《经典运筹学问题与模型》课程

第三章 背包问题及经典求解算法

唐加福 朱胳
东北财经大学管理科学与工程学院
jftang@mail.neu.edu.cn hanzhu@dufe.edu.cn
Tel:0411-84711310 / 84713592

第三章 背包问题及经典求解算法

典型运筹学问题与模型

基础知识

CH1: 绪论

CH2: 最优化理论

核心专题

CH3: 背包问题

CH4: 装箱问题

CH5: 指派问题

CH6: 旅行商问题

CH7: 路径规划问题

CH8: 集覆盖问题

CH9: 车间调度问题

CH10：最小生成树问题

CH:11: 最大割问题

前沿研究

CH12: 多臂老虎机问题

CH13: 报童问题

启发式方法

遗传算法

第三章 背包问题及经典求解算法

背包问题(Knapsack Problem)

- 基本的背包问题及特征
- 基本背包问题的数学模型
- 背包问题的变形（特征、模型描述）
- 基本背包问题的求解算法

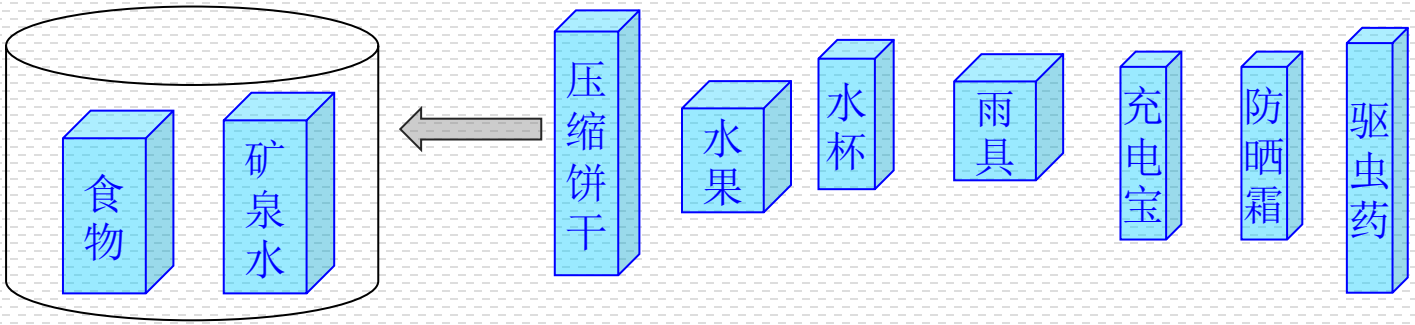
背包问题

- 所谓的背包问题，可以描述如下：
  - 小明要参加学校举办的“重走长征路”活动。
  - 有多种物品供他选择携带，每种物品有各自的重量和价值（如物品对小明的重要性）
  - 问题：小明要携带一个称重有限的背包，他应如何选择物品组合，从而使背包内物品总价值最大？

背包问题

- 可选物品（每个物品仅可选一次）

物品	压缩饼干	水果	水杯	登山杖	防晒霜	雨具	充电宝	驱虫药	矿泉水
重量	15	20	15	50	10	15	10	10	25
价值	40	5	2	60	35	20	45	60	40



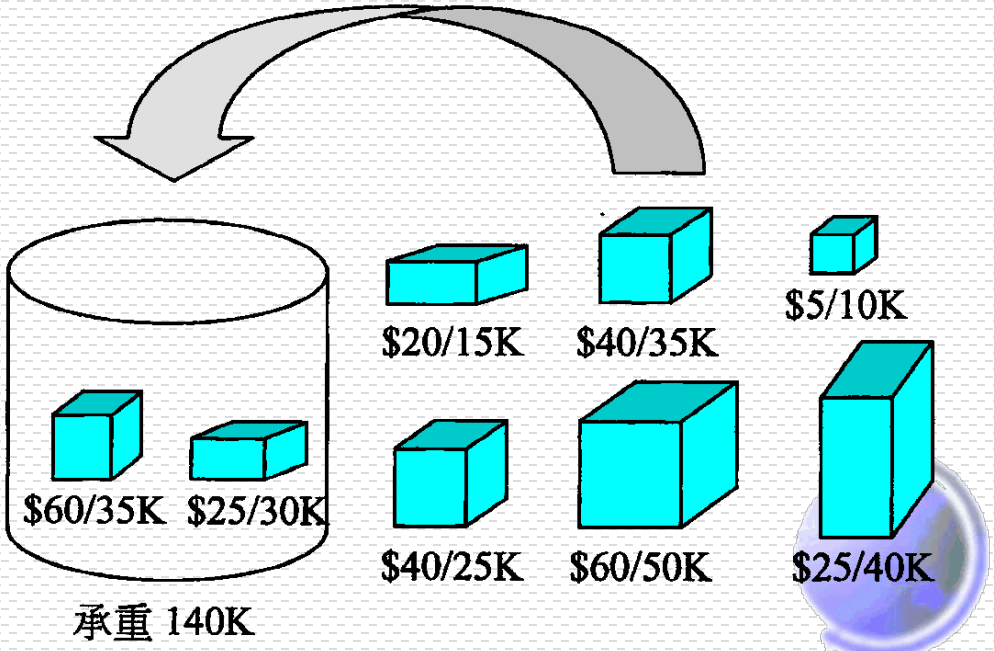
背包承重 W=130

背包问题

- 假设我们要从多种物品 (一般称为项目)中选择几件物品，装入背包。
- 若有 n 个不同的项目，对于项目 j 其重量为 w_j; 价值为 p_j，W 是背包承受的最大重量。
- 背包问题就是要在不超过背包承重量的前提下，使装入背包的项目的总价值最大。

背包问题的重量和价值

项目号	1	2	3	4	5	6	7	8
重量	35	50	30	15	10	35	25	40
价值	40	60	25	20	5	60	40	25



模型描述

- 背包问题的数学描述：

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

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j = 0 \text{ 或 } 1; \quad j = 1, 2, \cdots, n$$

$$x_j = \begin{cases} 1, & \text{项目 } j \text{ 被选入} \\ 0, & \text{其他} \end{cases}$$

0-1变量的引入

- 问题解的形式：背包里装了哪些物品
- 物品角度：每件物品是否放入背包
- 是否：引入0-1变量

- 背包问题吸引了许多理论和实际工作者，对此问题作了深入的研究。
- 理论研究的兴趣在于尽管问题的结构形式简单，但它却具有组合爆炸的性质，而且许多现实中的优化问题都可以通过解一系列背包子问题来解决。

特点

- 0-1背包问题(0-1 knapsack problem)
- 单约束的纯整数规划
  - 整数规划的一个重要类别
- NP完全问题

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

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \le W$$

$$x_j = 0 \text{ 或 } 1; \quad j = 1,2,\cdots,n$$

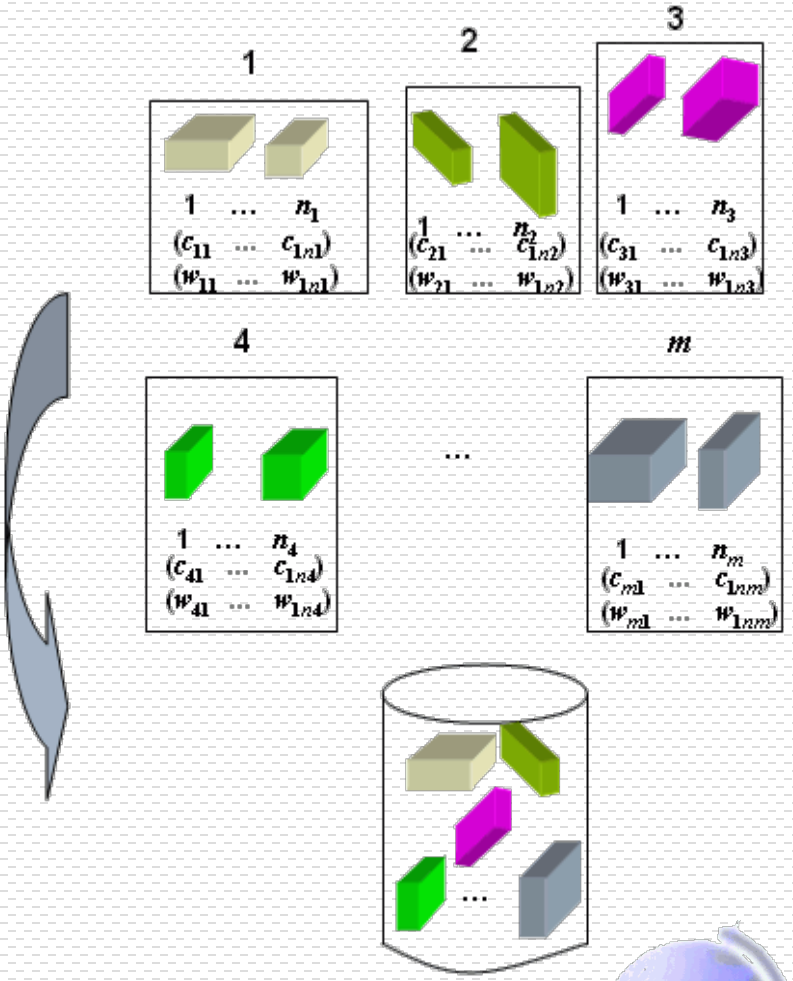
- 背包问题的变形
  - 多选择背包问题(multiple-choice knapsack problem)
  - 多约束背包问题(multi-constrained knapsack problem)
  - 有界（无界）背包问题(bounded knapsack problem)
  - 多背包问题（multiple knapsack problem)……

应用

- 资源分配问题
  - 设 w_j 是 n 项经营活动 x_j 各自所需的资源消耗，W是所能提供的资源总量，p_j 是人们从每项经营活动中得到的利润或收益，则背包问题就是*在资源有限的条件下，追求总的最大收益的资源有效分配问题*。
- 资金预算
  - 企业可投入的资金是有限的，不同的项目需要的投入资金是不一样的，投给不同的项目所能获得的净收益也是不一样的，问题是选择哪些项目进行投资，才能使投资的收益最大。
- 工厂的下料问题
  - 可以把原材料看做是背包，给不同的产品下料，所获利润是不一样的，问题是如何下料才能使生产的产品获利最大；
- 运输中的货仓装载问题
  - 例如集装箱装载问题，可以把集装箱看做是背包，待装载的货物具有不同的体积，以及不同的价值（或获利），问题是选择哪些物品装入箱中，能使价值（或获利）最大。

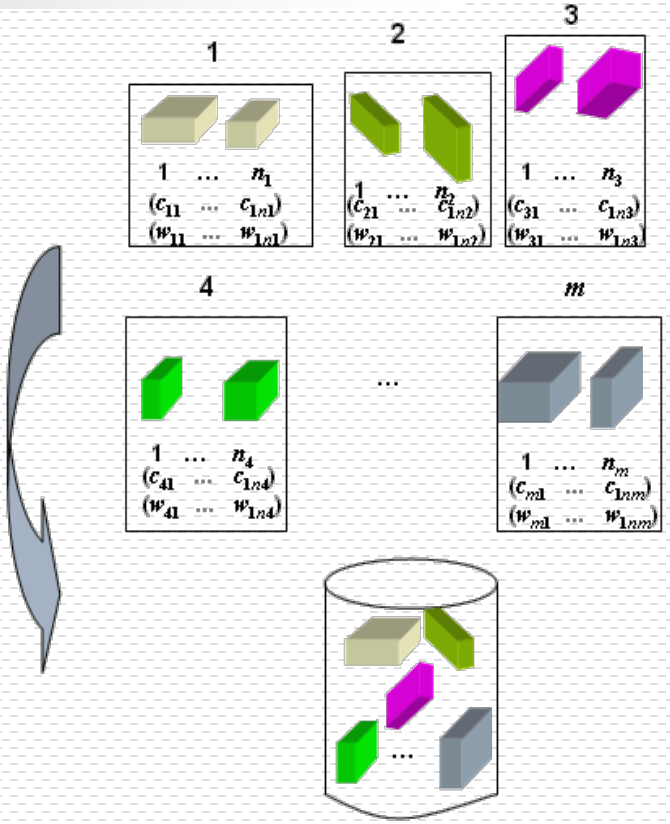
多选择背包问题

- 定义为有附加约束的背包问题
  - 该问题带有互不相关的多选择约束。
- 一般性描述：
  - 有一个承重有限的背包。将要放入背包的物品被分为相互排斥的若干类。每类中有若干不同的项目。
  - 问题就是从每类中选择一个项目使得项目总重量在满足背包承重约束的前提之下，最小化费用。
- 实例：
  - 产品结构的“质量一成本”优化问题



多选择背包问题--应用实例

- 产品结构的“质量一成本”优化问题
  - 一个产品由 m 个零部件组成（m类），
  - 每个零部件由多个备选件组成（一类中不同的物品），
  - 每个备选件都有一个质量权重（相当于价值）和成本（相当于重量），
  - 产品的设计成本不能超过目标成本（相当于背包的承重），
  - 问题是在不超过目标成本的前提下，选择什么样的零部件，才能使所装配的产品的质量最优（质量等级最高）



- 类似的问题：
  - 汽车组装问题
    - 一个汽车有很多零部件组成：像发动机、底盘、车身、轮胎、变速箱、电器设备、电子设备等等
  - 盖房子装修问题
    - 需要考虑：窗户、门、地板、瓷砖、墙面、热水器、淋浴器、抽油烟机等等。

多选择背包问题--模型描述

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}x_{ij}$$

$$\text{s. t.} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ij}x_{ij} \le W$$

$$\sum_{j=1}^{n_i} x_{ij} = 1, \qquad \forall i$$

$$x_{ij} \in \{0,1\}, \forall i, j \qquad x_{ij} = \begin{cases} 1 & \text{第}i\text{类的第}j\text{个项目被放入背包} \\ 0 & \text{其它} \end{cases}$$

这里采用双下标使得变量的 m 个多选择集相互排斥。约束称作多选择约束或广义上界

- 其中：
  - i：类的下标
  - j：每类中项目的下标
  - m：类的数量
  - n_i：第 i 类中项目的数
  - p_{ij}：第 i 类中第 j 个项目的质量权重,
  - w_{ij}：第 i 类中第 j 个项目的重量
  - W：背包的承重.

多选择背包问题--模型描述

$$\max \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{ij}x_{ij} \le W$$

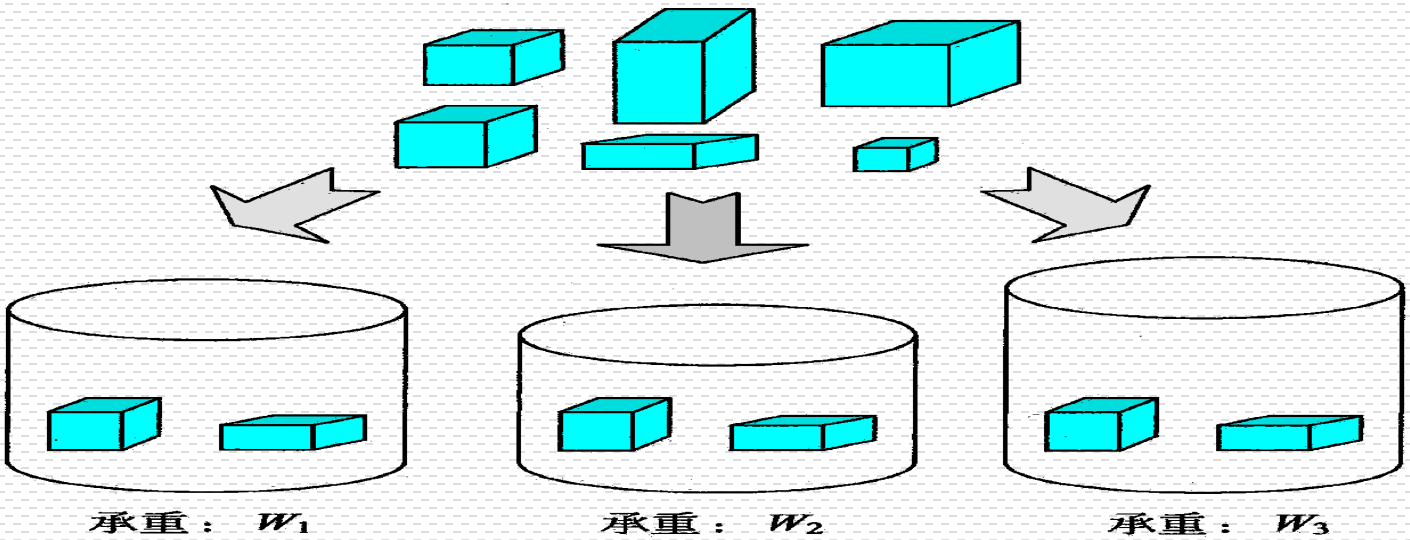
$$\sum_{j=1}^{n_i} x_{ij} = 1, \qquad \forall i$$

$$x_{ij} \in \{0,1\}, \forall i, j \qquad x_{ij} = \begin{cases} 1 & \text{第}i\text{类的第}j\text{个项目被放入背包} \\ 0 & \text{其它} \end{cases}$$

- 该问题是某种广义的指派问题，属于*NP*难题。
- 解决该问题最成功的方法是分枝定界算法，
- 这种算法采用了线性规划，*Lagrangian*松弛或*Lagrangian*松弛变形来定界。

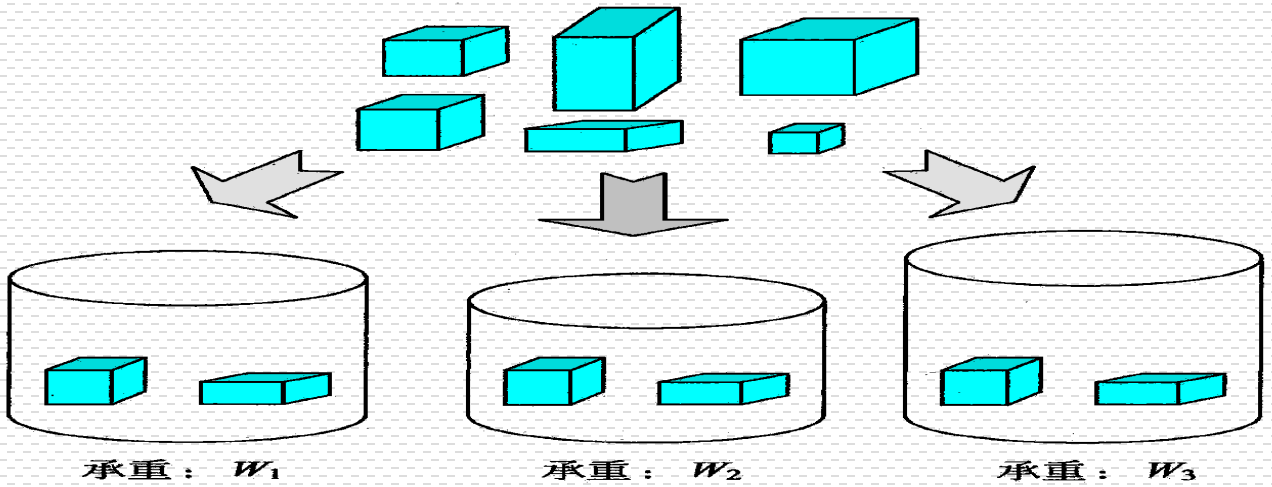
多约束背包问题

- 是带有一组约束 (比如重量、体积、可靠性等) 的背包问题
- 也称作多背包问题 、或多维背包问题
- 问题的描述：
  - 存在承重分别为 W_1，W_2，…，W_m 的 m 个背包和 n 个物品，每个物品的价值为 p_j，j=1,2,…,n。与简单背包问题中物品重量恒定所不同的是，如果第 j 个物品放入第 i 个容量为 W_i 的背包，则它在第 i 个约束中的权重为 w_{ij}。（假设不要求一个物品只能放在一个包里。）
  - 我们希望将尽量多的物品放入背包，在确保背包承重满足的前提之下最大化总价值。



多约束背包问题-应用举例

- 在资源配置中的应用
  - 把经营活动作为物品，不同的资源作为不同的背包。物品放入不同背包中的权重，相当于经营活动消耗的不同的资源。
  - 例如，在资源受限的情况下，生产哪些产品的组合能使利润最大的问题。产品相当于物品、资源相当于不同的背包、产品消耗的资源相当于放入不同背包时所表现出来的重量。
- 货物装载问题
  - 例如：一个集装箱，即受到重量的限制，同时也受到体积的限制。



多约束背包问题-模型描述

$$\max \quad f(x) = \sum_{j=1}^{n} \boldsymbol{p}_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} w_{ij} x_j \le W_i \quad i = 1,2,\cdots,m$$

$$x_j \in \{0,1\}, \forall j$$

$$x_j = \begin{cases} 1 \text{第} j \text{个物品被放入背包(所有的背包)} \\ 0 \text{其它} \end{cases}$$

- 其中：
  - i：背包的下标
  - j：项目的下标
  - m：背包的数目
  - n：项目的数目
  - c_j：第 j 个项目的费用,
  - w_{ij}：第 j 个项目放入第 i 个背包中的权重
  - W_i：第 i 个背包的承重.

基本背包问题的求解方法

- 分枝定界法
- 贪婪算法
- 领域搜索算法
- 遗传算法
- 动态规划

一般的求解方法-分枝定界法

- Brand and Bound Method
- 基本思想和原理

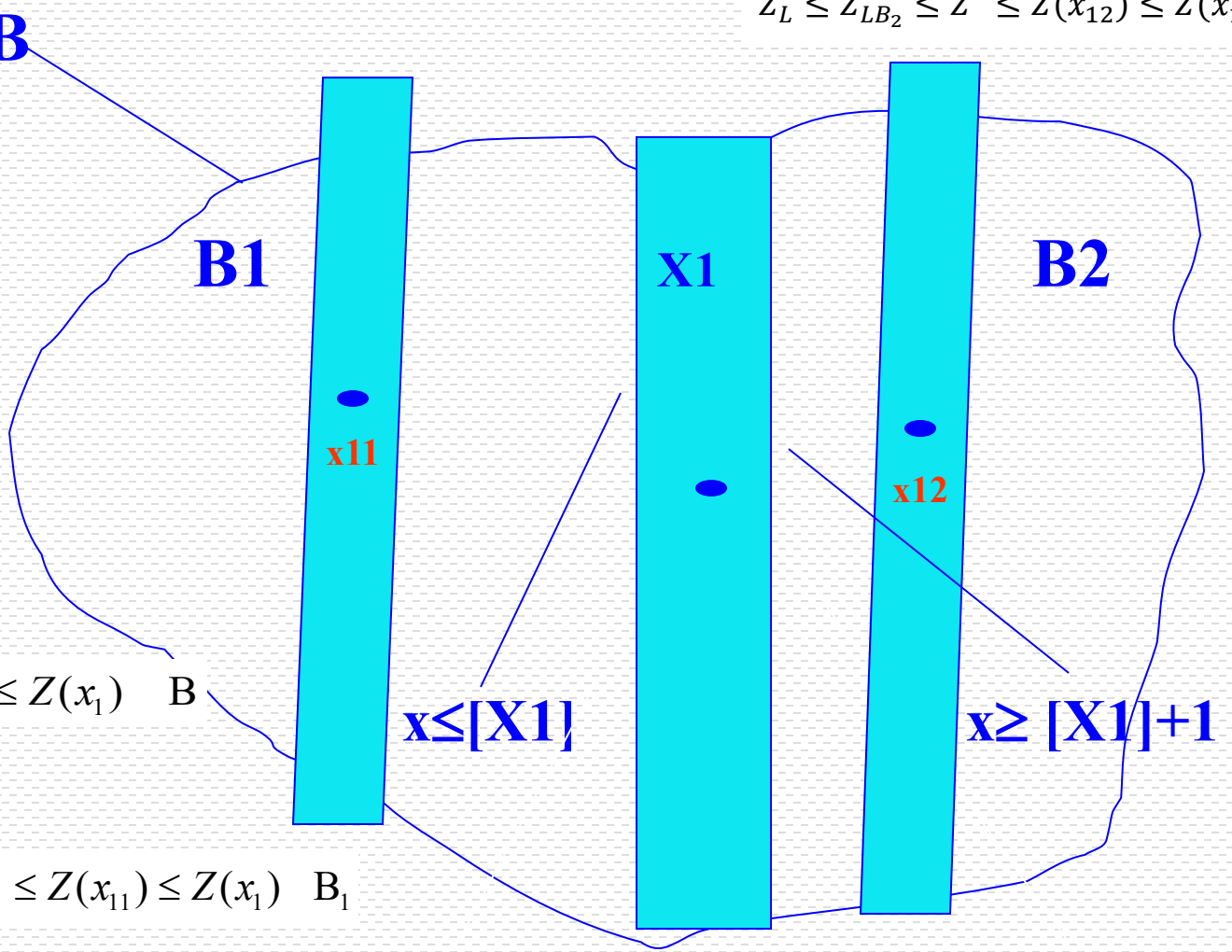
设A是最大化的整数规划问题，设其对应的线性规划问题为B；从求解B开始，设其最优解为X_B；若X_B不是原问题A的可行解，则B的最优目标Z^u是A的最优目标函数Z^*的上界；而A的任意可行解的目标函数Z_l必是A的最优目标Z^*的下界，即

$$Z_l \leq Z^* \leq Z^u$$

分枝定界方法将B的可行域分成两个子区域（分支），然后在子区域里逐步减少Z^u，增大Z_l；通过求解B最后获得A的解。

一般的求解方法-分枝定界法

- 图例解释原理



$Z_L \le Z_{LB_2} \le Z^* \le Z(x_{12}) \le Z(x_1)$ B_2

$Z_L \le Z^* \le Z(x_1)$ B

$Z_L \le Z_{LB_1} \le Z^* \le Z(x_{11}) \le Z(x_1)$ B_1

一般的求解方法-分枝定界法

- 分枝定界法的组成
  - 分枝：在B的最优解处分别增加两个整数约束，将可行域B分枝成两个子区域B1和 B2
  - 剪枝：剪枝B2，如何剪？什么规则
  - 定界：在什么结点定，如何定

$$Z_L \le Z_{LB_2} \le Z^* \le Z(x_{12}) \le Z(x_1) \quad \mathrm{B}_2$$

$$Z_L \le Z_{LB_1} \le Z^* \le Z(x_{11}) \le Z(x_1) \quad \mathrm{B}_1$$

一般的求解方法-分枝定界法

- 举例说明分枝定界方法

A: $\max\ z = x_1 + x_2$

$s.t.\quad 2x_1 + x_2 \le 6$

$4x_1 + 5x_2 \le 20$

$x_1, x_2 \ge 0, \text{integer}$

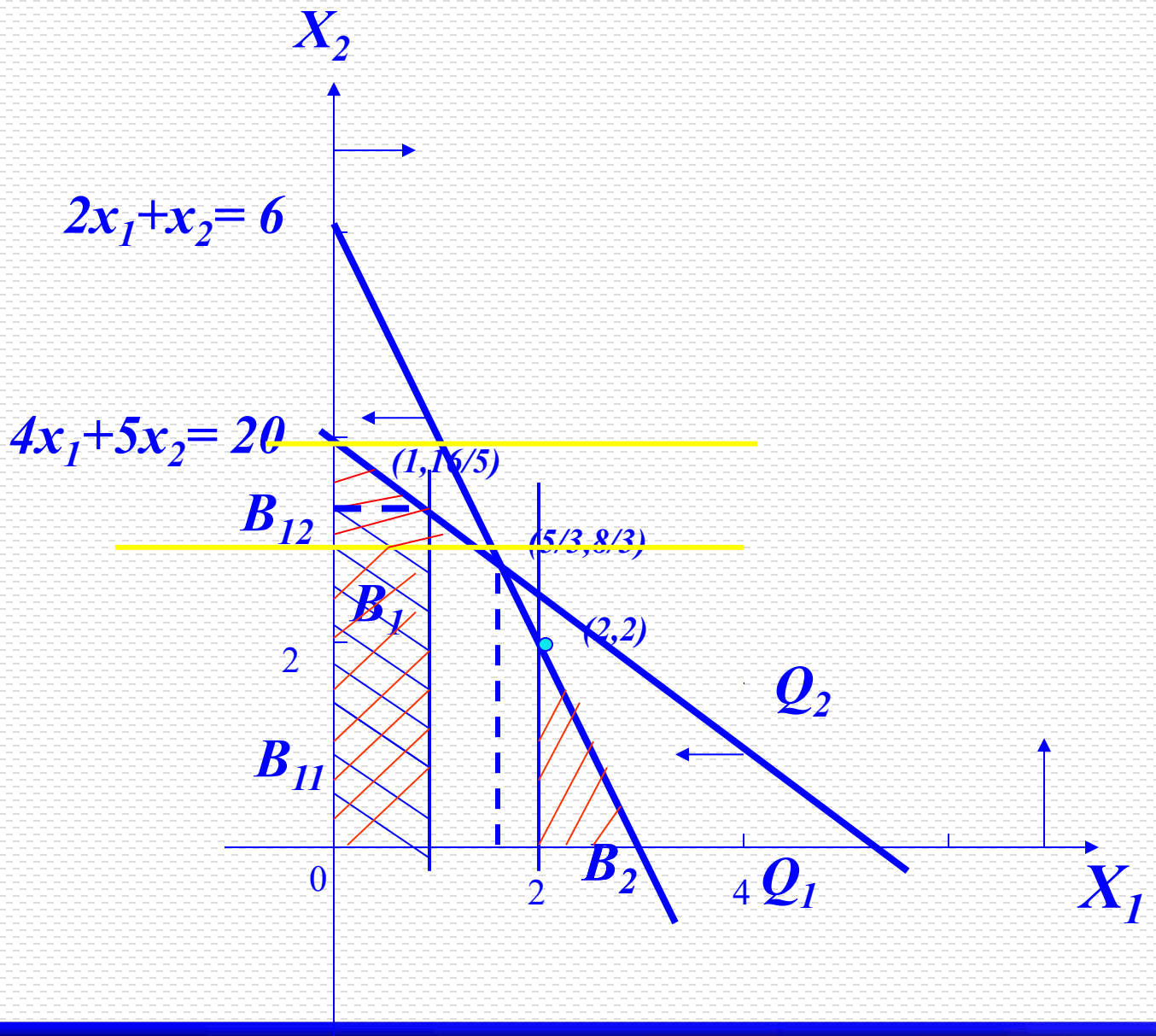
B: $\max\ z = x_1 + x_2$

$s.t.\quad 2x_1 + x_2 \le 6$

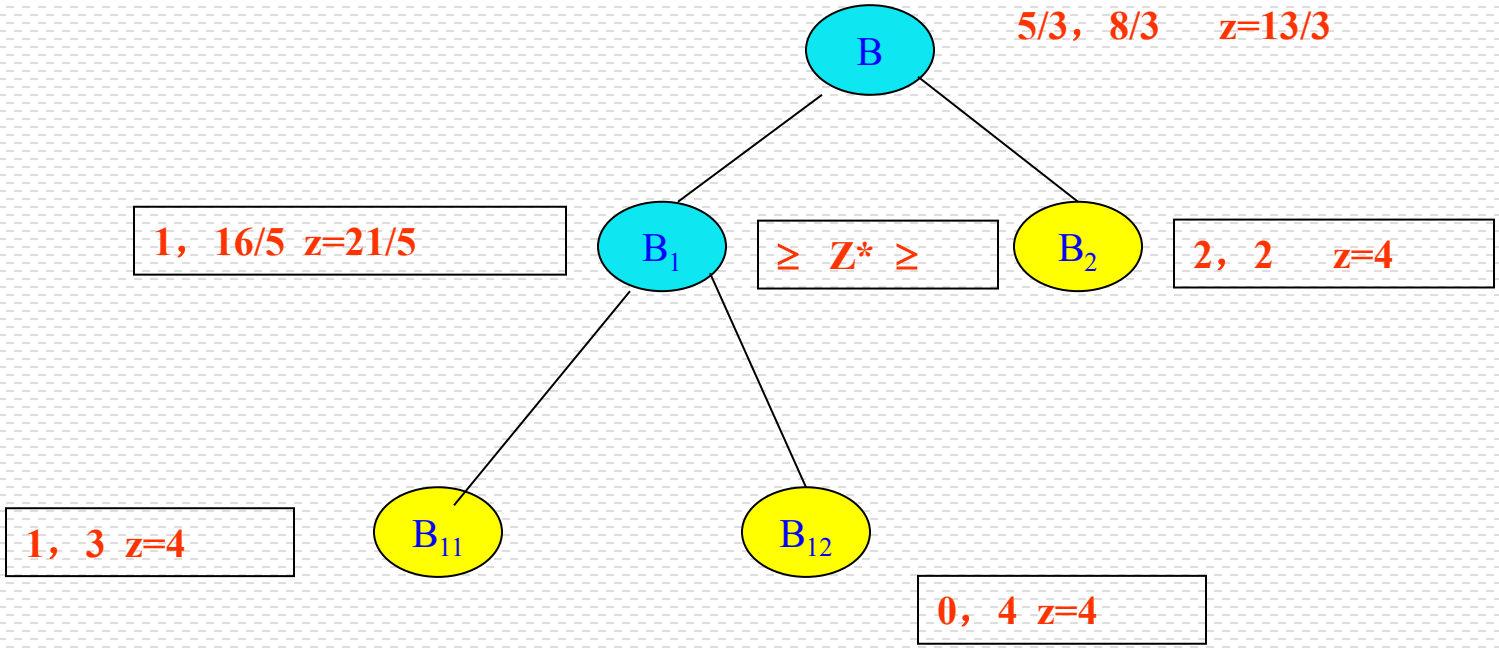
$4x_1 + 5x_2 \le 20$

$x_1, x_2 \ge 0$

- 求解
  - 求解问题B，得到 $x_1=5/3$， $x_2=8/3$，$Z^u=13/3$
  - 增加约束 x1≥2； x1≤1，分别得到可行域B1和B2
  - 分别求解问题B1和B2，得到相应的解及目标函数值


X2
2x1+x2= 6
4x1+5x2= 20
(1,16/5)
B12
(5/3,8/3)
B1
(2,2)
2
Q2
B11
0
2
B2
4
Q1
X1

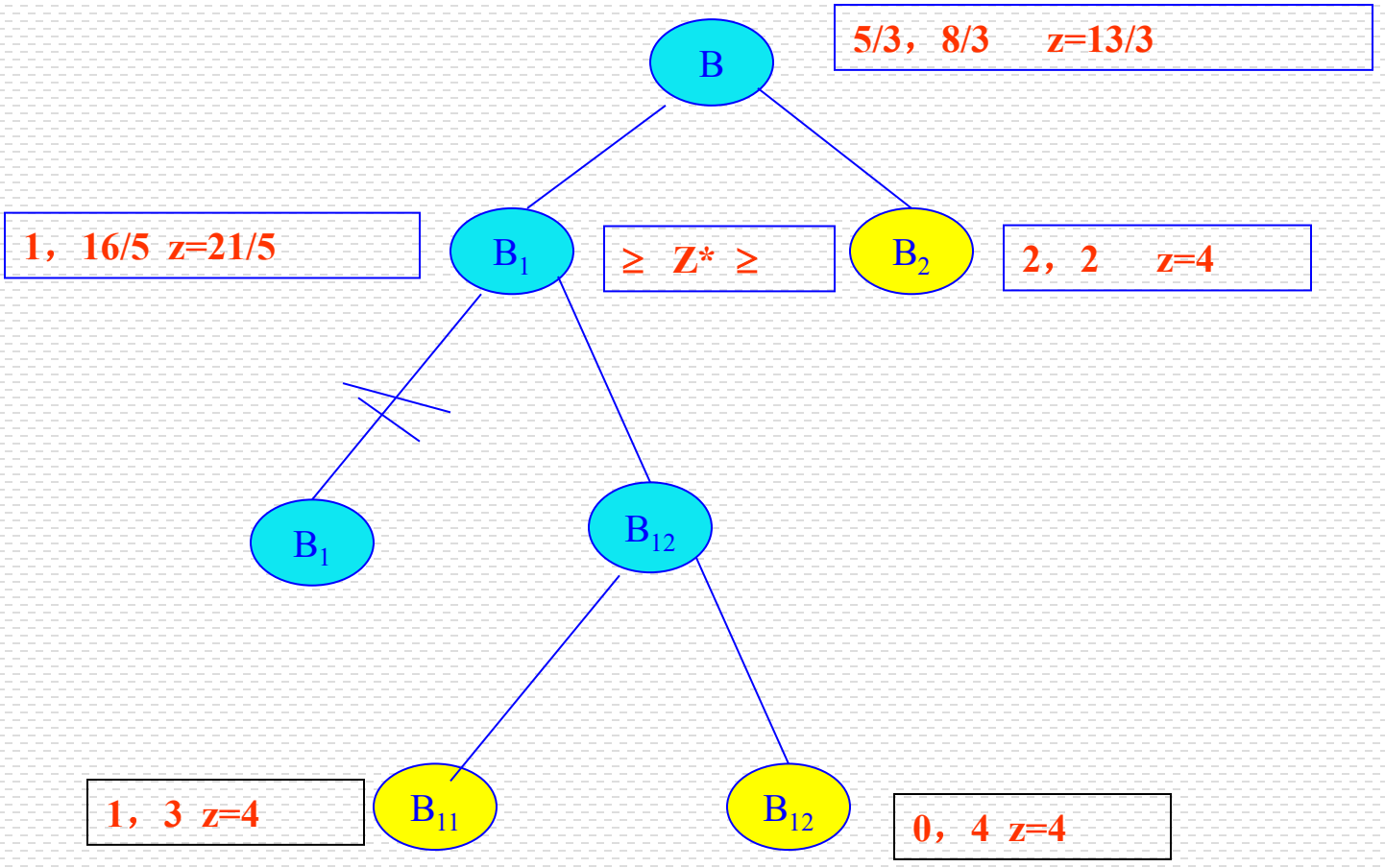
一般的求解方法-分枝定界法



1 如果B11得到的解不是整数，且其目标函数值小于B2的目标值，如何处理?

2 如果B11得到的解是整数，但其目标函数值小于B2的目标值，**目标函数的界**如何确定?

一般的求解方法-分枝定界法



一般的求解方法-分枝定界法

- 例

A: $\max\ z = 40x_1 + 90x_2$

$s.t.\quad 9x_1 + 7x_2 \le 56$

$7x_1 + 20x_2 \le 70$

$x_1, x_2 \ge 0, \text{integer}$

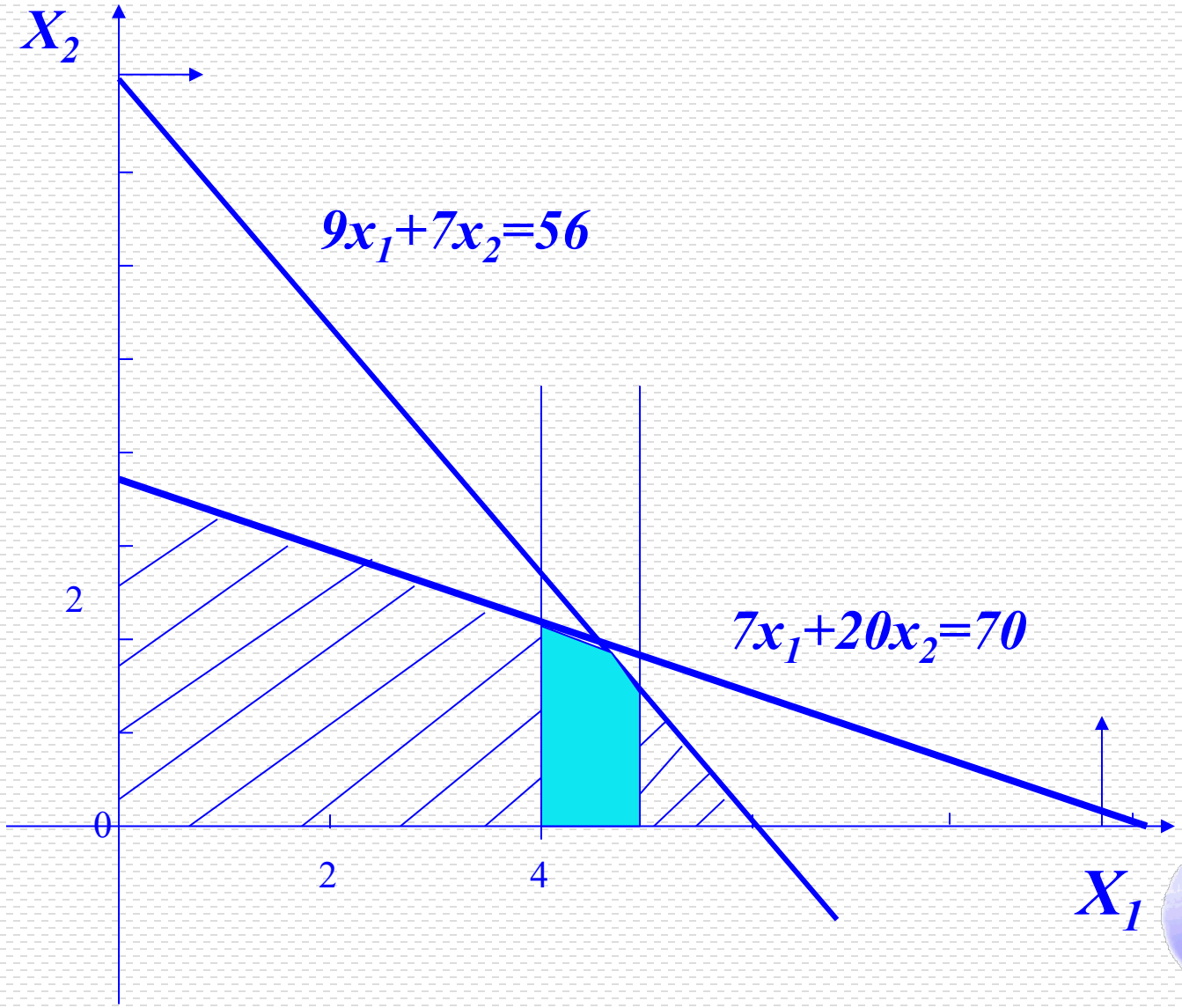
B: $\max\ z = 40x_1 + 90x_2$

$s.t.\quad 9x_1 + 7x_2 \le 56$

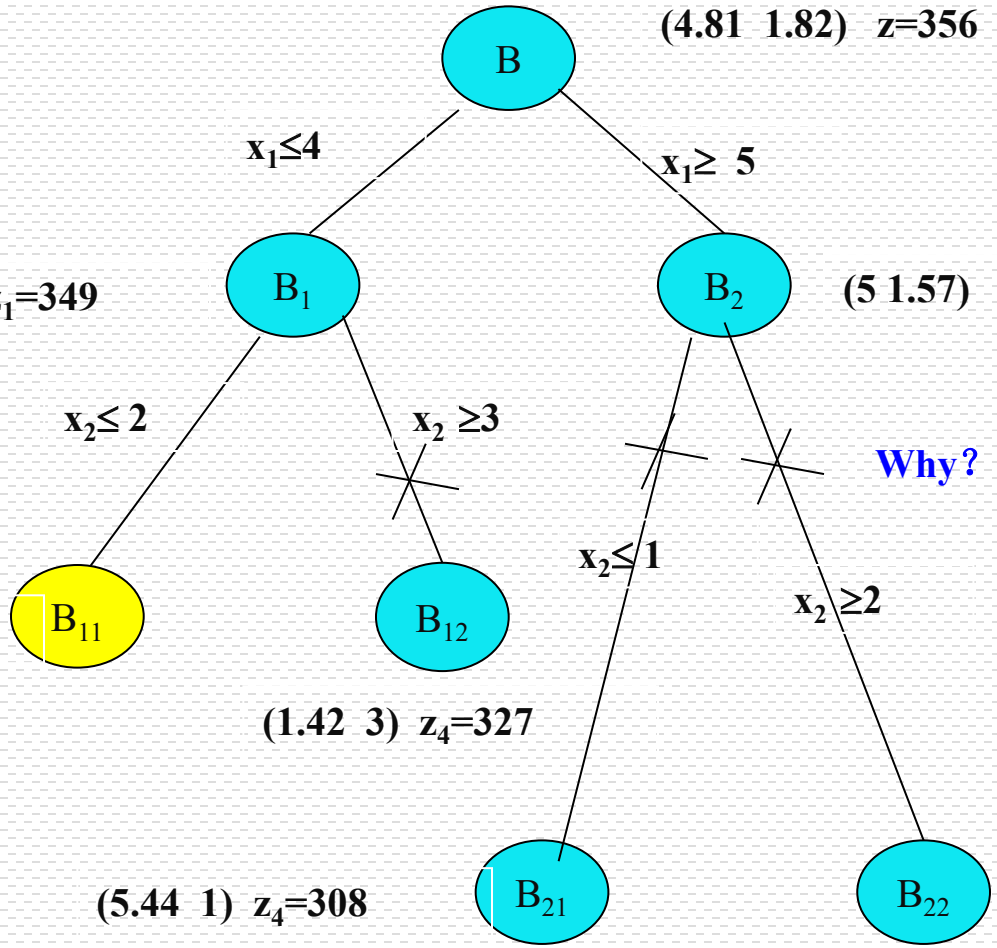
$7x_1 + 20x_2 \le 70$

$x_1, x_2 \ge 0$

一般的求解方法-分枝定界法



一般的求解方法-分枝定界法



$Z_l=0$ $Z_u=356$

$Z_l=0$ $Z_u=349$

Why?

$Z_l=340$ $Z_u=341$

Why?

Why?

基本背包问题的求解方法

- 分枝定界法
- 贪婪启发式算法
- 领域搜索算法
- 遗传算法
- 动态规划

一般的求解方法-贪婪启发式法

■ 背包问题的贪婪启发式

对背包问题可以构造下面的贪婪算法：

$step1$: 对物品以$\frac{p_i}{\omega_i}$从大到小排列，不妨把排列记成$\{1，2，...，n\}, k = 1$；

$step2$: 若$\sum_{i=1}^{k-1} \omega_i x_i + \omega_k \leq b$, 则$x_k = 1$, 否则，$x_k = 0$；$k = k + 1$

当$k = n$时，停止；否则，重复$step2$

一般的求解方法-贪婪启发式法

- 问题：贪婪启发式方法得到的一定是最优解吗？

假设有一个背包，最多能装20公斤物品，现有如下物品可选。

物品	电脑	收音机	钟	花瓶	书	油画
价格	220	20	175	50	15	90
重量	20	4	10	2	1	9

基本背包问题的求解方法

- 分枝定界法
- 贪婪算法
- 邻域搜索算法
- 遗传算法
- 动态规划

基本背包问题的求解方法-邻域搜索算法

例 2 简单的邻域搜索（Local search）算法

给定组合优化问题，假设其邻域结构已确定，设 S 为解集合，f 为 S 上的费用函数，N 为邻域算法，算法为：

Step1：任选一个初始解 $s_0 \in S$；

Step2：在 $N(s_0)$ 中按某一规则选一 s，若 $f(s) < f(s_0)$，则 $s_0 = s$；否则

$N(s_0) = N(s_0) - s$，若 $N(s_0) = \varnothing$，停止；否则返回 Step2

作业1：请设计求解背包问题的分枝定界算法、邻域搜索算法，编程实现并用一实例与贪婪算法比较

基本背包问题的求解方法

- 分枝定界法
- 贪婪算法
- 领域搜索算法
- **遗传算法**
- 动态规划

求解基本背包问题的GA

- 求解方法分类
  - 二进制表达法(Binary representation);
  - 顺序表达法(Order representation);
  - 变长表达法(Variable-length representation)。

求解基本背包问题的GA—二进制表达法

- 二进制串是 0-1 背包问题的自然表达，其中 1 表示选入，0 表示未选入。

$$x = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8 \ x_9 \ x_{10}]$$

染色体:	0	1	0	1	0	0	0	0	1	0

  - 这表示项目 2，4 和 9 被选入背包
- 问题：
  - 这种表达会产生不可行解（违反约束条件）。当染色体中的 1 太多时，可能超过背包的承重量。
- 解决方法：
  - 惩罚法：给每个不可行解指定一个惩罚值
  - 解码法：用贪婪近似法将不可行解转化为可行解

求解基本背包问题的GA—二进制表达法

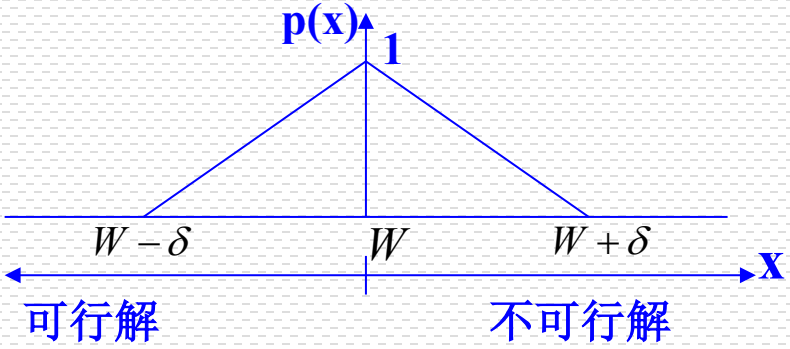
- 惩罚法
  - Gordon 和 Whitley 给每个不可行解一个简单的罚值。
  - 适值函数构造为：

    $eval(x) = f(x)\, p(x)$
  - 极大化问题的惩罚函数为：
  - 惩罚曲线：

约束违反量
或背包剩余量

$$p(x) = 1 - \frac{\left|\sum_{j=1}^{n} w_j x_j - W\right|}{\delta}$$

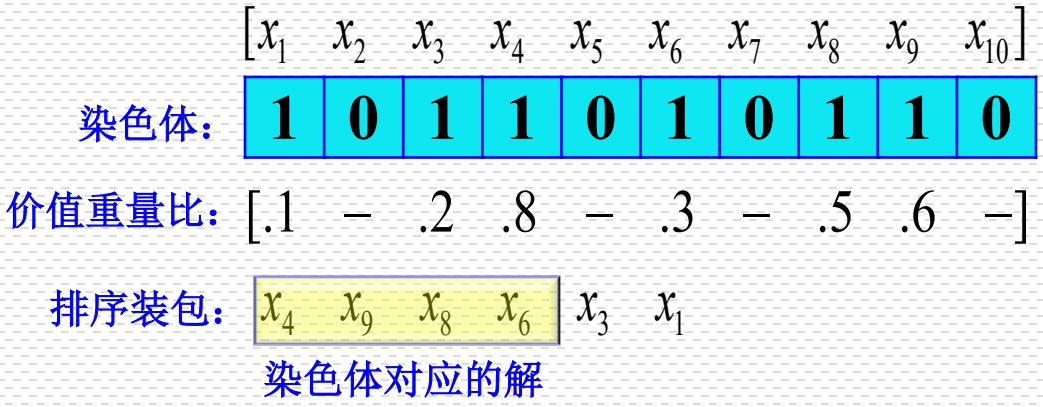
$$\delta = \max\left\{W, \left|\sum_{j=1}^{n} w_j - W\right|\right\}$$



  - 从图中可见，惩罚函数不仅惩罚超过背包承重量的不可行解，也惩罚背包承重量利用不足的可行解。
  - 对于这类最优解位于可行域与不可行域边界上的问题，这种惩罚函数正是所需要的。

求解基本背包问题的GA—二进制表达法

- 解码法
  - 是用贪婪近似法将不可行解转化为可行解，步骤如下：
    - 将 x_j=1 的项目按 价值 与 重量比(单位重量的价值)的降序排列
    - 按优先适合启发式选择项目，直到背包不能再装为止
    - 例如：



  - 问题：
    - 编码和解之间不存在简单的一一对应关系。多个表面上不同的染色体，可能对应一个解。
- Gordon 和 Whitley 试验了惩罚法和解码法
  - 一个20个项目的较难问题，惩罚法较好
  - 一个80个项目的较容易问题，解码法较好。

求解基本背包问题的GA—顺序表达法

- 对于一个 n 个项目的问题，染色体由 n 个基因构成，每个基因用一个不同的正整数代表一个项目。
  - 项目在顺序中的位置，可以看作是该项目选入背包的优先级。
  - 按项目的顺序用优先适合启发式方法，就可产生一个可行解。

例如，7个项目的染色体: | 1 | 6 | 4 | 7 | 3 | 2 | 5 | ➔ 可行解 [1 6 4 5]

$$x_1 = x_4 = x_5 = x_6 = 1$$
$$g(x) = 100 \le W$$
$$f(x) = 73$$

7项目背包问题

项目 j	1	2	3	4	5	6	7
重量 w_j	40	50	30	10	10	40	30
价值 c_j	40	60	10	10	3	20	60

$$W = 100$$

- 问题：
  - 这种染色体到可行解的映射是多对一的。这个特点使得该方法大大地减少了种群的多样性。
  - 因此，应该用大的种群来增加种群的多样性。
- 初始化：
  - 随机产生一个项目的排列顺序，作为一个染色体。

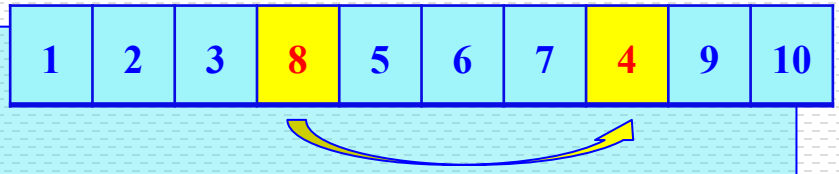
求解基本背包问题的GA—二进制表达法

- 染色体编码生成实例

输入: 基因个数 n
输出: 染色体 v_k
begin

```
for j = 1 to n                          // step 0
    v_k(j) ← j;                         // 基因值不会重复
for i = 1 to ⌈n/2⌉                      // step 1
    repeat
        j ← random[1, n];
        l ← random[1, n];
    until l ≠ j
    swap (v_k(j), v_k(l));              //随机交换两个随机位置的基因
output the chromosome v_k.              // step 2
```

end



- 上述处理过程可以保证染色体各基因的值是随机的，且不会重复

求解基本背包问题的GA—变长表达法

- 该方法属于直接编码法
  - 染色体中的基因构成一个可行的背包
    - 例如，染色体：

2	6	5

1	6	4	5

都是可行的背包。

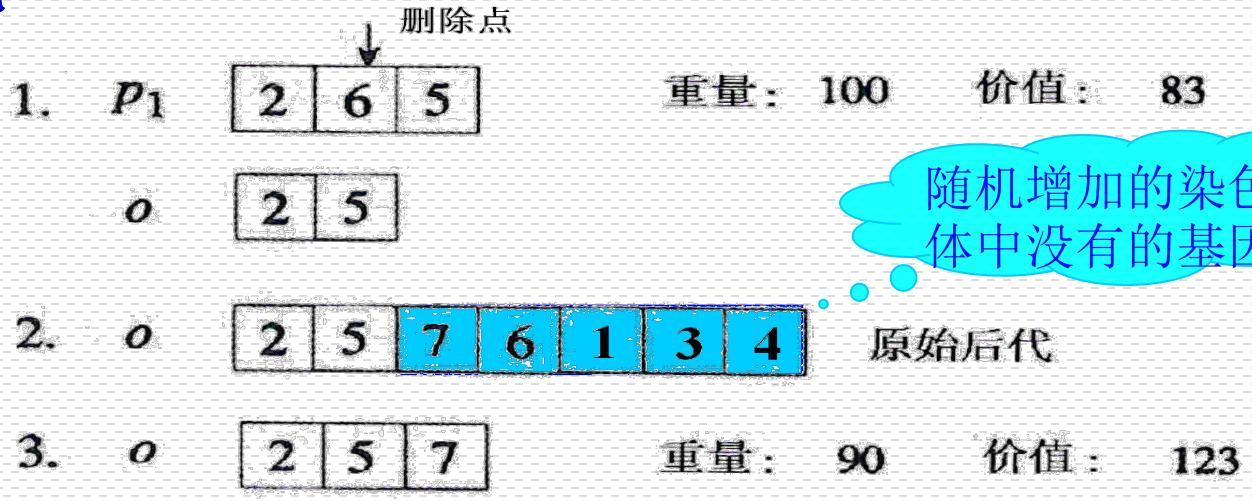
  - 由于可行背包中的项目数是不固定的，所以染色体的长度是可变的，而且染色体中项目的顺序没有意义。2.6.5， 2.5.6 都是一样的。
- 初始化过程：
  - 步骤1: 产生一个随机的项目顺序
  - 步骤2: 基于这个顺序按优先适合启发式构造一个可行背包。

求解基本背包问题的GA—二进制表达法

- 插入交叉
  - 用来处理变长染色体。计算步骤：
    - 在第一个双亲上选择一个断点，在第二个双亲上选一截片断;
    - 将片断插入第一个双亲的断点处;
    - 删除片断外的重复基因，得到一个原始后代，
    - 按优先适合启发式从原始后代中选择基因，即得到一个可行背包

求解基本背包问题的GA—二进制表达法

- 变异过程
  - 计算步骤：
    - 随机地删除一个基因
    - 以随机的顺序增加染色体中没有的基因（所有的基因）
    - 对于以上形成的原始后代用优先适合启发式产生一个可行的背包



  - 问题：
    - 因为染色体只含有选入背包的项目，变异只能引进新的项目
    - 因此要探索其他项目的选择组合，需要更多的遗传代数

求解基本背包问题的GA-适值函数

- 适值函数的设计，直接影响遗传算法的进化操作，对算法的性能有很大的影响。
- 适值设计不当，相差太悬殊的话，就可能导致早熟的问题，也就是算法过早地收敛于一个局部可行解。
  - 例如：一个染色体的适值是100，其它染色体的适值都小于1，如果采用滚花轮选择的话，可能新的种群中只有适值为100的染色体了。
- 适值函数的设计：

$$eval(v) = \frac{\sum_{j=1}^{n} p_j x_j}{\sum_{j=1}^{n} p_j}$$

  - 它给出了一个 0 和 1 之间的适值，通常该值总是小于 1，除非背包能装下所有项目。
  - 这种适值函数的设计，不会出现适值相差太悬殊的情况，因而可以避免早熟问题的出现。

GA求解多选择背包问题

- Gen等人提出的方法
- 染色体编码：
  - 对于多选择背包问题来说，某种排列对于编码来说更加自然也更为有效。
  - 一个染色体由 m 个基因组成，对应着 m 个类。
  - 第 i 个基因从相互排斥集合 N_i 中选取整数， N_i 是第 i 类所包含的项目的集合。
  - 基因的位置用来表示物品的类，而基因的值则用来表示从类中选出的项目。
  - 定义指示变量 y_i：

$$y_i = j \quad \text{if } x_{ij} = 1,\ j \in N_i$$

  - 排列编码：

1	2	3	4	…	m
y_1	y_2	y_3	y_4	…	y_m

GA求解多选择背包问题

- 例：

$$\min\ f(x)=x_{11}+4x_{12}+5x_{13}+7x_{14}+7x_{21}+9x_{22}+10x_{23}+5x_{31}+6x_{32}+11x_{33}$$

$$\text{s.t.}\ \ g_w(x)=10x_{11}+8x_{12}+5x_{13}+4x_{14}+6x_{21}+5x_{22}+3x_{23}+8x_{31}+6x_{32}+4x_{33}\le 16$$

$$g_1(x)=x_{11}+x_{12}+x_{13}+x_{14}=1$$

$$g_2(x)=x_{21}+x_{22}+x_{23}=1$$

$$g_3(x)=x_{31}+x_{32}+x_{33}=1$$

- 问题的可行解：
  - $x_{12}=1,\ x_{23}=1,\ x_{33}=1.$

$$g_1(x)=1,\ g_2(x)=1,\ g_3(x)=1$$

$$g_w(x)=15\le 16$$

$$f(x)=26$$

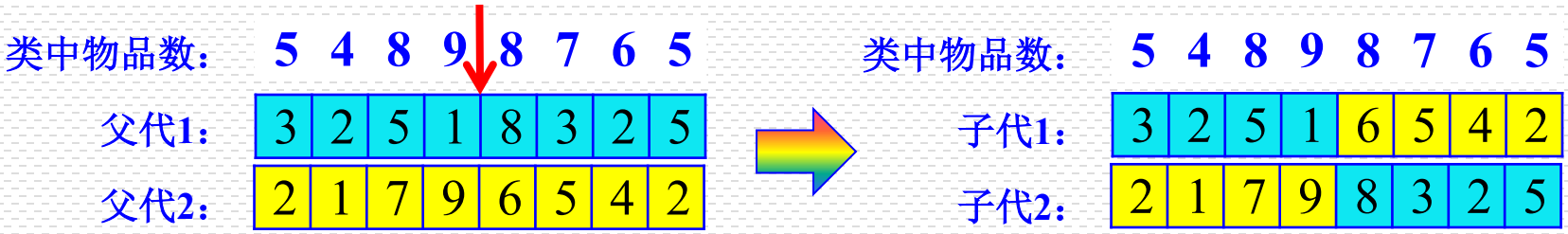
- 染色体编码为：

1	2	3
2	3	3

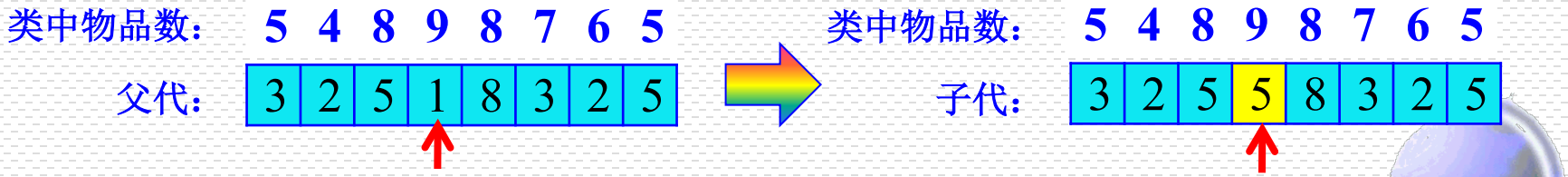
- 种群初始化：
  - 随机生成，第 i 个基因的值，从 $1\sim N_i$ 中随机选择。

GA求解多选择背包问题

- 交叉
  - 可采用传统的单点、两点、均匀交叉等



    - 由于各基因位的值，不超过相应类中物品的个数，所以不论如何交叉，其结果都是有效的染色体。
- 变异
  - 随机扰动。
    - 被选中的基因 y_i 用集合 N_i 中随机选出的一个整数替代；



    - 可保证变异后的染色体仍为有效的染色体。

GA求解多选择背包问题

- 评价
  - 适值函数由两部分构成
    - 总费用（计算出来的目标函数）：

$$f(v_k)=\sum_{i=1}^{m}\sum_{j\in N_i}c_{ij}x_{ij}^{k}$$

    - 对不可行解的惩罚

$$p_k=\begin{cases}0, & \text{满足约束}\\ \alpha_0(g_w(x^k)-W), & \text{不满足约束}\end{cases}$$

where α_0 is the large positive penalty value.

  - 适值函数

$$eval(v_k)=\frac{1}{f(v_k)+p_k},\quad k=1,2,\ldots,popSize$$

GA求解多选择背包问题

- 选择
  - 将最优性 (elitist) 方法嵌入轮盘赌 (roulette wheel) 实现选择操作，可以增强其在下一代中保持最优染色体的能力并克服采样中的随机误差。
  - 如果上一代的最优个体没有被轮盘赌在下一代中得到复制，则从下一代中随机去掉一个染色体并将上一代中的最优染色体加入到下一代中。

基本背包问题的求解方法-动态规划算法

- 动态规划概述
- 多阶段决策过程及特点
- 有关概念及描述
- 动态规划方法
- 动态规划应用

End