分支限界及其应用

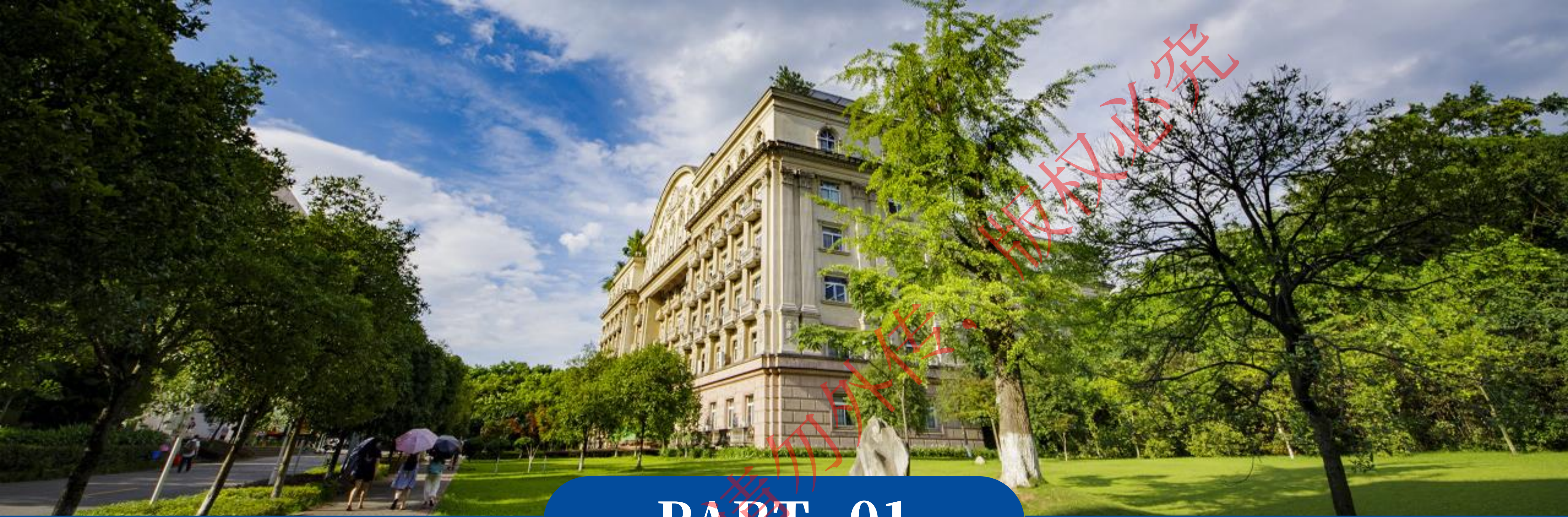
张里博
lbzhang@swu.edu.cn

目录

1 分支限界的思想

2 背包问题

3 最大团问题



PART 01

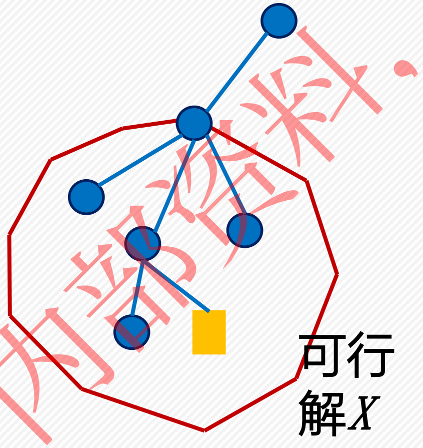
分支限界的思想

Backtracking Algorithm

分支限界

- 分支限界是一种特殊的回溯算法
- 回溯算法适用于求解所有可行解和最优解问题（回溯算法会找到问题的所有可行解），一般采用深度优先等搜索策略，回溯判断是约束条件；
- 分支限界算法只适用于求解最优解的问题，采用深度优先、最小耗费（最大收益）优先等策略，回溯条件是约束条件和最优性条件
- 出发点：
- 回溯条件越多，回溯机会越多，裁剪的分支越多，效率越高

- 含义：当前已得到可行解目标函数的最大值(极小化问题相反）
- 初值：极大化问题初值为 0 (极小化问题为最大值)
- 更新：对极大化问题，如果新的可行解的目标函数值大于当前的界，则把界更新为该可行解的目标函数值



$$B = \max\{v(X)\}$$

代价函数的定义

- 以该点为根的子树所有可行解的目标函数值的上界(极小化问题为下界)

代价函数是一个事前估计值
界函数是一个事后的计算值

- 搜索树上含根节点的所有结点都有代价函数，未延申到底的结点也有代价函数

代价函数唯一么？
找到可行解，才能构造代价函数么？

- 性质：对极大化问题父结点代价不小于子结点的代价 (极小化问题相反)

停止分支，回溯父结点的依据（剪枝）：

1.不满足约束条件；

2.最优性条件对于极大化问题，某点的代价函数值小于等于当前界（极小化问题，大于等于界）；

含弘光大　继往开来

(1) 定义搜索问题的解向量和每个分量的取值集合

解向量为 $<x_1, x_2, \dots, x_n>$

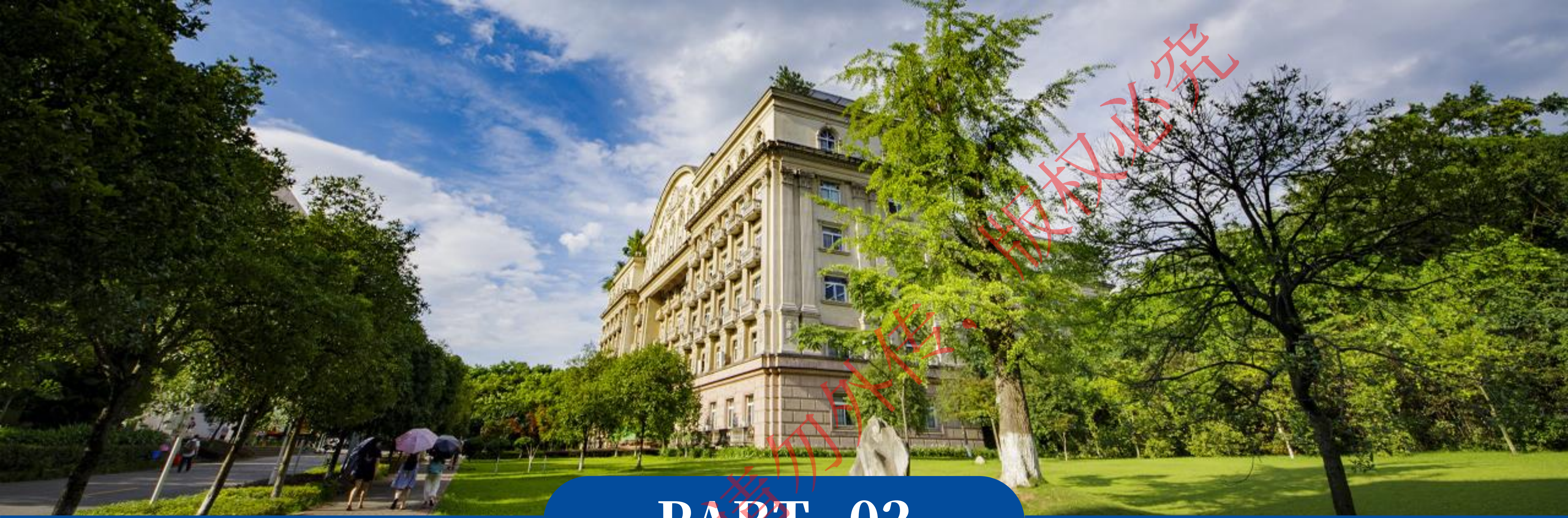
确定 x_i 的理论上可能取值的集合为 X_i, $i = 1, 2, \dots, n$.

(2) 确定结点儿子的排列规则

(3) 判断是否满足多米诺性质

(4) 搜索策略-----最小耗费优先等

(5) 确定每个结点约束条件和限界条件（最优性条件，关键是代价函数的构造）



PART 02

背包问题

Backtracking Algorithm

实例：完全背包问题

有4 种物品可选，重量 w_i 与价值 v_i 分别为

$$v_1 = 1, v_2 = 3, v_3 = 5, v_4 = 9$$

$$w_1 = 2, w_2 = 3, w_3 = 4, w_4 = 7$$

背包重量限制为10，求使得背包价值最大的方案

优化问题可以表示为：

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$

$$2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10$$

$$x_i \in N, i = 1, 2, 3, 4$$

是否满足多米诺性质？

代价函数要如何设计？

背包问题中代价函数

- 对结点x_k $<x_1, x_2, \ldots, x_k>$，构造其代价函数.
- 代价函数 = 已装入物品的价值+ Δ
- 已装入物品的价值：已装入物品的种类和数量

  Δ：使用剩余种类物品可继续装入的最大价值

Δ：背包剩余重量 ×剩余物品种类中单位重量价值的最大值

若一开始，将物品按v_i / w_i从大到小排序，

Δ = 背包剩余重量 × v_{k+1}/w_{k+1}(剩余可装)

代价函数形式

结点 $<x_1, x_2, \ldots, x_k>$ 的代价函数

代价函数 = 已装入物品的价值+ Δ

Δ：背包剩余重量 ×剩余物品种类中单位重量最大值

$$\sum_{i=1}^{k} v_i x_i + \left(b - \sum_{i=1}^{k} w_i x_i\right) \square v_{k+1} / w_{k+1}$$

若对物品 $j > k$ ，若 $b - \sum_{i=1}^{k} w_i x_i \geq w_j$，则第 j 个物品能装进书包；

若 $b - \sum_{i=1}^{k} w_i x_i < w_j$ 则第j个物品不能装进书包（约束）；

实例：背包问题

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$
$$2x_1 + 3x_2 + 4x_3 + 7x_4 \le 10$$
$$x_i \in N, i = 1, 2, 3, 4$$

变元重新排序（单位重量的价值）后，使得 $\dfrac{v_i}{w_i} \ge \dfrac{v_{i+1}}{w_{i+1}}$

$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$
$$7x_1 + 4x_2 + 3x_3 + 2x_4 \le 10$$
$$x_i \in N, i = 1, 2, 3, 4$$

$$0 \le x_i \le \left\lfloor \frac{b}{\omega_i} \right\rfloor$$

每个分量的理论可能取值集合

$$X_1 = \{0,1\};$$
$$X_2 = \{0,1,2\};$$
$$X_3 = \{0,1,2,3\};$$
$$X_4 = \{0,1,2,3,4,5\};$$

实例

$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$

$$7x_1 + 4x_2 + 3x_3 + 2x_4 \le 10, x_i \in N, i = 1,2,3,4$$

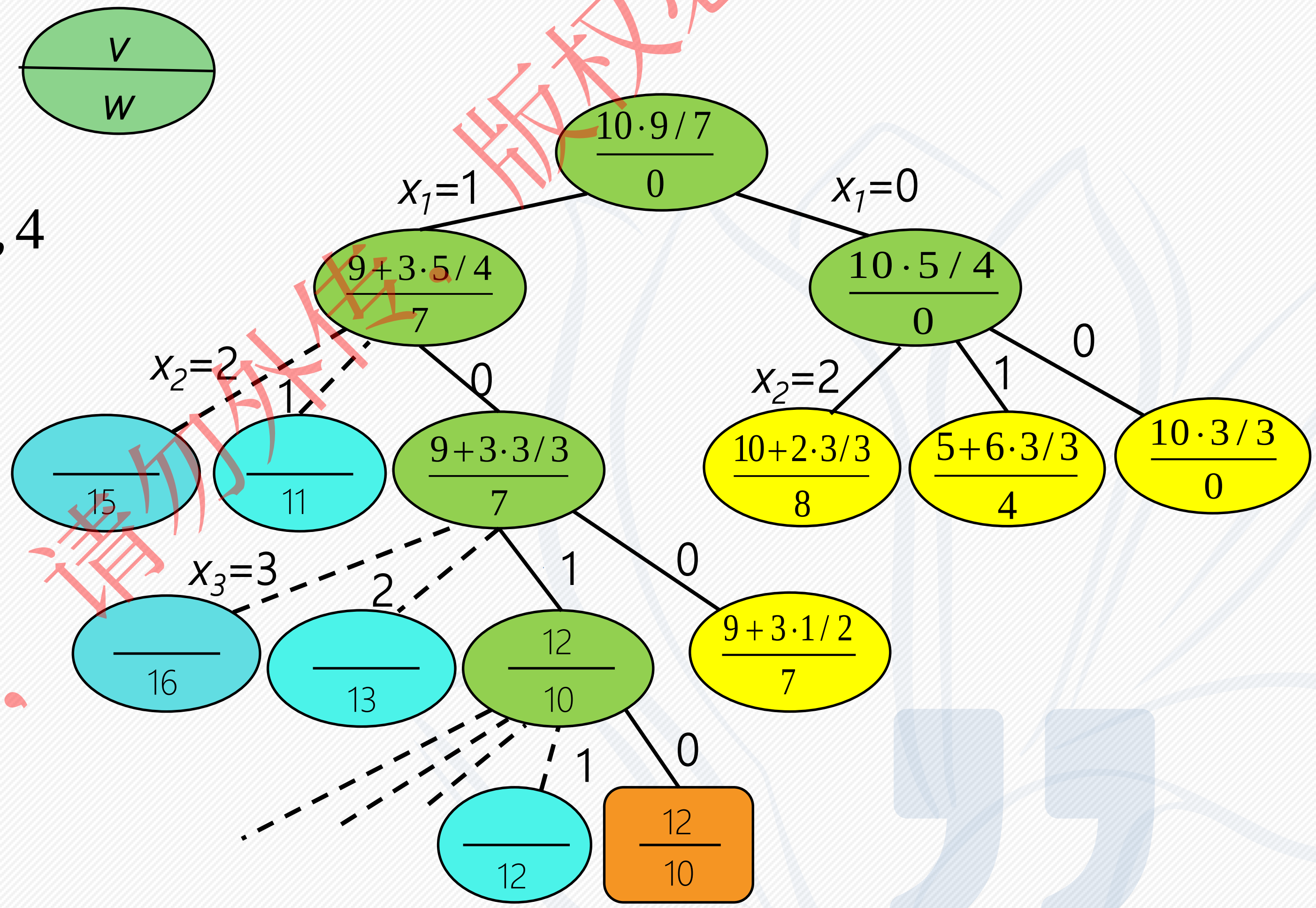
$$0 \le x_i \le \left\lfloor \frac{b}{\omega_i} \right\rfloor$$

$$X_1 = \{0,1\};$$

$$X_2 = \{0,1,2\};$$

$$X_3 = \{0,1,2,3\};$$

$$X_4 = \{0,1,2,3,4,5\};$$



$$\sum_{i=1}^{k} v_i x_i + \left(b - \sum_{i=1}^{k} w_i x_i\right) \square v_{k+1} / w_{k+1}$$

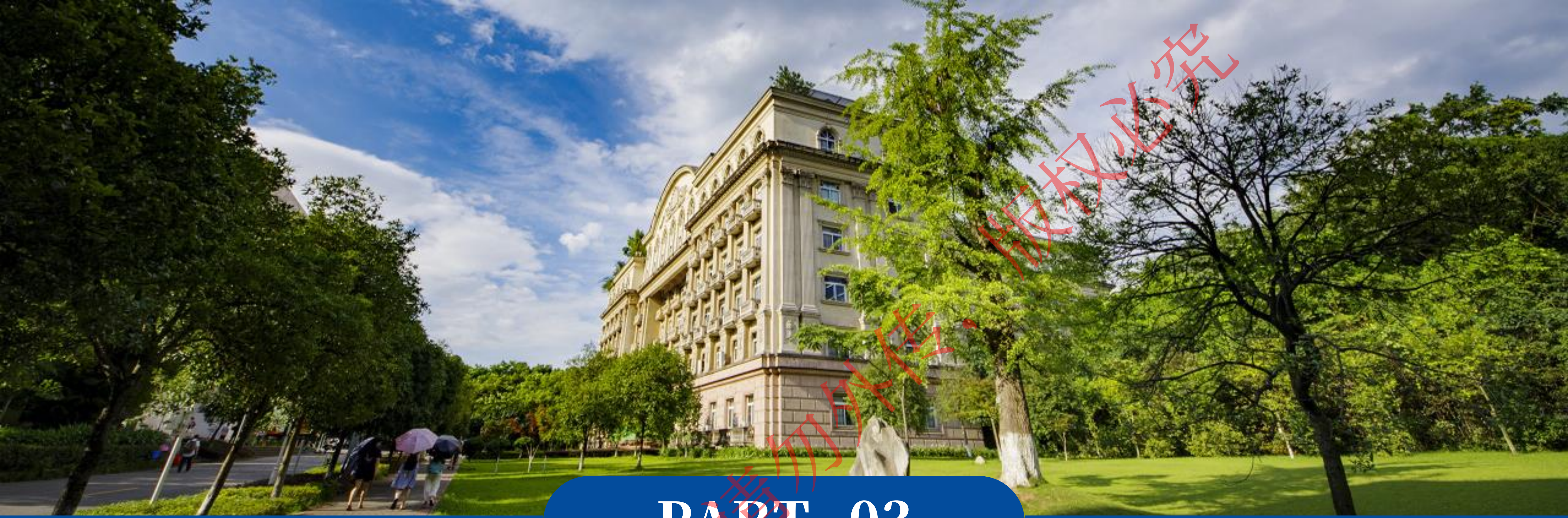
小结

- 分支限界适用于组合优化和搜索问题（分量取值是离散的），而且只能求解问题的最优解

- 对结点$<x_1,\ldots,x_k>$构建代价函数
- 估计当前结点为根子树的可行解的上界或下界
- 极大化问题与极小化问题的区别

- 计算界的初值

得到新的更好的可行解就更新界



PART 03

最大团问题

Biggest group Algorithm

最大团问题

问题：无向图$G=<V,E>$，求G的最大团.

G的子图：$G'=<V',E'>$，其中 $V'\subseteq V, E'\subseteq E$

完全图：每对不同的顶点之间都恰连有一条边相连

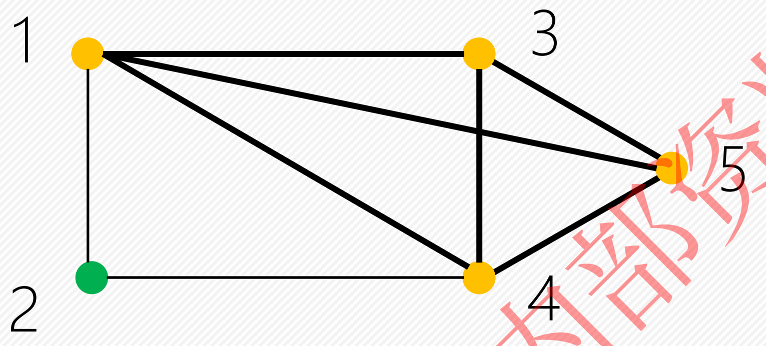
G中的团：G的完全子图 子图+完全图

G的最大团：顶点数最多的团

了解内容：

最大团与补图中的最大点独立集

编码问题中的混淆图



实例

最大团：{ 1, 3, 4, 5 },

（1）定义搜索问题的解向量和每个分量的理论取值集合

解向量为 $<x_1, x_2, \ldots, x_n>$

确定 x_i 的理论上可能取值的集合为 X_i , $i = 1, 2, \ldots, n$.

问题：给定无向图 $G=<V,E>$，其中顶点集 $V=\{1,2,\ldots,n\}$，边集为 E，求G中的最大团.

解：$<x_1, x_2, \ldots x_n>$为0-1向量

$x_k = 1$当且仅当顶点 k 属于最大团.

蛮力算法：对每个顶点子集，检查是否构成团，即其中每对顶点之间是否都有边。n个顶点，因此 有2^n个子集。至少需要指数时间.

分支限界算法的设计步骤

(1) 定义搜索问题的**解向量和每个分量的取值集合**

解向量为 $<x_1, x_2, \ldots, x_n>$

确定 x_i 的**理论上可能取值**的集合为 X_i, $i = 1, 2, \ldots, n$.

(2) 确定结点儿子的**排列规则（从大到小）** 分支规定左子树为 1，右子树为 0.

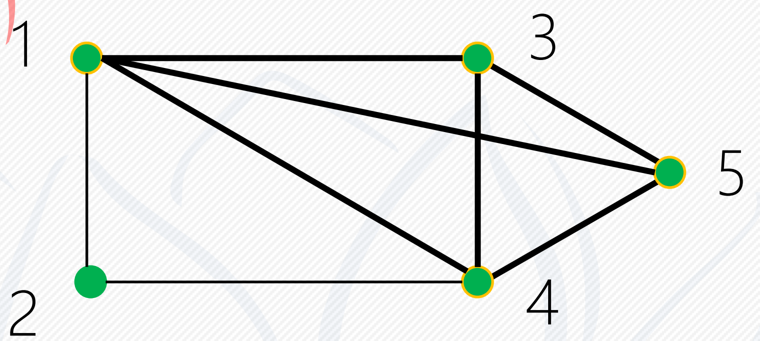
(3) 判断是否满足**多米诺性质** 若前k个点无法成团，则前k+1个点也无法成团

(4) 搜索策略-----最小耗费优先等

(5) 确定每个结点**约束条件和限界条件**

搜索树为子集树.

$x_i = 1$，表示顶点 i 在当前的团内



约束条件：该顶点与当前团内每个顶点都有边相连

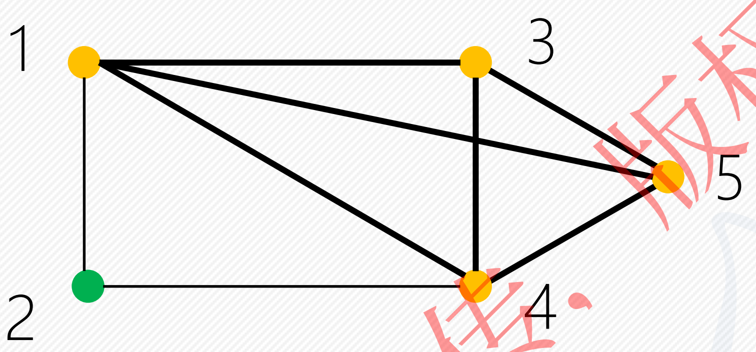
界：当前已检索到的极大团的顶点数B

代价函数：目前的团可能扩张为极大团的顶点数上界.

$$F = C_n + n - k$$

其中 C_n 为目前团的顶点数（初始为0），k 为结点层数

$$F = C_n + n - k$$



a: 当前的极大团 {1,2,4}，
顶点数为 3, 界 B=3;

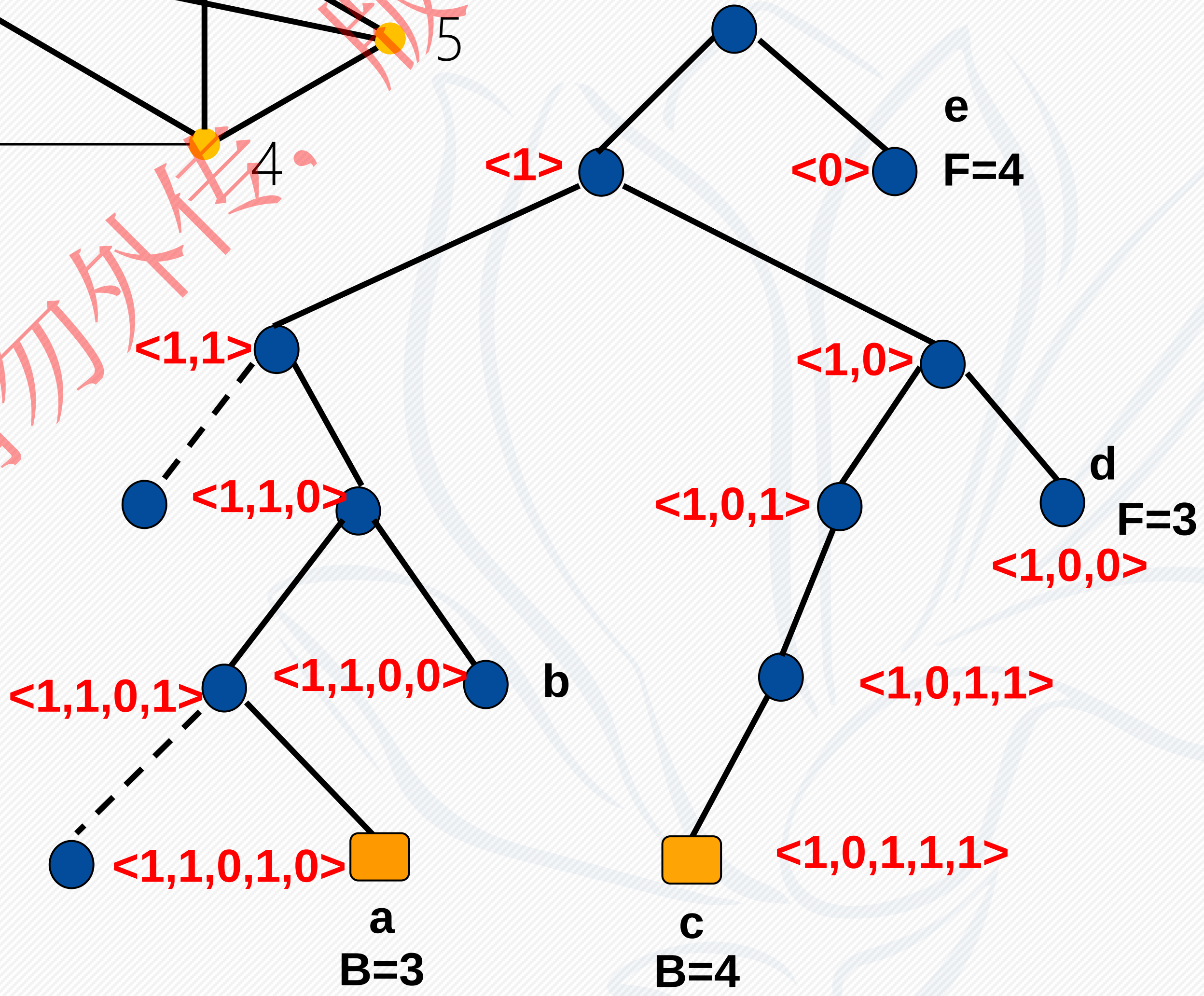
b:代价函数值 F=3, 回溯;

c: 极大团 {1,3,4,5},
顶点数为 4,
修改界 B=4;

d: F=3,不必搜索;

e: F=4, 不必搜索.

输出最大团 {1,3,4,5},顶点数为 4.



(1) 定义搜索问题的解向量和每个分量的取值集合

解向量为 $<x_1, x_2, \ldots, x_n>$

确定 x_i 的理论上可能取值的集合为 X_i, $i = 1, 2, \ldots, n$.

(2) 确定结点儿子的排列规则

(3) 判断是否满足多米诺性质

(4) 搜索策略-----最小耗费优先等

(5) 确定每个结点约束条件和限界条件（代价函数）

(6) 确定存储搜索路径的数据结构