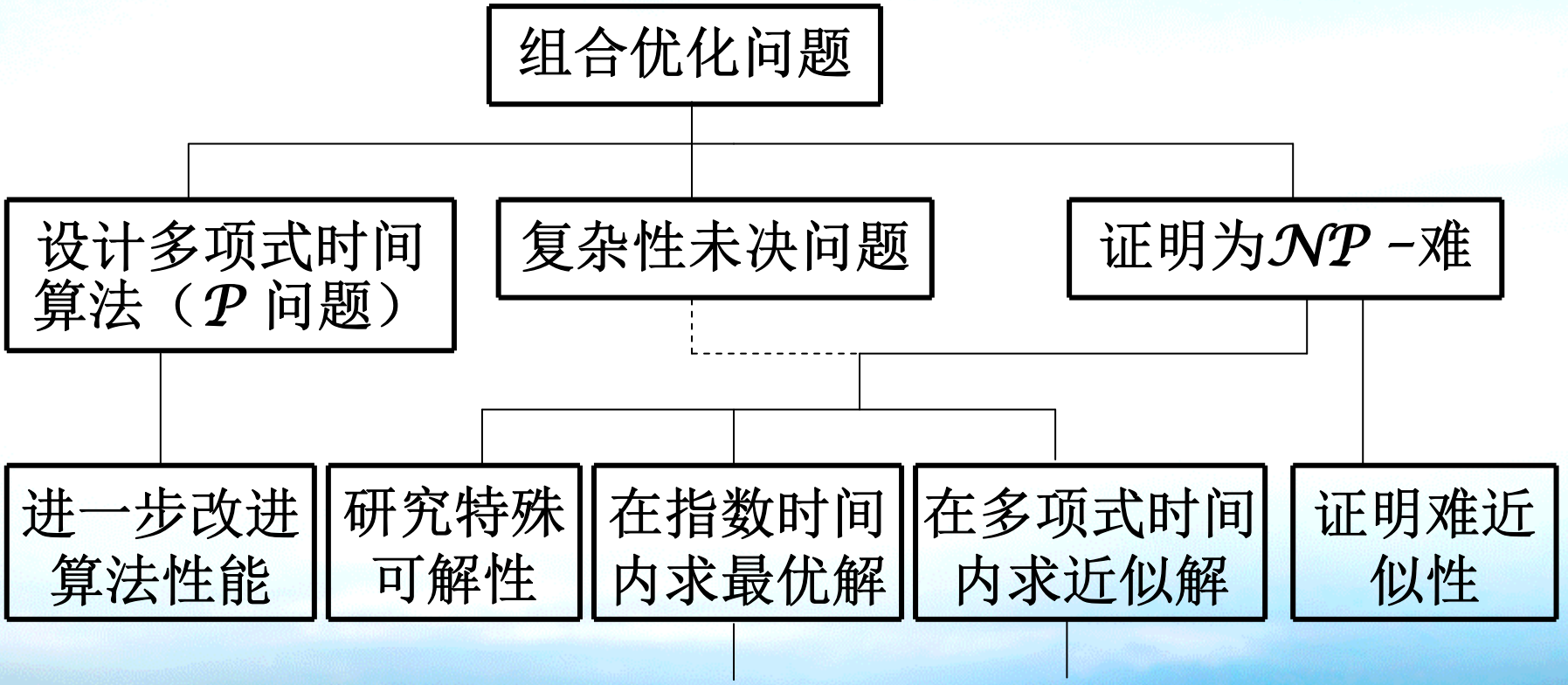
组合优化

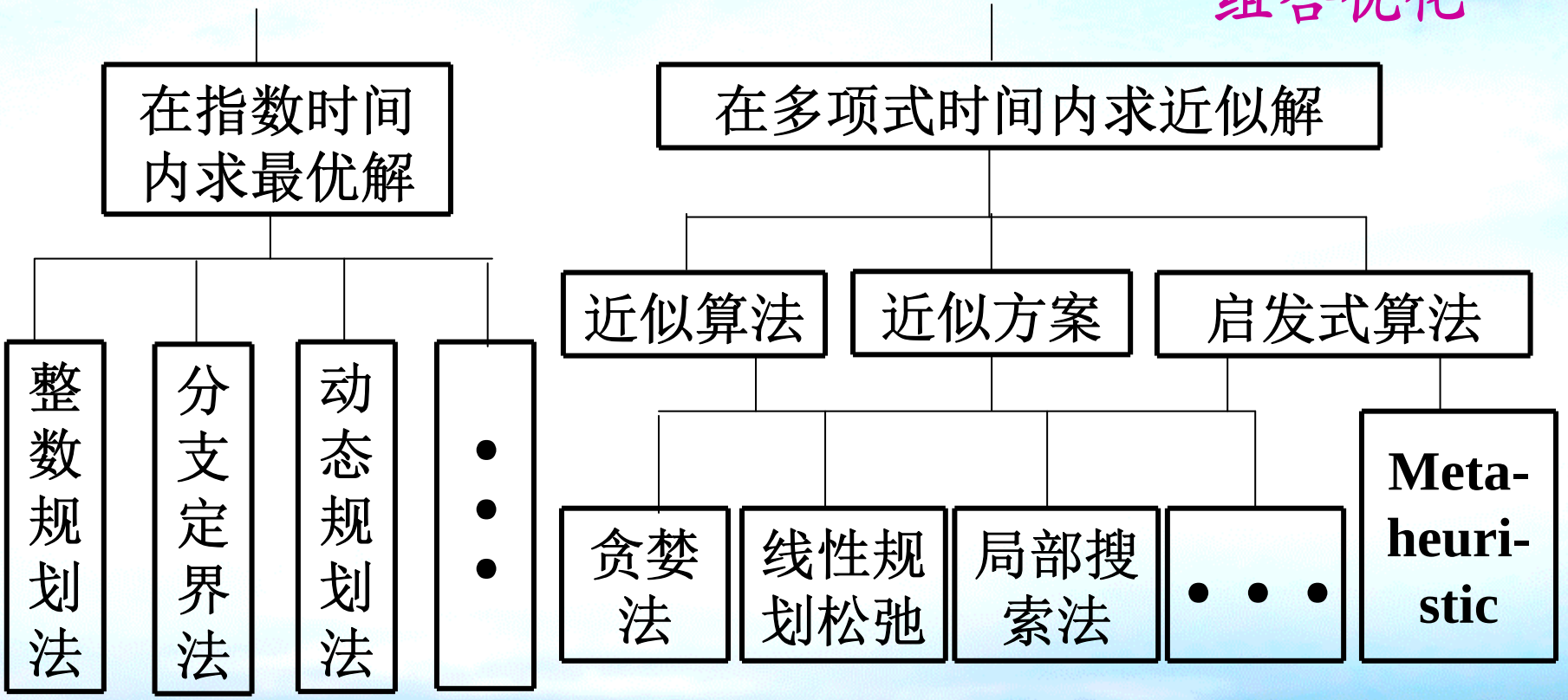
浙江大学数学系　谈之奕

组合优化问题的求解方法

组合优化问题的求解方法



组合优化问题的求解方法



整数规划

- 组合优化问题多可用整数规划描述。建立整数规划的主要步骤有确定决策变量、给出目标函数、列出约束条件
- 整数规划求解也是$\mathcal{NP}$—难的，但可借助整数规划的算法或软件求解一些具体的实例，也可利用整数规划的理论和方法分析解决问题
- 一个组合优化问题可能存在多个整数规划描述，需根据实际情况选择、完善和优化。复杂组合优化问题实例的整数规划求解往往需结合问题性质

LINGO 15.0：
Optimization Modeling Software for Linear, Nonlinear, and Integer Programming

CPLEX Optimizer：
High-performance mathematical programming solver for linear programming, mixed integer programming, and quadratic programming

IBM

背包问题

- 现有 n 件物品，物品 j 的价值为 p_j，大小为 w_j，背包容量为 C。要求选择若干物品放入背包，在放入背包物品大小之和不超过背包容量前提下使放入背包物品价值之和尽可能大
  - 决策变量 $x_j = \begin{cases} 1 & \text{放入第 } j \text{ 种物品} \\ 0 & \text{其他} \end{cases}$ $j = 1, 2, \cdots, n$
  - 整数规划

$$\max \sum_{j=1}^{n} p_j x_j$$ 放入背包物品价值之和

$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C$$ 放入背包物品大小之和不超过 C

$$x_j = 0,1, \quad j = 1, \cdots, n$$

指派问题

- 设有 n 项任务需分配给 n 位员工，每人完成其中一项，员工 i 完成任务 j 所需时间为 c_{ij}，如何分配可使完成所有任务所用总时间最少
  - 决策变量 $x_{ij}=\begin{cases}1, & \text{员工 } i \text{ 被分配完成工作 } j\\ 0, & \text{其他}\end{cases}$ $i,j=1,2,\cdots,n$
  - 整数规划

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$ 完成所有任务所用总时间

$$s.t. \quad \sum_{j=1}^{n} x_{ij}=1,\ i=1,\cdots,n$$ 每位员工完成一项工作

$$\sum_{i=1}^{n} x_{ij}=1,\ j=1,\cdots,n$$ 每项工作由一位员工完成

$$x_{ij}=0 \text{ 或 } 1,\ i,j=1,\cdots,n$$

整数规划 $\in \mathcal{NP}$-C

指派问题 $\in \mathcal{P}$?

指派问题

- 记决策变量向量为 $\mathbf{x}=(x_{11},x_{12},\cdots,x_{1n},x_{21},\cdots,x_{2n},\cdots,x_{n1},\cdots,x_{nn})^T$，整数规划 $\min\{\mathbf{cx}\mid\mathbf{Ax}=\mathbf{b},x_{ij}\in\{0,1\}\}$ 的系数矩阵为

$$\mathbf{A}=\begin{pmatrix}1&1&\cdots&1&0&0&\cdots&0&&0&0&\cdots&0\\0&0&\cdots&0&1&1&\cdots&1&\cdots&0&0&\cdots&0\\\vdots&\vdots&&\vdots&\vdots&\vdots&&\vdots&&\vdots&\vdots&&\vdots\\0&0&\cdots&0&0&0&\cdots&0&&1&1&\cdots&1\\&&&&&&&&&&&&\\\mathbf{I}&&&&\mathbf{I}&&&&\cdots&&\mathbf{I}&&\end{pmatrix}$$

$$\min\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}$$

$$s.t.\quad\sum_{j=1}^{n}x_{ij}=1,\ \ i=1,\cdots,n$$

$$\sum_{i=1}^{n}x_{ij}=1,\ \ j=1,\cdots,n$$

$$0\le x_{ij}\le 1\quad i,j=1,\cdots,n$$

- 若矩阵的所有子式均为**0**或 ±1，则称该矩阵为全幺模（**totally unimodular**）矩阵
- 系数矩阵为全幺模矩阵的整数规划的松弛线性规划的最优解必为整数解

TSP问题

浙江大学 ZheJiang University

组合优化

- 决策变量 $x_{ij}=\begin{cases}1, & \text{离开城市 } i \text{ 后到达城市 } j\\ 0, & \text{其他}\end{cases}$ $i,j=1,2,\cdots,n$

决策变量能表示环游，同时能计算环游长度

- 整数规划

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \quad \sum_{j=1}^{n} x_{ij}=1, \quad i=1,\cdots,n$$ 离开城市 i 后到达另一个城市

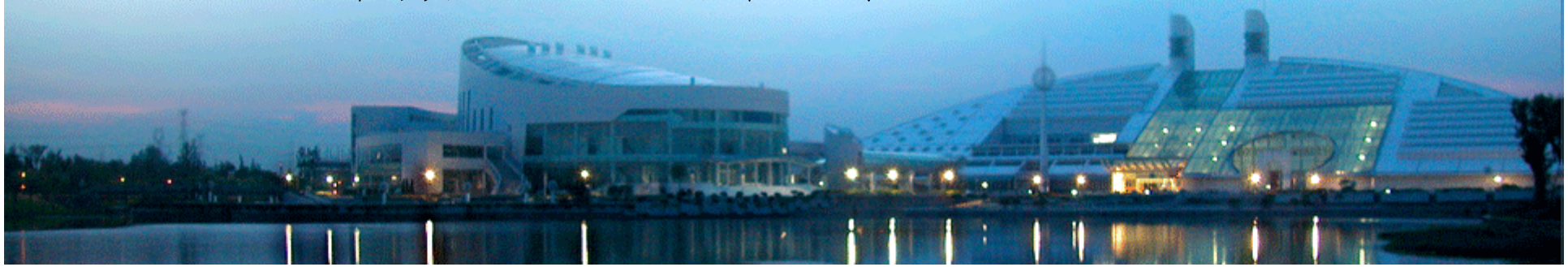
$$\sum_{i=1}^{n} x_{ij}=1, \quad j=1,\cdots,n$$ 从一个城市来到城市 j

$$x_{ij}=0,1,\ i,j=1,\cdots,n$$

问题可行解与其整数规划可行解之间未一一对应

指派问题 $\in \mathcal{P}$

TSP $\in \mathcal{NP}$-C ?



TSP问题

- **TSP**环游不允许出现经过城市数小于 n 的子环游
- 若存在一相继经过 k 个城市 $i_1, i_2, \cdots, i_k$ 的子环游，则

$$x_{i_1 i_2} = x_{i_2 i_3} = \cdots = x_{i_{k-1} i_k} = x_{i_k i_1} = 1$$

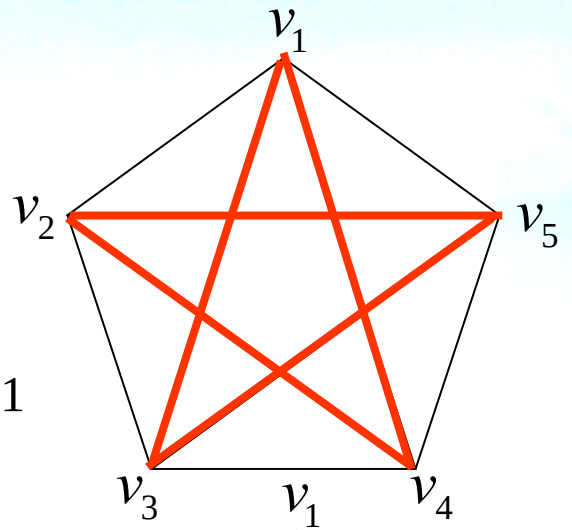
$$\sum_{i,j \in \{i_1, i_2, \cdots, i_k\}} x_{ij} \geq k$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \cdots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \cdots, n$$

$$x_{13} = x_{35} = x_{52} = x_{24} = x_{41} = 1$$

$$x_{ij} = 0, (i, j) \notin \{(1,3), (3,5), (5,2), (2,4), (4,1)\}$$

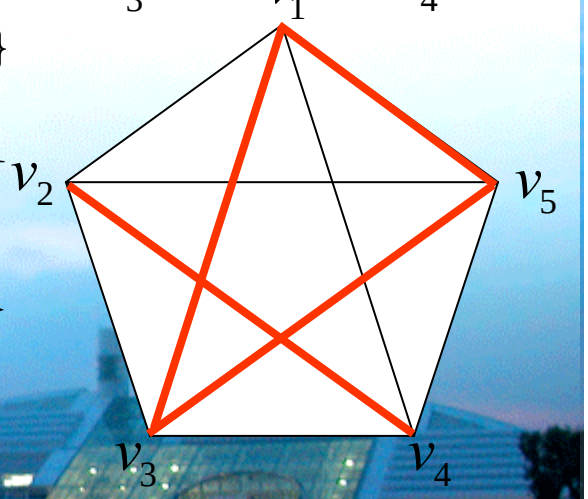


- 若不存在相继经过 k 个城市 $i_1, i_2, \cdots, i_k$ 的子环游，则

$$\sum_{i,j \in \{i_1, i_2, \cdots, i_k\}} x_{ij} < k - 1$$

$$x_{13} = x_{35} = x_{51} = x_{24} = x_{42} = 1$$

$$x_{ij} = 0, (i, j) \notin \{(1,3), (3,5), (5,1), (2,4), (4,2)\}$$



TSP问题的整数规划

- 为使整数规划的解不存在任意子环游，需增加约束以消除子环游

$$\sum_{i,j\in S} x_{ij} \le |S| - 1, \quad \forall \varnothing \ne S \subseteq \{2,3,\cdots,n\}$$

  - 增加约束后的整数规划约束个数为 $O(2^n)$
- 增加 $n-1$ 个实决策变量 $u_2, u_3, \cdots, u_n$ 和 $(n-1)^2$ 个约束

$$(n-1)x_{ij} + u_i - u_j \le n-2, \quad i, j = 2,3,\cdots,n$$

也可消除子环游

Miller CE, Tucker AW, Zemlin RA. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7, 326-329, 1960

MTZ整数规划

- 若解含有子环游，则约束不被满足
  - 若含有子环游，则必有一子环游 $(i_1\ i_2\cdots i_k)$不经过城市 1
  - 若该子环游仅含一个城市 i，则 $x_{ii}=1$

$$(n-1)x_{ii}+u_i-u_i=n-1>n-2$$

矛盾

  - 若该子环游至少含有两个城市，则 $x_{i_j i_{j+1}}=1\left(i_{k+1}=i_1\right)$,

$$(n-1)x_{i_j i_{j+1}}+u_{i_j}-u_{i_{j+1}}=n-1+u_{i_j}-u_{i_{j+1}},\ j=1,2,\cdots,k$$

$$\sum_{j=1}^{k}\left((n-1)x_{i_j i_{j+1}}+u_{i_j}-u_{i_{j+1}}\right)=\sum_{j=1}^{k}\left(n-1+u_{i_j}-u_{i_{j+1}}\right)=k(n-1)>k(n-2)$$

$$(n-1)x_{ij}+u_i-u_j\le n-2,\ \ i,j=2,3,\cdots,n$$

矛盾

MTZ整数规划

- 对任意可行环游，存在一组可行解满足约束
  - 对任意可行环游，选定城市 1为起点，若城市 i是环游中的第 k 个经过的城市，取 $u_i = k$，$2 \le u_i \le n$，x_{ij}的取值同前定义
  - 若对某个 i, j 组合，$x_{ij} = 0$，则

$$(n-1)x_{ij} + u_i - u_j = u_i - u_j \le n-2$$

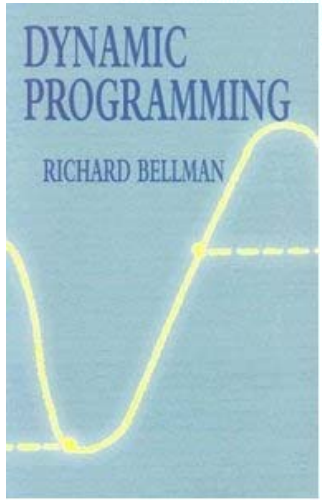
  - 若对某个 i, j 组合，$x_{ij} = 1$，则 $u_i - u_j = -1$，

$$(n-1)x_{ij} + u_i - u_j = (n-1) - 1 = n-2$$

$$(n-1)x_{ij} + u_i - u_j \le n-2,\ i, j = 2, 3, \cdots, n$$

动态规划

- 动态规划（**dynamic programming，DP**）是求解多阶段决策优化问题的一种数学方法和算法思想
- 动态规划的基本思路是将需求解的实例转化为规模较小的实例，并利用递推关系导出两者最优解之间的关系，从而可由初始条件出发逐步求得最优解

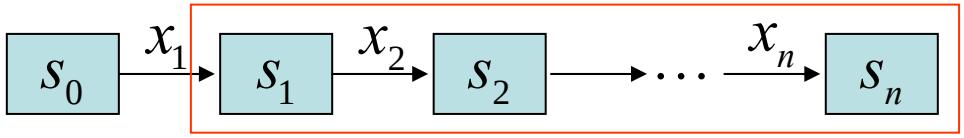


Bellman RE, *Dynamic Programming*, Princeton University Press, 1957

Richard Ernest Bellman (1920-1984)
美国运筹学家

最优化原理

- 最优化原理（**Principle of Optimality**）
  - 一个过程的最优决策 $(x_1^*, x_2^*, \cdots, x_n^*)$具有如下的性质：无论初始状态 s_0 及初始决策 x_1如何确定，之后的决策$(x_2^*, \cdots, x_n^*)$对于以 x_1^*所造成的状态 s_1^*为初始状态的后部过程而言，必为最优策略



$$f_n(s_0; x_1, x_2, \cdots, x_n)$$

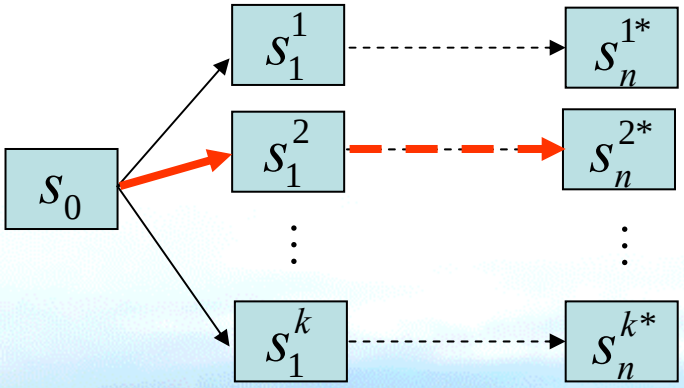
$$f_n(s_0; x_1^*, x_2^*, \cdots, x_n^*) = f_1(s_0; x_1^*) + f(s_1^*; x_2^*, \cdots, x_n^*)$$

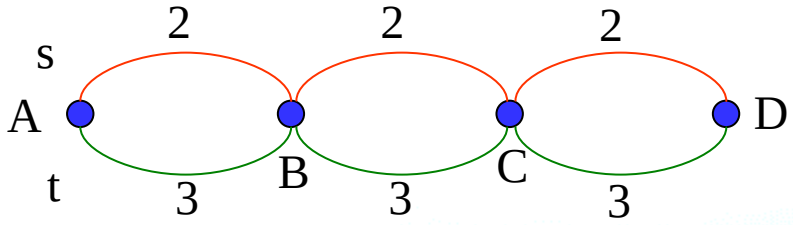
$$f_n(s_0; x_1^*, \hat{x}_2, \cdots, x_n) = f_1(s_0; x_1^*) + f(s_1^*; \hat{x}_2, \cdots, x_n)$$

PRINCIPLE OF OPTIMALITY. *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

最优化原理

- 最优化原理（**Principle of Optimality**）：一个 n 阶段过程的最优策略可以这样构成：首先求出以初始决策 x_1 造成的状态 s_1 为初始状态的 $n-1$ 阶段子过程的最优策略，然后在附加第一阶段效益（或费用）的情形下，从所有可能初始决策得到的解中选择最优者





最短路的子路也是最短路

路长定义为边长的模4加法时，t为A到D的最短路，但C到D的最短路仍为s，最优化原理不再成立

背包问题的动态规划

组合优化

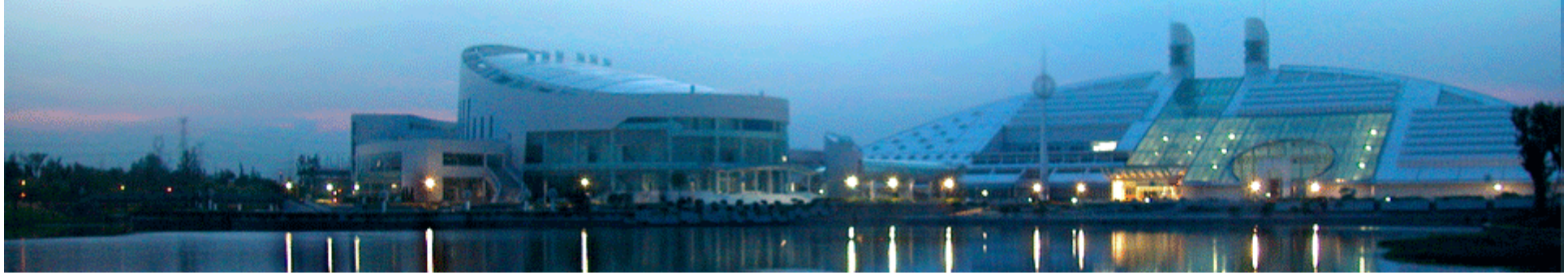
- 依次考虑所有物品，每个物品对应动态规划的一个阶段，可能决策为该物品放入与不放入两种
- 构造一系列背包问题实例 $I(k,w), 0 \le k \le n, 0 \le w \le C$，其中背包容量为 w，物品数为 k，包含原实例中的前 k 个物品
- 记 $C^*(k,w)$ 为实例 $I(k,w)$ 的最优解，原实例的最优值即为 $C^*(n,C)$

递推关系

- $C^*(k,w)$ 满足递推关系

$$C^*(k,w)=\begin{cases} C^*(k-1,w) & \text{若} w_k > w \\ \max\{C^*(k-1,w), C^*(k-1,w-w_k)+p_k\} & \text{若} w_k \le w \end{cases}$$

  - 若 $w_k > w$，则物品 k 不能放入容量为 w 的背包中，因此 $I(k,w)$ 的最优值与 $I(k-1,w)$ 的最优值必相同
  - 若 $w_k \le w$，则物品 k 可以放入容量为 w 的背包中，$I(k,w)$ 的最优解有两种可能
    - 若物品 k 未放入背包，$I(k,w)$ 的最优值必与 $I(k-1,w)$ 的最优值相同
    - 若物品 k 放入背包，背包剩余容量为 $w-w_k$，可放入的物品必为 $I(k-1,w-w_k)$ 的最优解中放入的物品



动态规划DPS

- $C^*(k,w)$ 满足初始条件

$$C^*(0,w)=0,\ w=0,\cdots,C,\ C^*(k,0)=0,\ k=0,\cdots,n$$

- 由初始条件和递推关系可逐步求得 $I(n,C)$，外层循环为 k 自 0到 n，内层循环为 w 自 0到 C
- 动态规划**DPS**的时间复杂性为 $O(nC)$，背包问题的实例规模为 $n+\lceil \log_2 L \rceil$，其中 $L=\max\left\{\max_{j=1,\cdots,n} p_j, \max_{j=1,\cdots,n} w_j, C\right\}$，因此**DPS**是伪多项式时间算法，背包问题是普通意义下的 $\mathcal{NP}$ −完全问题

动态规划DPS

- 背包问题实例

  $C=5, n=4$

  $p_1=3, p_2=4, p_3=5, p_4=6$

  $w_1=2, w_2=3, w_3=4, w_4=5$

- 最优值为 7，最优解为物品1，2 放入背包

w \ k	0	1 +3	2 +4	3 +5	4 +6
0	0	0	0	0	0
1	0	0	0	0	0
2 w_1	0	3	3	3	3
3 w_2	0	3	4	4	4
4 w_3	0	3	4	5	5
5 w_4	0	3	7	7	7

$$C^*(k,w)=\begin{cases} C^*(k-1,w) & \text{若} w_k > w \\ \max\left\{C^*(k-1,w), C^*(k-1,w-w_k)+p_k\right\} & \text{若} w_k \le w \end{cases}$$

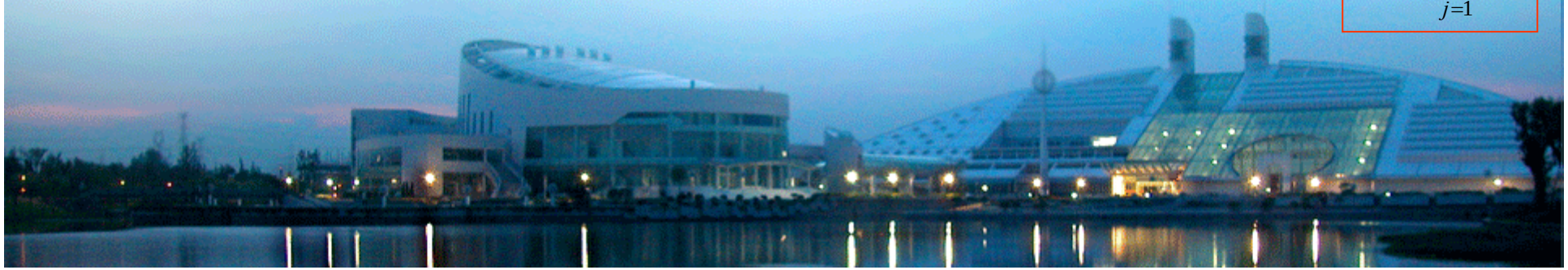
动态规划DPV

- 记 $y(k,q)$ 为在前 k 个物品中选出若干物品，使其价值之和恰为 q 的前提下，选出物品大小之和所能取到的最小值
  - 若价值之和恰为的物品子集不存在，则令 $y(k,q)=C+1$
  - 原实例的最优值即为 $\max\{q \mid y(n,q)\le C\}$
- $y(k,q)$ 满足递推关系

$$y(k,q)=\begin{cases} y(k-1,q), & \text{若} q<p_k, \\ \min\{y(k-1,q),y(k-1,q-p_k)+w_k\}, & \text{若} q\ge p_k. \end{cases}$$

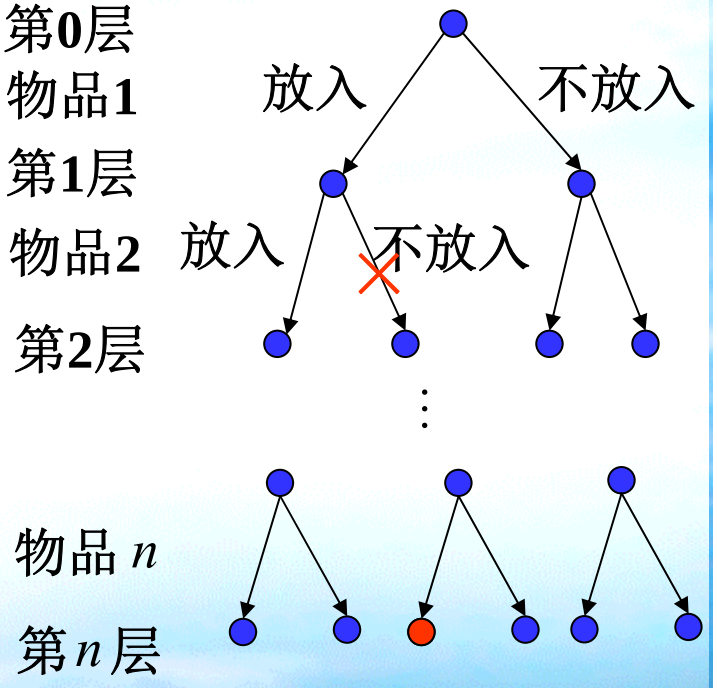
- $y(k,q)$ 满足初始条件 $y(k,0)=0,k=0,\cdots,n,y(0,q)=C+1,q=1,\cdots,P$
- **DPV的时间复杂性为 $O(nP)$，与DPS同为伪多项式时间算法，两者的时间复杂性不存在明确的优劣关系**

$$P=\sum_{j=1}^{n}p_j$$



分枝定界法

- 背包问题的分枝定界法
  - 实例求解过程可用一颗高度为 n 的二叉树表示。各层依次对应一个物品，最后一层的每个节点对应一个解
  - 分枝：每个节点有两条出边连结两个子节点，分别代表该节点下一层对应物品放入背包与不放入背包
  - 剪枝
    - 该枝不存在可行解
      - 按该枝含义确定放入背包的物品大小之和超过背包容量
    - 该枝不存在最优解　　如何提前判断



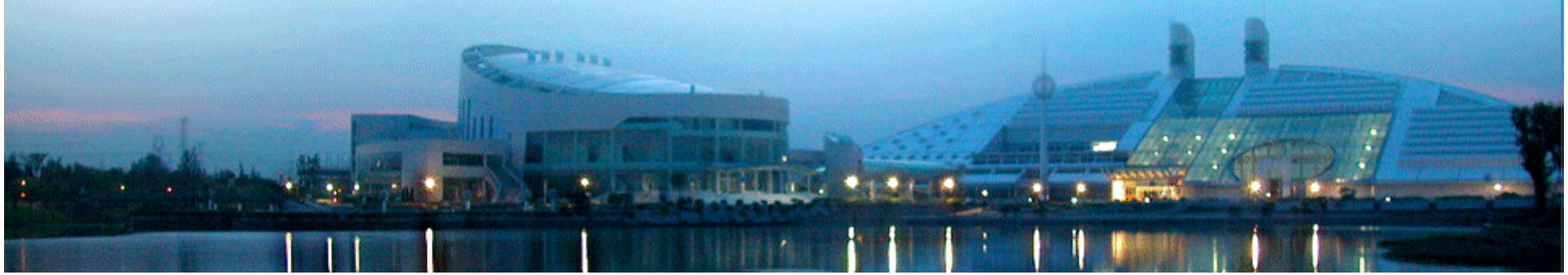
分枝定界法

- 定界　　下界对所有节点均有效，上界仅对该枝有效
  - 任一个已获得的可行解的目标值均是最优值的下界
  - 对每一枝，求该枝所有可行解目标值的上界，若该上界不大于下界，则该枝不存在更好的可行解
    - 按该枝含义，所有确定已放入物品和所有未确定物品价值之和

      上界越小越好，下界越大越好

      上界和下界的计算尽量简单
    - 该枝对应的整数规划的松弛线性规划的最优值

      只考虑未确定物品



松弛线性规划

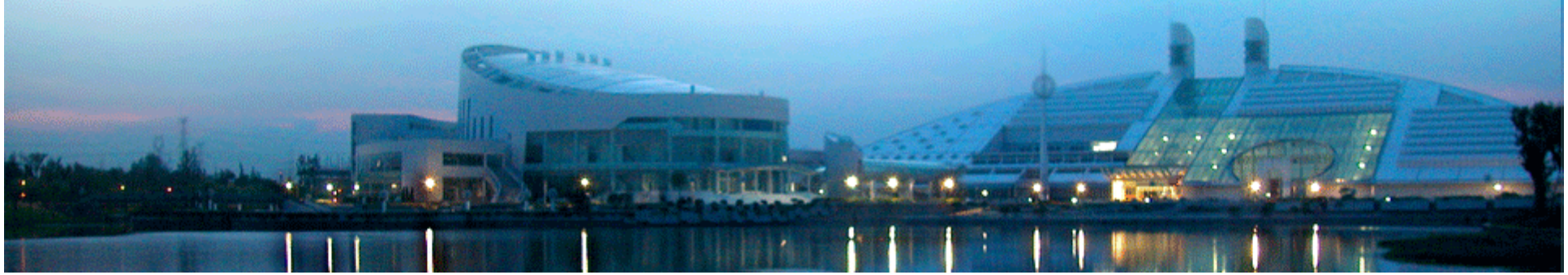
- 松弛线性规划最优解 $\mathbf{x}=(x_1^*,x_2^*,\cdots,x_n^*)^T$ 的性质
  - $\sum_{j=1}^{n} w_j x_j^* = C$
    - 若不然，必存在 $i, x_i^* < 1$。令 $x_i' = x_i^* + \varepsilon$ 可得一更好的解
  - 若 $\frac{p_i}{w_i} > \frac{p_k}{w_k}$，则若 $x_i^* < 1$，则 $x_k^* = 0$
    - 若不然，令 $x_i' = x_i^* + \frac{w_k}{w_i}\varepsilon, x_k' = x_k^* - \varepsilon$，

      $$\sum_{j=1}^{n} p_j x_j' - \sum_{j=1}^{n} p_j x_j^* = p_i \frac{w_k}{w_i}\varepsilon - p_k\varepsilon > 0$$ 矛盾
  - $\frac{p_j}{w_j}$ 称为物品 j 的价值密度，最优解应优先放入价值密度大的物品

$$\max \sum_{j=1}^{n} p_j x_j$$
$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C$$
$$x_j = 0,1, \quad j = 1,\cdots,n$$

$$\max \sum_{j=1}^{n} p_j x_j$$
$$s.t. \quad \sum_{j=1}^{n} w_j x_j \le C$$
$$0 \le x_j \le 1, \quad j = 1,\cdots,n$$

$$p_j > 0, j = 1,\cdots,n, \sum_{j=1}^{n} w_j > C$$

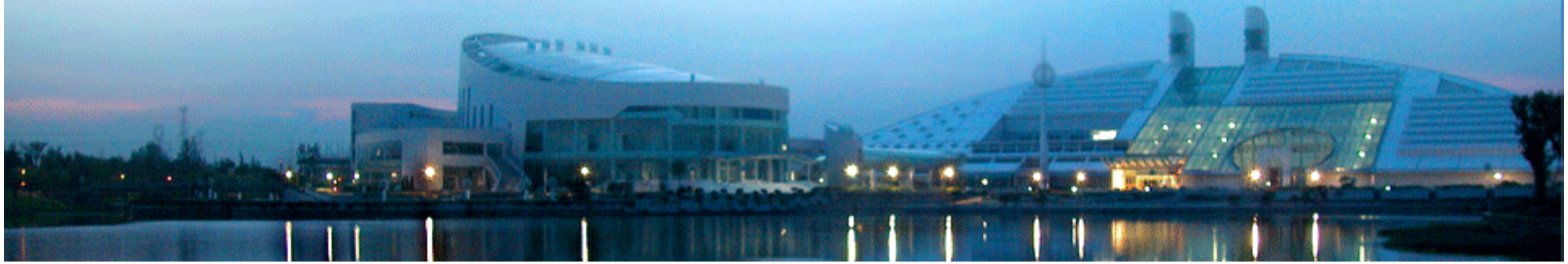


松弛线性规划

- 假设物品按价值密度非增顺序排列，即 $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}$
  - 若 $\frac{p_i}{w_i} = \frac{p_k}{w_k}$，可将物品 i 和物品 k 合并为一个物品
- 记 $j = \min\left\{k \middle| \sum_{i=1}^{k} w_i > C\right\}$ 为按价值密度非增顺序第一个不能全部放入背包的物品
- 松弛线性规划的最优解为

$$x_i^* = 1, i = 1, 2, \cdots, j-1, \quad x_j^* = \frac{1}{w_j}\left(C - \sum_{i=1}^{j-1} w_i\right), \quad x_i^* = 0, i = j+1, \cdots, n$$

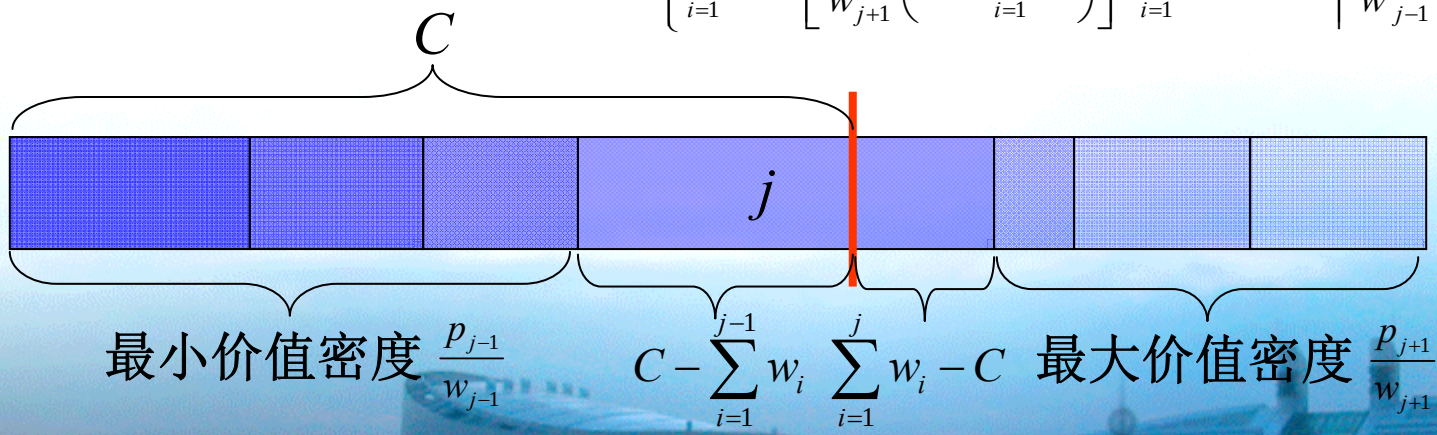
- 松弛线性规划的最优值为 $\sum_{i=1}^{j-1} p_j + \frac{p_j}{w_j}\left(C - \sum_{i=1}^{j-1} w_i\right)$。由于物品价值均为整数，背包（整数规划）最优值的上界也可取为 $UB_1 = \sum_{i=1}^{j-1} p_i + \left\lfloor \frac{p_j}{w_j}\left(C - \sum_{i=1}^{j-1} w_i\right) \right\rfloor$



上界

- 背包最优解可能不放入物品 j 而放入之后大小适中的物品，也可能因放入物品 j 而不得不取出之前的部分物品
  - 若不放入物品 j，最优值上界为 $\sum_{i=1}^{j-1} p_i + \frac{p_{j+1}}{w_{j+1}}\left(C - \sum_{i=1}^{j-1} w_i\right)$
  - 若放入物品 j，最优值上界为 $\sum_{i=1}^{j-1} p_i + p_j - \frac{p_{j-1}}{w_{j-1}}\left(\sum_{i=1}^{j} w_i - C\right)$
- 背包最优值上界可取为 $\max\left\{\sum_{i=1}^{j-1} p_i + \left\lfloor \frac{p_{j+1}}{w_{j+1}}\left(C - \sum_{i=1}^{j-1} w_i\right)\right\rfloor, \sum_{i=1}^{j-1} p_i + p_j - \left\lceil \frac{p_{j-1}}{w_{j-1}}\left(\sum_{i=1}^{j} w_i - C\right)\right\rceil\right\}$

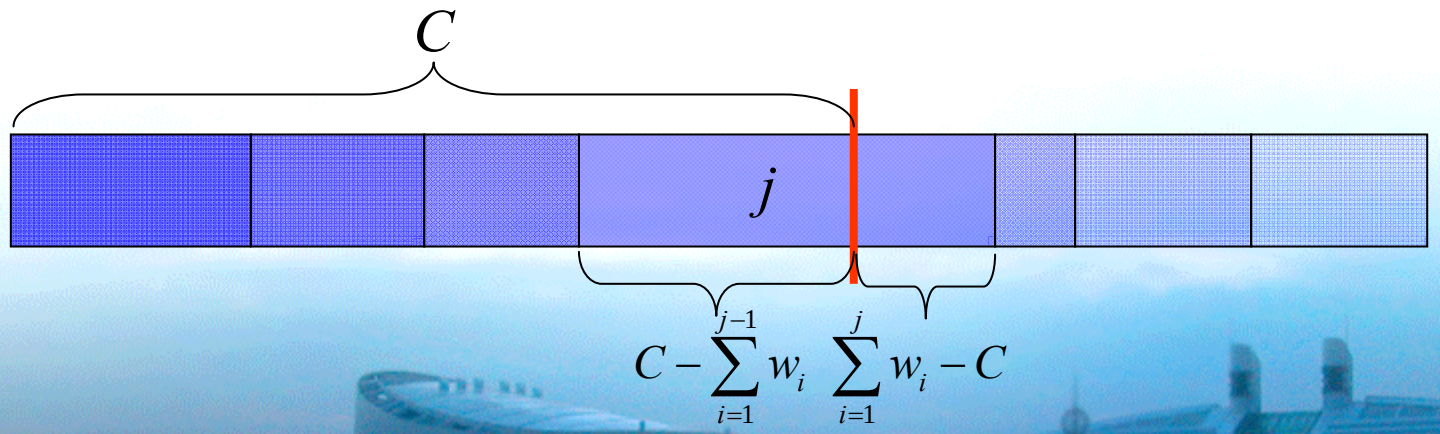
优于 UB_1



下界

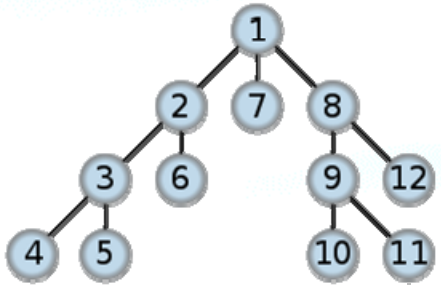
- 放入前 $j-1$ 个物品为可行解，目标值为 $LB_1 = \sum_{i=1}^{j-1} p_i$
- 再放入物品 j 之后某个大小适中的物品，或放入物品 j 而取出之前的某个物品可能为更好的可行解

$$LB_2 = \max\left\{LB_1, \max_{k=j+1,\cdots,n}\left\{LB_1 + p_k \,\middle|\, \sum_{i=1}^{j-1} w_i + w_k \le C\right\}, \max_{k=1,\cdots,j-1}\left\{LB_1 + p_j - p_k \,\middle|\, \sum_{i=1}^{j-1} w_i + w_j - w_k \le C\right\}\right\}$$

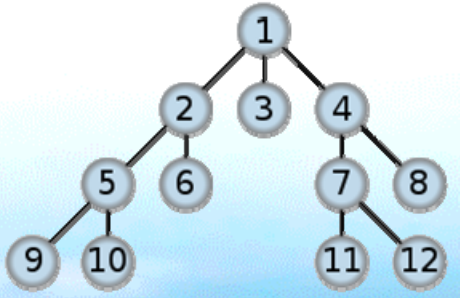


分枝定界法

- 从根节点开始，按深探法（**Depth-first search, DFS**）或广探法（**Breadth-first search, BFS**）给定的顺序检查每个节点
  - 对每个节点，按节点含义计算上界和下界
  - 若下界大于当前下界，则修正当前下界
  - 若当前节点不满足剪枝条件，则进行分枝
- 若所有节点都检查完毕且不可再分枝，当前下界即为实例的最优值
- 分枝定界法是指数时间算法，但实际效果未必劣于伪多项式时间动态规划



深探法



广探法


浙江大学
ZheJiang University
谢　谢