Z5 | Z15, Z17 | 9 |
| 10 | Z6 | Z4, Z5, Z16, Z17 | 11 |
| 12 | Z7 | Z3, Z4, Z15, Z17 | 13 |
| 14 | Z8 | Z3, Z4, Z15, Z17 | 15 |
| 16 | Z9 | Z6, Z7, Z8, Z18 | 17 |
| 18 | Z10 | Z6, Z7, Z8, Z18 | 19 |
| 20 | Z11 | Z6, Z7, Z8, Z9 | 21 |
| 22 | Z12 | Z9, Z11 | 23 |
| 24 | Z13 | Z9, Z10, Z11, Z12 | 25 |
| 26 | Z14 | Z13 | 27 |
| 28 | Z15 | Z1 | 29 |
| 30 | Z16 | Z1 | 31 |
| 32 | Z17 | Z1 | 33 |
| 34 | Z18 | Z1 | 35 |

При этом предполагается, что каждой операции соответствует два отдельных события: начальное и конечное. Например, для операции Z9 начальным событием является 16 и конечным 17.

8: **Построение сетевого графика**

В основе метода сетевого планирования и управления лежит понятие сетевого графика.

***Сетевой график*** представляет собой взвешенный ориентированный корневой графбез контуров (ациклический) и изолированных вершин, который построен по определенным правилам.

Пусть *Z* = {*z*1, *z*2, …, *zm*} – множество операций, а *I =* {*i*0*, i*1*,…in*} – множество событий комплекса операций проекта .

Построим ориентированный граф  по следующим правилам.

1. Количество вершин графа равно количеству событий комплекса операций ;

2. Количество дуг графа равно количеству операций комплекса операций ;

3. Должны быть заданы две биективные функции разметки, сохраняющие инцидентность событий и операций:

1). ,  – на множестве вершин графа ;

2). ,  – на множестве дуг графа .

9: 4. Граф должен иметь только одну вершину , не имеющую входящих дуг, она должна соответствовать исходящему событию  комплекса.

5. Граф должен иметь только одну вершину , не имеющую исходящих дуг, она должна соответствовать завершающему событию *it = f* (*vt* ).

6. Граф не должен содержать контуров.

7. Любая пара вершин графа должна быть соединена не более чем одной дугой.

10: Граф  представляет собой математическую модель комплекса операций. Каждая дуга *ek* этого графа соответствует одной операции *zk*, а каждая вершина *vi* – одному событию . Причем, если *ir* является начальным, а  – конечным событиями для операции , то  является начальной, а  – конечной вершинами для дуги , т.е. инцидентность вершин и дуг графа моделирует инцидентность событий и операций комплекса.

Математическая модель, заданная с помощью графа, построенного по описанным выше правилам, называется **сетевым графиком комплекса операций**.

Попытка построить сетевой график для комплекса операций, представленного в табл. 3.2 приводит к некоторым сложностям.

11: Рассмотрим сначала отдельно операцию Z4, имеющую начальное событие 6 и конечное 7. Она непосредственно предшествует операциям Z6, Z7 и Z8. Первоначальное построение приводит к графу, изображенному на рис. 1. Вершины графа (события) здесь изображены овалами, а дуги (операции) сплошными линиями со стрелками. Дуги соединяют вершины, соответствующие начальному и конечному событиям соответствующей операции.



Событие 7 должно предшествовать событиям 10, 12 и 14. Но в перечне (табл. 3.2) нет операций, связывающих эти события. В таких случаях необходимо расширить комплекс операций, добавив ***фиктивные*** операции, позволяющие отразить недостающие логические связи между событиями. На рис. 2 изображен предыдущий граф (рис. 1), в который добавлены три дуги, соответствующие фиктивным операциям F1, F2 и F3. Как правило, фиктивные операции изображаются на рисунках штриховой линией. Дальше всегда будем придерживаться такого обозначения.

12:



Фиктивные операции отражают технологическую или ресурсную зависимость в выполнении некоторых операций. В данном примере фиктивные операции F1, F2 и F3 отражают тот факт, что *кодирование интерфейсов пользователей* (операция Z6), *кодирование процедур СУБД* (Z7) и *кодирование классов* (Z8) в проекте WSP не могут быть начаты раньше, чем закончится *проектирование классов* (Z4). Для того, чтобы подчеркнуть, что операция является не фиктивной, будем использовать термин ***действительная операция***. Таким образом, все операции, приведенные в табл. 1 и 2, являются действительными операциями.

13: На рис. изображен сетевой график всего комплекса операций проекта WSP. Как и раньше, действительные операции отображены сплошными помеченными дугами, а фиктивные операции пунктирной линией. Обозначения для фиктивных операций вводить не обязательно – для этого будем использовать упорядоченные пары инцидентных вершин. Например, фиктивные операции <3,4 >, <29,4> и <33,4 > указывают на то, что действительная операция Z3 не может быть начата до тех пор, пока не закончатся действительные операции Z2, Z15 и Z17.

## Лекция 16.

## Векторная оптимизация.

3: *Постановка задачи векторной оптимизации.* Эффективность функционирования экономической системы оценивается, как правило, несколькими критериями. Математической формой критерия эффективности в оптимизационных экономико-математических задачах является целевая функция.

Пусть имеется  критериев, которые можно записать в виде целевых функций , где . Поскольку , то для простоты в дальнейшем будем предполагать, что все целевые функции максимизируются. Задача многокритериальной оптимизации в этом случае запишется

 (1)

; (2)

. (3)

4: Если точки максимума , определенные при решении задач по каждому критерию  не совпадают, то решение задачи (1)-(3) может быть только компромиссным. В области допустимых значений задачи находится область компромиссов. При перемещении из одной точки области компромиссов в другую, невозможно одновременное улучшение всех критериев. Принадлежащие области компромиссов планы называются эффективными или оптимальными по Парето (по имени итальянского экономиста, впервые сформулировавшего проблему многокритериальной оптимизации и принцип оптимальности).

План  оптимален по Парето, если он допустим и не существует другого плана  для которого



и хотя бы для одного критерия выполняется строгое неравенство.

5: К задачам векторной оптимизации приходят в следующих случаях:

1. Качество моделируемого процесса нужно оценить с точки зрения нескольких показателей. Это могут быть прибыль, себестоимость, рентабельность и т.д.
2. Моделируемый процесс представляет собой составляющую нескольких процессов (частей), и каждая из этих частей имеет свой критерий качества.
3. Моделируемый процесс расчленяется на несколько шагов и на каждом шаге его качество определяется своей функцией. (Например, на отдельных временных промежутках)

6: При разработке методов решения многокритериальных задач приходится решать ряд специфических проблем.

1. *Проблема нормализации* возникает наиболее часто. Отдельные критерии как правило имеют различные единицы и масштабы измерения, что делает невозможным их непосредственное сравнение. К единому и безразмерному виду критерии приводятся посредством операции нормирования. Наиболее распространенными способами нормирования является замена абсолютных значений критериев их относительными величинами

,

или относительными значениями отклонений от оптимальных значений критериев

.

7: 2. *Проблема учета приоритета критериев* встает, если критерии имеют различную значимость. В этом случае необходимо найти математическое определение приоритета и степень его влияния на решение задачи.

3. *Проблема определения области компромисса* возникает при решении многомерных нелинейных задач, поэтому для их решения необходимо применять методы, гарантирующие эффективное решение.

8: Методы решения задач многокритериальной оптимизации можно подразделить на четыре группы:

* + - * методы, основанные на свертывании критериев;
      * методы, использующие ограничения на критерии;
      * методы целевого программирования;
      * методы, основанные на отыскании компромиссного решения.

Вместо исходной многокритериальной задачи в соответствии с выбранным методом, формируется замещающая задача. В состав замещающей задачи входит один критерий, а к исходной системе ограничений добавляется одно или несколько дополнительных ограничений. Решение замещающей задачи называется субоптимальным.

9: *Метод последовательных уступок.*

Рассмотрим один из методов, использующих ограничения на критерии – метод *последовательных уступок*. Алгоритм метода следующий:

1. Критерии нумеруются в порядке убывания важности.

2. Решается задача



;

.

Определяется значение .

3. Устанавливается уступка , по этому критерию.

10: 4. Решается задача



;

.

Если в задаче более двух критериев, то пункты 3 и 4 повторяются для , ..., .

11: Пример. Решить задачу методом последовательных уступок, если уступка по первому критерию составляет 10% от его оптимального значения.



12: Решим задачу по критерию . Получим . В соответствии с условием задачи величина уступки . Дополнительное ограничение будет иметь вид , то есть . Решая задачу



получим , , 