Integer Programming - Lec1

Zhengyu Jin

Department of Computer Science and Technology
Zhejiang University

2025 年 10 月 30 日

Overview

1. **Premilinaries**
2. **Examples of Integer Programming**
3. **Methods of Calculating Integer Programming**
   3.1 Totally Unimodular Matrix
   3.2 Gomory Cutting Plane Method
   3.3 Branch and Bound Method
4. **Approximate algorithm for IP based on LP**
5. **Primal-dual approximation algorithm**

1. Premilinaries

整数规划

定义 (整数规划)

设 $A \in \mathbb{R}^{m\times n}$，$\mathbf{b} \in \mathbb{R}^m$，$\mathbf{c} \in \mathbb{R}^n$。整数规划（integer programming, IP）是指如下形式的优化问题：

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \le \mathbf{b}, \\ & \mathbf{x} \in \mathbb{Z}^n. \end{aligned}$$

换言之，我们的可行域从多面体变为了多面体中所有的整数点。

整数规划的松弛问题

线性整数规划可以"松弛"（Relax）成线性规划问题。将整数规划中的 $\mathbf{x} \in \mathbb{N}^n$ 的约束变为 $\mathbf{x} \geq \mathbf{0}$，我们称得到的 LP 问题就是松弛 ILP 后的线性规划问题。整数规划 (P) 与它的松弛问题 $(\tilde{P})$ 有下述关系：

- 若 $(\tilde{P})$ 无可行解，则 (P) 无可行解；
- 对于最大（小）化问题，$(\tilde{P})$ 的目标函数最优值给出了 (P) 目标函数最优值的一个上（下）界；
- 若 $(\tilde{P})$ 的最优解恰是整数解，则该解也是 (P) 的最优解。

近似算法基础概念

假设有某类问题 $\mathcal{I}$，其中一个具体实例为 I，且有一个复杂度为多项式的近似算法 A。定义：

- $A(I)$ 代表用算法 A 求解实例 I 得到的解；
- $OPT(I)$ 代表实例 I 的最优解。

并进一步定义：若存在实数 $r \geq 1$，对任意的 $I \in \mathcal{I}$ 都有

$$A(I) \leq r \cdot OPT(I)$$

那么称算法 A 为该类问题的 r-近似算法。我们特别关心其中可以取到的最小的 r，称

$$\rho = \inf\{r : A(I) \leq r \cdot OPT(I), \forall I \in \mathcal{I}\}$$

为近似比（approximation ratio）。它可以等价定义为

$$\rho = \sup_{I \in \mathcal{I}} \frac{A(I)}{OPT(I)}$$

近似算法基础概念

反之，如果问题是最大化问题，那么应该将上式中的分子分母交换位置，改为

$$\rho = \sup_{I \in \mathcal{I}} \frac{OPT(I)}{A(I)}$$

将两者合并，可以统一写作

$$\rho = \sup_{I \in \mathcal{I}} \left\{ \frac{OPT(I)}{A(I)}, \frac{A(I)}{OPT(I)} \right\}$$

近似算法基础概念

给定一类问题 $\mathcal{I}$ 和算法 A，我们很难根据定义求出 A 的近似比，因为 $OPT(I)$ 一般未知。因此，只能通过 $OPT(I)$ 的范围来确定近似比，以最小化问题为例，确定近似比需要下面两个条件：

- 首先寻找一个 $r > 1$，对于任何实例 I 都有 $A(I) \leq r \cdot OPT(I)$ 成立，则 r 是近似比的一个下界。（可以首先寻找到 $OPT(I)$ 的一个下界 $LB(I) \leq OPT(I)$，然后使 $A(I) \leq r \cdot LB(I)$ 即可。）
- 接下来证明 r 是不可改进的，即：对任意的 $\varepsilon \geq 0$，都存在一个实例 $I_\varepsilon \in \mathcal{I}$，使得 $A(I_\varepsilon) \geq (r - \varepsilon)OPT(I_\varepsilon)$。

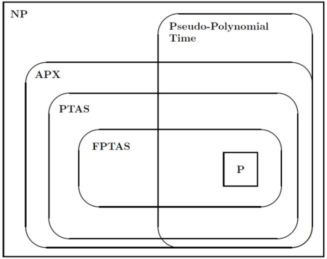
近似算法的复杂性分类

在 $P \neq NP$ 的假设下，没有多项式算法解决 NP 问题，因此近似比不可能为 1；不过，我们希望设计近似比尽可能小（尽可能接近 1）的多项式时间近似算法。那么什么样的近似是最好可能的呢？设问题类为最小化问题，$|I|$ 代表问题实例 I 的规模，f 代表某个可计算（computable）函数，不过不一定为多项式函数；那么有

- **PTAS**（多项式时间近似方案，Polynomial time approximation scheme）：存在算法 A，使得对于每一个固定的 $\varepsilon > 0$，对任意的实例 I 都有 $A(I) \leq (1+\varepsilon) \cdot OPT(I)$，且算法 A 的运行时间以问题规模 $|I|$ 的多项式为上界，则称 A 是该问题类的一个 PTAS。一般可以将 PTAS 的复杂度记为 $O\left(|I|^{f(1/\varepsilon)}\right)$。
- **EPTAS**（Efficient PTAS）：在 PTAS 的基础上，要求算法 A 的复杂度是 $O(|I|^c)$ 的，其中 $c \geq 0$ 是与 ε 无关的常数。可以将 EPTAS 的复杂度记为 $|I|^{O(1)} f(1/\varepsilon)$。
- **FPTAS**（Fully PTAS）：在 PTAS 的基础上，要求算法 A 运行时间关于 $|I|$ 和 ε 皆呈多项式，运行时间为 $|I|^{O(1)}(1/\varepsilon)^{O(1)}$。

近似复杂性类的关系

除了上面三者外，还有范围最广的 APX，代表可近似（approximable），如果某个 NP 问题存在近似比为常数的多项式时间近似算法，则称该问题属于 APX.



2. Examples of Integer Programming

分配问题（Assignment Problem）

例 (分配问题)

- n 个人要执行 n 项工作;
- 每个人恰好执行一项工作;
- 将人 i 分配给工作 j 产生代价 c_{ij};
- 求使总代价最小的分配方案。

决策变量：对于 $(i, j \in [1, n] := \{1, \ldots, n\})$,

$$x_{ij} = \begin{cases} 1 & \text{如果人}i\text{ 被分配执行工作}j, \\ 0 & \text{Otherwise} \end{cases}$$

分配问题的数学模型

约束条件：

- 每个人恰好做一项工作：

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i \in [1, n])$$

- 每项工作恰好被一个人完成：

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j \in [1, n])$$

- 变量为二元变量：

$$x_{ij} \in \{0, 1\} \quad (i, j \in [1, n])$$

目标函数：总代价 $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

分配问题的完整模型

$$
\begin{aligned}
\min_{x\in\mathbb{R}^{n\times n}} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1,\ldots,n, \\
& \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for } j = 1,\ldots,n, \\
& x_{ij} \in \{0,1\} \quad \text{for } i,j = 1,\ldots,n
\end{aligned}
$$

0-1 背包问题

例 (0-1 背包问题)

- 容量为 b 的背包需要装入 n 件物品的一个子集；
- 物品 i 的体积为 a_i，价值为 c_i；
- 选择物品使得总价值最大。

决策变量：

$$x_i = \begin{cases} 1 & \text{如果物品} i \text{ 被选择}, \\ 0 & \text{否则} \end{cases}$$

背包问题模型：

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \leq b, \end{aligned}$$

旅行商问题（TSP）

例 (旅行商问题)

- 旅行商需要访问 n 个城市各一次，然后返回起点；
- 对于每对城市 $i, j \in [1, n]$，存在从 i 到 j 的直接航线。有向图 $G = (V, E)$ 中顶点为城市，有向边为航线，假设为完全图；
- 沿边 ij 从城市 i 到城市 j 需要 c_{ij} 小时；
- 求访问城市的顺序使得总旅行时间最小。

TSP 的二元整数规划形式

决策变量：对所有 $i, j \in [1, n]$，

$$x_{ij} = \begin{cases} 1 & \text{如果路径包含弧}(i, j), \\ 0 & \text{否则} \end{cases}$$

约束：

- 旅行商从城市 i 恰好离开一次：

$$\sum_{j: j \neq i} x_{ij} = 1 \quad (i = 1, \ldots, n)$$

- 旅行商恰好到达城市 j 一次：

$$\sum_{i: i \neq j} x_{ij} = 1 \quad (j = 1, \ldots, n)$$

- 为消除子回路，引入割集约束：

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subset V, S \neq \emptyset$$

TSP 完整模型

$$
\begin{aligned}
\min_{x} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j:j\neq i} x_{ij} = 1 \quad (i \in [1,n]), \\
& \sum_{i:i\neq j} x_{ij} = 1 \quad (j \in [1,n]), \\
& \sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \quad (S \subset V, S \neq \emptyset), \\
& x \in \{0,1\}^{n\times n}
\end{aligned}
$$

不过，这里一共有 $O(2^n)$ 个约束，使用基于松弛的方法很难求解，我们甚至很难确定某个解是否可行；（理论上）该问题可以用椭球法（Ellipsoid Method）求解。

TSP 的 Miller-Tucker-Zemlin (MTZ) 约束

设 u_i 是顶点 i 的辅助变量，$(i,j) \in E$ 代表一条从顶点 i 到 j 的边，且 $(i,j) \neq (j,i)$，换言之这是一条有向边；特别地，序号 0 代表起点 s；
$x_{ij} \in \{0,1\}$ 代表 (i,j) 是否包含在最终的 Hamilton 回路中；w_{ij} 代表边 (i,j) 的权值。那么该问题可写作

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i=0}^{n} x_{ij} = 1, \quad j = 0, \ldots, n \\
& \sum_{j=0}^{n} x_{ij} = 1, \quad i = 0, \ldots, n \\
& u_i - u_j + n x_{ij} \leq n - 1, \quad 1 \leq i, j \leq n, i \neq j
\end{aligned}
$$

MTZ 约束的理论分析

其中仅有 $O(n^2)$ 个约束，前两个约束不变，第三条约束仍然是为了保证只有一个圈：

$$u_i - u_j + nx_{ij} \leq n - 1, \quad 1 \leq i, j \leq n, i \neq j \quad \text{(TSP1)}$$

注意，这条约束中没有 u_0（代表起点 s），且所有的 $u_i, i = 1, \ldots, n$ 都不会出现在目标函数或任何影响中，它们是纯粹的辅助变量。我们引理如下：

定理

所有的 *Hamilton* 回路都满足（*TSP1*），且满足（*TSP1*）的解一定是 *Hamilton* 回路。

MTZ 约束正确性的证明

证明.

首先证明满足约束（TSP1）的解都代表了哈密顿回路。用反证法，设该约束下的一个可行解不是哈密顿回路，那么在前两条约束下，该解一定是多个圈的并，那么至少有一个圈不包含点 s，设该圈为 P，其中有 $k \geq 2$ 条边，即 $|P| = k$。那么对于任意的 $(i,j) \in P$，有 $x_{ij} = 1$，于是 $u_i - u_j + n \leq n - 1$，进而有

$$\sum_{(i,j)\in P} (u_i - u_j + n) \leq \sum_{(i,j)\in P} (n-1) \Rightarrow kn \leq k(n-1)$$

矛盾！再证明所有的哈密顿回路都满足（TSP1）。假设一组解代表了一条哈密顿回路 H，即

$$x_{ij} = \begin{cases} 1, & (i,j) \in H \\ 0, & (i,j) \notin H \end{cases}$$

此时，我们只需构造出一组满足约束的 u_i 即可：因为 0 不受限制所以一定可以构造。

3. Methods of Calculating Integer Programming

3.1 Totally Unimodular Matrix

全幺模矩阵（Totally unimodular matrix）

定义 (全幺模矩阵)

设 $A \in \mathbb{Z}^{m\times n}$。若对任意由 A 的若干行和若干列选取而成的子方阵 B，均有 $\det(B) \in \{0, 1, -1\}$，则称 A 为全幺模矩阵（totally unimodular matrix）。

定理

设 $A \in \mathbb{Z}^{m\times n}$ 为全幺模矩阵，$\mathbf{b} \in \mathbb{Z}^m$。则线性规划 $\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge \mathbf{0}\}$ 的每个极点解均为整数解。

全幺模矩阵（Totally unimodular matrix）

证明.

设 $P=\{x\in\mathbb{R}^n : Ax\le b,\ x\ge 0\}$，令 x^* 为 P 的一个顶点。

由顶点的性质，存在恰好 n 个在 x^* 处紧的约束，$A'x^*=b'$，其中 A' 由 A 的某 n 行组成且线性无关，b' 为 b 的对应分量。

因为 A 是全幺模的，任意子方阵的行列式均为 0 或 ±1。这里 A' 线性无关，故 $\det(A')=\pm1$。

由 Cramer 法则，对于 $j=1,\dots,n$ 有

$$x_j^*=\frac{\det(A_j')}{\det(A')},$$

由于 A' 与 b' 都为整数矩阵/向量，A_j' 的行列式 $\det(A_j')\in\mathbb{Z}$。又 $\det(A')=\pm1$，因此 $x_j^*=\pm\det(A_j')\in\mathbb{Z}$。 □

全幺模矩阵的例子

常见的全幺模矩阵示例：

1. 有向图的点-弧关联矩阵（node-arc incidence matrix）。设有向图 $G=(V,E)$，按顶点为行、弧为列构造矩阵 A，元素

$$A_{v,e}=\begin{cases}+1, & \text{若} v \text{ 为弧} e \text{ 的起点（tail）}, \\ -1, & \text{若} v \text{ 为弧} e \text{ 的终点（head）}, \\ 0, & \text{Otherwise.}\end{cases}$$

2. 二分图的点-边关联矩阵（undirected incidence of a bipartite graph）。边 $e=\{u,v\}$ 对应的列在 u,v 行处为 1，其余为 0。
3. 单位矩阵及其与全幺模矩阵的拼接。单位矩阵 I 显然全幺模；若 A 全幺模，则拼接矩阵 $[A;-I]$ 或 $[A\ \ -I]$（把 $-I$ 作为额外行/列）仍然是全幺模。这在用点-弧矩阵表示流问题并加入上界约束时经常用到。

为什么这些矩阵是全幺模?

给出简单的证明要点（sketch）:

- 有向点-弧关联矩阵：归纳，每一列最多两个非零元，如果有一列全 0 则行列式为 0；如果有一列只有一个非零元则可变换成分块对角矩阵；如果全都是两个非零元则矩阵秩为 1.
- 二分图点-边矩阵：同有向点-弧关联矩阵。
- 拼接与子矩阵稳定性：直接对新拼接上去的单位矩阵依次展开即可。

例 1：最大流问题—矩阵形式

考虑有向网络 $G=(V,E)$，源点 s，汇点 t，变量 x_e 表示弧 e 上的流量。流守恒与源汇约束可写为

$$Ax=b, \qquad b_s=F,\ b_t=-F,\ b_v=0\ (v\neq s,t),$$

其中 A 为点-弧关联矩阵（每列一个弧，列在弧的 tail 行为 +1，head 行为 -1）。
加上容量约束 $0\le x_e\le u_e$，若所有 u_e 与 b 为整数，则任一极点解 x^* 都为整数，因 A 全幺模。

例 2：指派问题 / 二分匹配

考虑一个二分图 $G = (U \cup V, E)$，其中 $|U| = |V| = n$。目标是找到一个完美匹配，使得所选边的总权重最小。设 c_{uv} 为选择边 (u, v) 的成本。

设 $x_{uv} = 1$ 若边 (u, v) 在匹配中，否则为 0。其线性规划形式为：

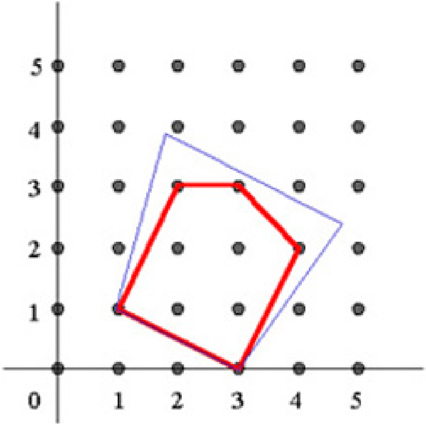
$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in E} c_{uv} x_{uv} \\
\text{s.t.} \quad & \sum_{v:(u,v)\in E} x_{uv} = 1, \quad \forall u \in U \\
& \sum_{u:(u,v)\in E} x_{uv} = 1, \quad \forall v \in V \\
& x_{uv} \geq 0
\end{aligned}
$$

约束矩阵 A 的行对应于顶点，列对应于边。每列（边）恰有两个 1（在其端点对应的行）。这是二分图的点-边关联矩阵，因此是全幺模的。所以，上述 LP 的最优解一定是整数解，直接给出了指派问题的最优解。

3.2 Gomory Cutting Plane Method

Gomory 割平面法

基本思想：首先不考虑整数的条件直接求解，若得到的不是整数解，则增加特定的约束条件（称为割平面），使得在原可行域被"切掉"包含该解的一部分，被切掉的这部分不包含任何的整数可行解。这样经过有限次的切割，最终可得到某个顶点的坐标恰好是整数，并且是问题的最优解。



Gomory 割平面法

考虑一个线性规划问题的可行域多面体 $P = \{x \in \mathbb{R}^n : Ax \le b, x \ge 0\}$，其对应的整数规划可行域为 $Q = P \cap \mathbb{Z}^n$。

给定 P 的一个非整数顶点 $\bar{x}$，一定有 $\bar{x} \notin \text{conv}(Q)$。因为 $\text{conv}(Q) \subseteq P$，由于 $\bar{x}$ 是 P 的顶点，若 $\bar{x} \in \text{conv}(Q)$ 则 $\bar{x}$ 必然也是 $\text{conv}(Q)$ 的顶点，但 $\text{conv}(Q)$ 的所有顶点都是整数向量，矛盾。

我们希望添加一个割平面 $\alpha^T x = \beta$，将这个顶点切去，但又不会影响所有整数可行解——也就是说，我们需要 Q 与 $\bar{x}$ 的一个分割超平面。

有效不等式与 Chvátal-Gomory Cuts

我们将满足 $\alpha^T x \leq \beta, \forall x \in Q$ 的不等式之为（集合 Q 的）有效不等式（Valid Inequality），下面就来讨论如何寻找有效不等式。

一种思路如下：对于任意的整数可行解 $x \in Q$，有 $Ax \leq b$ 成立，即 $\sum_{i=1}^n A_i x_i \leq b$ 成立，其中 A_i 代表矩阵 A 的第 i 列；对于任意的 $u \in \mathbb{R}^n, u \geq 0$，不等式两边同乘 u^T 有 $\sum_{i=1}^n u^T A_i x_i \leq u^T b$；由于 $x \geq 0$，

因此 $\sum_{i=1}^n \lfloor u^T A_i \rfloor x_i \leq \sum_{i=1}^n u^T A_i x_i$，于是有 $\sum_{i=1}^n \lfloor u^T A_i \rfloor x_i \leq u^T b$ 成立。注意到 $\sum_{i=1}^n \lfloor u^T A_i \rfloor x_i$ 一定是整数，因此下式成立：

$$\sum_{i=1}^n \lfloor u^T A_i \rfloor x_i \leq \lfloor u^T b \rfloor$$

该不等式便是一个有效不等式，其确定的割平面称为 Chvátal-Gomory cut。

Gomory Cut 的构造

设有 ILP 问题松弛而来的 LP 问题 $\max\{c^T x : Ax = b, x \geq 0\}$，假设使用单纯形求解后获得的最优解 x 不是整数解，那么选择一个非整数的基变量 $x_i \notin \mathbb{Z}$；在最终的单纯形表中，应有

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{b}_i \quad \text{(CP1)}$$

其中 x_i 是基变量，$x_j, j = m+1, \ldots, n$ 是所有的非基变量。既然 x_i 不是整数，说明 $\bar{b}_i$ 一定不是整数。当然 $\bar{a}_{ij}$ 也可能不是整数。取 $u = 1$，对 (CP1) 应用 Chvátal-Gomory cuts，可以得到

$$x_i + \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor \quad \text{(CP2)}$$

式 (CP2) 便是一个有效不等式；式 (CP1) 的整数解一定符合式 (CP2)，而用单纯形法求解 (CP1) 得到的非整数解就不符合式 (CP2) 了。我们只要把式 (CP2) 再加入原来的线性规划问题，作为新的限制，直到找到整数解。

Summary

大部分参考资料不会直接加入式 (CP2)，而是加入 (CP1) - (CP2)，即

$$\sum_{j\in N}(a_{ij} - \lfloor a_{ij}\rfloor)x_j \geq b_i - \lfloor b_i\rfloor \quad \text{(CP3)}$$

效果是一样的。上式被称作 Gomory cut，由于其中的所有系数都是 $[0,1)$ 之间的分数，因此也被称作 Gomory fractional cut。

Gomory 割平面法的步骤如下：

1. 将 ILP 问题的整数约束松弛为线性不等式约束，得到一个 LP 问题；
2. 使用单纯形法（或对偶单纯形法）求解 LP 问题，得到最优解 x^*（若 LP 无最优解，则原 ILP 也无最优解，算法终止）；
3. 若 x^* 是整数解，结束算法；否则存在某个 $x_i^* \notin \mathbb{Z}$；
4. 在 LP 问题中加入约束 (CP2)（或者 (CP3)），得到一个新的 LP 问题，回到第 2 步。

3.3 Branch and Bound Method

分支定界法

分支定界法（branch and bound）由 A. Land 和 G. Doig 在上世纪 60 年代被提出。其思想和最优性剪枝或者 min-max 搜索树类似。

我们先将原 ILP 问题松弛成 LP 问题求解，并假设最优解中 $N < x_i < N+1$ 不是整数，那么有两种情况：$x_i \le N$ 或 $x_i \ge N+1$，我们对这两种情况分别进行搜索。换言之，分别将这两个约束加入 LP 问题中，会得到两个新的子 LP 问题（" 分支"），分别求解之，并递归地应用上面的步骤。我们可以将这视作构建了一个搜索树。

设 ILP 为最大化问题。由于我们一开始将整数的约束进行了松弛，因此 LP 问题求解得到的最优值便是 ILP 最优值的一个上界。但分支之后，由于添加了新的约束，因此子 LP 问题最优值必然不会优于原来的（父节点的）LP 问题，分支层数越来越多，这个上界也越来越低。

分支定界法

通过" 分支"，我们将全部可行解空间划分成了越来越小的（互斥的）子集。为每个子集计算目标函数的上下界的过程，就是所谓" 定界"：

- 在任一枝内，求解 LP 得到的最优目标函数值，构成了该枝内整数规划问题目标函数值的上界；
- 如果在某一枝内求解 LP 得到了一个整数解，那么该解便对应原整数规划最优解的下界。

在分支之后，我们根据子问题计算结果来决定是否要做进一步的分支：若某一个子 LP 问题的最优值还没有当前下界更优，那么这一枝就可以直接不作考虑（因为 LP 问题的解是该枝能找到的最优解的上界），即" 剪枝（pruning）"。我们不断地降低每一枝的上界、提升全局的下界；直至每一枝都已探明，此时搜索树叶子节点的 LP 最优值均不大于当前 ILP 下界，算法结束。

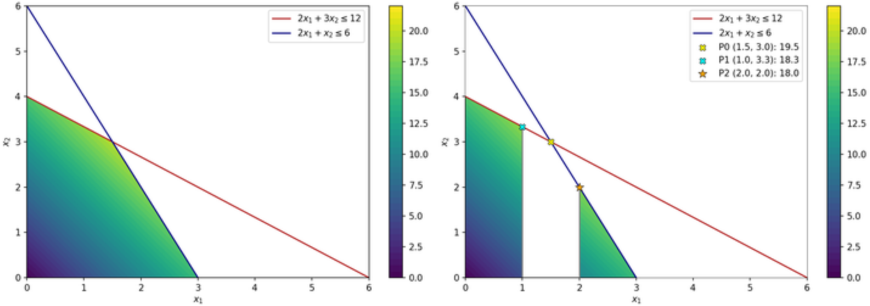
分支定界法例子

例：考虑下面的 ILP 问题，我们记作 (P)，

$$
\begin{aligned}
(P) \quad \max \quad & 5x_1 + 4x_2 \\
\text{s.t.} \quad & 2x_1 + 3x_2 \le 12 \\
& 2x_1 + x_2 \le 6 \\
& x_1, x_2 \in \mathbb{N}
\end{aligned}
$$

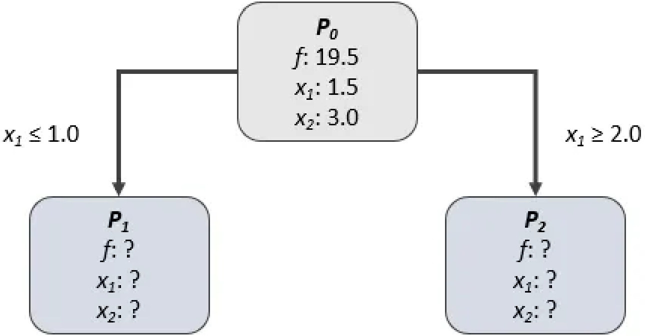
松弛得到 LP 问题，再加入松弛变量 x_3, x_4 将不等约束改为等号。记该问题为 (P_0)，用单纯形法求解得到最优解为 $(x_1, x_2) = \left(\frac{3}{2}, 3\right)$，目标函数最优值为 $39/2 = 19.5$。接下来按照"$x_1 \le 1$" 和"$x_1 \ge 2$" 进行分支，记子 LP 问题分别为 (P_1) 和 (P_2)。如下所示，类似于割平面法，分支的过程事实上也" 切除" 了可行域内解全为分数的一部分。

分支定界法例子（续）

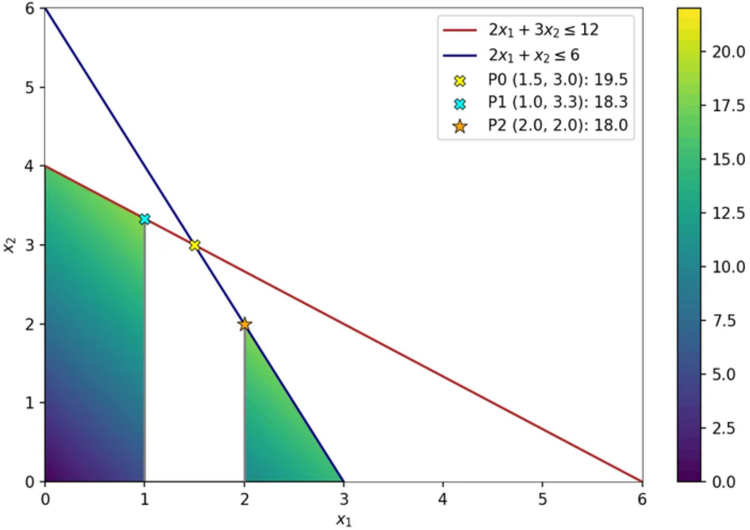


$2x_1 + 3x_2 \leq 12$
$2x_1 + x_2 \leq 6$
P0 (1.5, 3.0): 19.5
P1 (1.0, 3.3): 18.3
P2 (2.0, 2.0): 18.0
x_1
x_2

分支定界法例子（续）

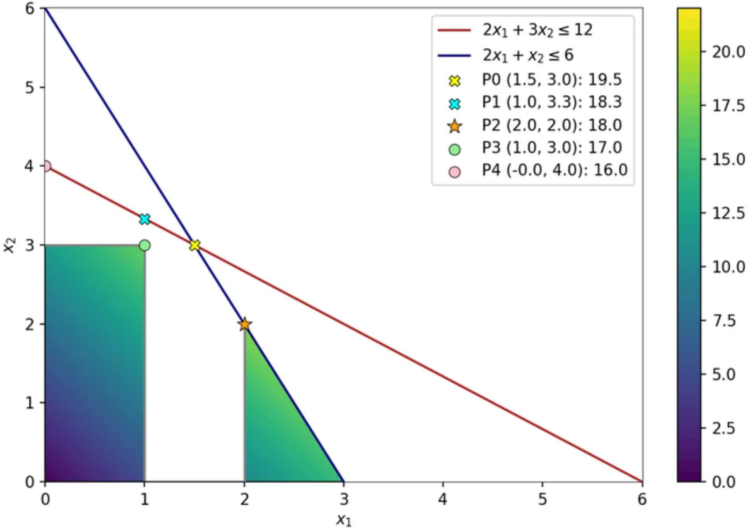


分支定界法例子（续）



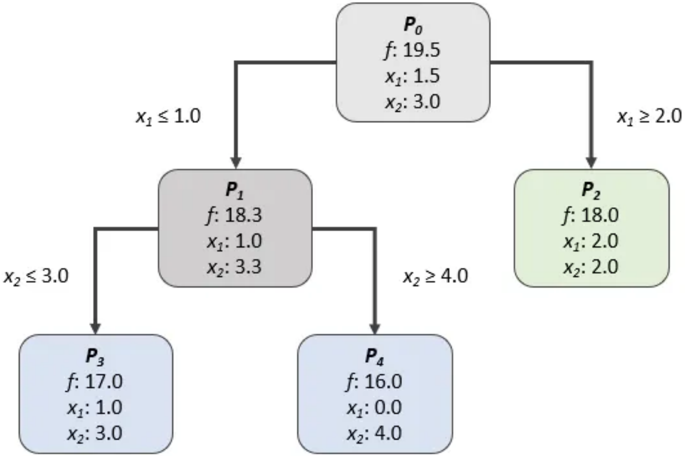
$2x_1 + 3x_2 \leq 12$
$2x_1 + x_2 \leq 6$
P0 (1.5, 3.0): 19.5
P1 (1.0, 3.3): 18.3
P2 (2.0, 2.0): 18.0
x_1
x_2
0
1
2
3
4
5
6
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

分支定界法例子（续）



$2x_1 + 3x_2 \leq 12$
$2x_1 + x_2 \leq 6$
P0 (1.5, 3.0): 19.5
P1 (1.0, 3.3): 18.3
P2 (2.0, 2.0): 18.0
P3 (1.0, 3.0): 17.0
P4 (-0.0, 4.0): 16.0
x_1
x_2
0 1 2 3 4 5 6
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0

分支定界法例子（续）



P_0
f: 19.5
x_1: 1.5
x_2: 3.0
$x_1 \leq 1.0$
$x_1 \geq 2.0$
P_1
f: 18.3
x_1: 1.0
x_2: 3.3
P_2
f: 18.0
x_1: 2.0
x_2: 2.0
$x_2 \leq 3.0$
$x_2 \geq 4.0$
P_3
f: 17.0
x_1: 1.0
x_2: 3.0
P_4
f: 16.0
x_1: 0.0
x_2: 4.0

4. Approximate algorithm for IP based on LP

基于 LP 的近似算法

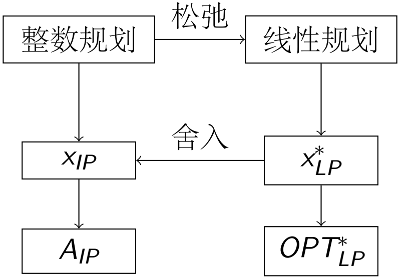
在求解某个（极小化）ILP 问题时，

1. 我们先进行条件的松弛，将 $x \in \mathbb{N}^n$ 的条件松弛为 $x > 0$，这样就将 ILP 问题转换为了 LP 问题；
2. 而这个 LP 问题可以用单纯形或者内点法等解决，得到其解 x^*_{LP}。
3. 我们将 x^*_{LP} 进行舍入（rounding），就得到了一个整数解 x_{IP}。可以认为 x_{IP} 就是对 ILP 最优解的一个近似。

这种利用"Relax → Solve LP → Rounding" 的方法就成为基于 LP 的近似算法（LP-based approximation algorithm）。更一般地说，将复杂问题实例进行简化，求解简化后的问题实例作为原实例的近似解，这是构造近似算法的一个标准方法。

近似比分析

记 LP 的最优目标函数值为 OPT_{LP}，该近似算法的目标函数值为 A_{IP}，如下。



记 x^*_{IP} 和 OPT_{IP} 分别是整数规划的（未知的）最优解和最优目标函数值。简单分析可以发现有 $OPT_{LP} \leq OPT_{IP} \leq A_{IP}$ 成立，且近似比有上界

$$\rho \leq \frac{A_{IP}}{OPT_{IP}} \leq \frac{A_{IP}}{OPT_{LP}}$$

舍入比率与整数间隙

A_{IP} 与 OPT_{LP} 的不同是舍入造成的，于是这个上界又称" 舍入比率（Rounding Ratio, RR）"；我们可以用舍入比率 A_{IP}/OPT_{LP} 来估算近似比 ρ。不过，若 OPT_{IP} 和 OPT_{LP} 相差很大，那么我们分析得到的上界也不会很好。注意到，整数规划与线性规划最优值之比是舍入比率的一个下界

$$\frac{A_{IP}}{OPT_{LP}} \geq \boxed{\frac{OPT_{IP}}{OPT_{LP}}}$$

我们比较关心右侧项的最大值，即 $\sup_{I\in\mathcal{I}} \frac{OPT_{IP}(I)}{OPT_{LP}(I)}$，称这个值为" 整数间隙（Integrality Gap, IG）"。这是一个不低于 1 的值，我们分析得到的近似比上界不会小于整数间隙；要想得到好的估计结果，那么整数间隙就不能太大。换言之，整数间隙限制了算法的近似能力，因为近似比无法保证比整数间隙更好。大多数时候，近似比都恰好等于整数间隙。

对于某些问题，x^*_{LP} 同时也是整数解，那么 $x^*_{IP} = x^*_{LP}$，此时整数间隙正好为 1。这样的松弛也称" 精确松弛（exact relaxation）"。

0-1 背包问题的 LP 松弛近似算法

我们来分析 0-1 背包问题的 LP 松弛近似算法性能。回顾 0-1 背包问题：

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} c_i x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \leq b, \\
& x_i \in \{0,1\}, \quad i = 1, \ldots, n
\end{aligned}$$

LP 松弛问题为：将约束 $x_i \in \{0,1\}$ 松弛为 $0 \leq x_i \leq 1$。

贪心算法思想：不妨设 $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$，那么线性规划问题的一个最优解是

$$(1,1,\ldots,1,\alpha,0,0,\ldots,0)$$

其中 α 前面的 1 有 k 个，$0 < \alpha \leq 1$，满足 $\sum_{i=1}^{k} a_i + \alpha a_{k+1} = b$。

背包问题近似算法的分析

设前 k 个物品的总价值为 V_1，第 $k+1$ 个物品的价值为 V_2。我们可以分别将" 前 k 个物品" 和" 第 $k+1$ 个物品" 视作是整数规划的两个可行解，自然有

$$OPT_{LP} = V_1 + \alpha V_2 \le V_1 + V_2 \le OPT_{IP} + OPT_{IP} = 2OPT_{IP}$$

于是 $OPT_{LP}/OPT_{IP} \le 2$，即整数间隙不超过 2。另一方面，这个 2 是紧的，考虑例子

$$\left\{\left(\frac{K}{2}, V\right), \left(\frac{K+1}{2}, V\right)\right\}$$

其中 K 是背包容量，则 $OPT_{LP} = V + \frac{K}{K+1}V$，而 $OPT_{IP} = V$，显然二者之比为 $\frac{2K+1}{K+1}$，当 K 充分大即可得到 $OPT_{LP}/OPT_{IP} \to 2$。

综上，使用基于 LP 的近似算法求解 0-1 背包问题，其整数间隙是 2。

顶点覆盖问题

定义 (顶点覆盖)

回忆一下，顶点覆盖（vertex cover）是指一个简单无向图 $G=(V,E)$ 一个顶点子集 $V' \subseteq V$，使得任意一条 E 中的边都至少有一个端点在 V' 中，而顶点覆盖问题是求一个包含顶点最少的顶点覆盖。

我们将顶点覆盖问题形式化为一个整数规划：

$$
\begin{aligned}
\min \quad & \sum_{v \in V} x_v \\
\text{s.t.} \quad & x_u + x_v \geq 1 \quad \forall (u,v) \in E \\
& x_v \in \{0,1\} \quad \forall v \in V
\end{aligned}
$$

其中 $x_v = 1$ 表示顶点 v 被选入覆盖集合，$x_v = 0$ 表示不被选入。

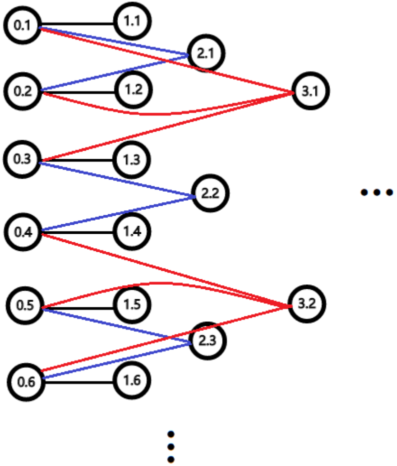
顶点覆盖的贪心算法

关于无权顶点覆盖问题，有一个直观的思路：

思路 **1**：对于无权顶点覆盖问题，算法将度数最大的点 u 选入答案集合，并将 u 与端点包含 u 的边都删去。重复这个过程，直到所有边都被删去为止。

这是一个思路非常自然的贪心算法，但其近似比非常差。考虑这样一个图结构：该图的顶点共有 $k+1$ 列，第 1 列共有 $k!$ 个点。第 $i+1$ 列有 $k!/i$ 个点，$i=1,2,\dots,k$，且这些点均与第 1 列中 i 个顶点相连。见下图：

顶点覆盖的贪心算法



顶点覆盖的贪心算法

那么这张图一共有 $k! + k!\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}\right) \approx k! + k! \log k$ 个顶点。

考虑这张图上的顶点覆盖问题。显然，只要将第一列的 $k!$ 个点加入答案集合中，就能获得最小的顶点覆盖。显然第一列的 $k!$ 个点度数均为 k，可是最后一列的点度数也为 k。若算法选择了最后一列的所有点，则第一列的点度数就都会减小至 $k-1$。由于倒数第二列的点度数也为 $k-1$，算法依旧可以选择倒数第二列的所有点；以此类推，我们会把从第二列到最后一列的所有点都选入答案集合，才能获得一个顶点覆盖。所以，该近似算法的近似比至少为 $\frac{k! \log k}{k!} = \log k$，是一个比较差的算法。

顶点覆盖的其他算法思路

思路 **2**：对于无权顶点覆盖问题，随机选择图中的一条边 (u, v)，将 u 与 v 都加入解的集合，并删去 u、v 以及所有端点包含 u 或 v 的边。重复这个过程，直到所有边都被删去为止。

这是一个听起来很不自然的算法，然而它的近似比为 2：假设该算法选中了 k 条边，那么这 k 条边是原图中的一个匹配（因为这 k 条边没有相同的端点）。为了覆盖这 k 条边形成的匹配，每条边至少要有一个端点被选中。也就是说，最优解至少为 k 个顶点，而该算法选择了 $2k$ 个，那么近似比至多为 2，容易证明这个界是紧的。

加权顶点覆盖问题

对于有权图顶点覆盖的情况，使用基于 LP 的近似算法。
考虑顶点赋权的情况，设简单无向图 $G=(V,E)$ 的顶点数目为 $|V|=n$。用 x_i 表示第 i 个点是否在答案集合中，w_i 表示第 i 个点的权重。则该问题可以写作

$$\begin{aligned} \min \quad & \sum_{i\in n} w_i x_i \\ \text{s.t.} \quad & x_i + x_j \geq 1, \quad \forall (i,j) \in E \\ & x_i \in \{0,1\} \end{aligned}$$

其中的约束代表每条边至少有一个顶点被选中。对该问题进行 LP 松弛，将 $x_i \in \{0,1\}$ 改为 $x_i \geq 0$（为什么没有 $x_i \leq 1$ 呢？因为 $x_i > 1$ 时一定不是最优解，所以这个约束可以省略）。

可以证明：上述线性规划的基本可行解是" 半整（half-integral）" 的，即该问题的解 x 中有 $x_i \in \{0, \frac{1}{2}, 1\}$。

半整数性质的证明

证明.

用反证法。设有一个基本可行解不是半整的，并记

$$V_+ = \left\{ i : \frac{1}{2} < x_i < 1 \right\}, \quad V_- = \left\{ i : 0 < x_i < \frac{1}{2} \right\}$$

则 $V_+ \cup V_-$ 非空，我们构造两个可行解 y 和 z:

$$y_i = \begin{cases} x_i + \varepsilon & \text{if } i \in V_+ \\ x_i - \varepsilon & \text{if } i \in V_- \\ x_i & \text{otherwise} \end{cases} \qquad z_i = \begin{cases} x_i - \varepsilon & \text{if } i \in V_+ \\ x_i + \varepsilon & \text{if } i \in V_- \\ x_i & \text{otherwise} \end{cases}$$

其中 ε 足够小。那么 $x = \frac{y+z}{2}$ 可被表示为两个可行解的凸组合，必不是极点，矛盾！ $\square$

加权顶点覆盖的 2-近似算法

接下来，我们求解松弛的 LP 问题，并按照如下方式对 LP 的最优解 x_{LP}^* 做舍入：

$$(x_{IP})_i = \begin{cases} 1, & (x_{LP}^*)_i \geq 0.5 \\ 0, & (x_{LP}^*)_i < 0.5 \end{cases}$$

由于每条边 (i,j) 存在 $(x_{LP}^*)_i + (x_{LP}^*)_j \geq 1$ 的限制，则 $\max\{(x_{LP}^*)_i, (x_{LP}^*)_j\} \geq 0.5$，所以 $(x_{IP})_i + (x_{IP})_j \geq 1$ 仍然成立，我们构造的解是可行的。容易看出，$x_{IP} \leq 2x_{LP}^*$。于是有

$$\sum_{i=1}^{n} w_i (x_{IP})_i \leq 2 \sum_{i=1}^{n} w_i (x_{LP}^*)_i$$

$$\implies OPT_{IP} \leq A_{IP} \leq 2OPT_{LP}$$

这就证明了算法的整数间隙不超过 2。这个界是紧的，设所有权重都为 1，那么对完全图 K_n 而言 $C_{LP}^* = \frac{n}{2}$（所有的点都取 1/2），于是 $A_{IP} = n$，而显然 $OPT_{IP} = n - 1$。于是 $OPT_{IP}/OPT_{LP} = 2\left(1 - \frac{1}{n}\right) \to 2$。综上：（有权图）顶点覆盖问题的整数间隙为 2。同时，选择 x_{LP}^* 中非零变量对应顶点作为近似算法的解，构成了一个 2-近似算法。

5. Primal-dual approximation algorithm

考虑线性规划问题及其对偶

(P)

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \dots, m \\
& x_j \geq 0, \quad j = 1, \dots, n
\end{aligned}
$$

(D)

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m} b_i y_i \\
\text{s.t.} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j = 1, \dots, n \\
& y_i \geq 0, \quad i = 1, \dots, m
\end{aligned}
$$

其互补松驰条件为

$$
\begin{cases}
1. \forall 1 \leq j \leq n, \text{ 有} x_j = 0 \text{ 或} \displaystyle\sum_{i=1}^{m} a_{ij} y_i = c_j \\
2. \forall 1 \leq i \leq m, \text{ 有} y_i = 0 \text{ 或} \displaystyle\sum_{i=1}^{n} a_{ij} x_j = b_i
\end{cases}
$$

放松的互补松驰条件

如果 (P) 问题是某个 ILP 问题的松弛，那么对于 ILP 问题和 (D) 而言，互补松驰条件一般不再满足。于是，我们将互补松驰条件再进行放松：

$$\begin{cases} 1.\ \text{存在}\alpha \geq 1,\ \text{对}\forall 1 \leq j \leq n,\ \text{有} x_j = 0 \text{ 或} \dfrac{c_j}{\alpha} \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j \\ 2.\ \text{存在}\beta \geq 1,\ \text{对}\forall 1 \leq i \leq m,\ \text{有} y_i = 0 \text{ 或} b_i \leq \sum_{i=1}^{n} a_{ij} x_j \leq \beta b_i \end{cases}$$

这被称作" 放松的互补松驰条件（relaxed complementary slackness）"。

定理

若原始可行解 $\boldsymbol{x}$ 和对偶可行解 $\boldsymbol{y}$ 满足上述" 放松的互补松驰条件"，那么 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 分别是原始和对偶问题的 $\alpha\beta$-近似解。

放松的互补松驰条件的证明

证明.

设 OPT_D 和 OPT_P 分别代表 (P) 和 (D) 的目标函数最优值，则

$$\sum_{j=1}^{n} c_j x_j \le \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i$$
$$\le \alpha\beta \sum_{i=1}^{m} b_i y_i \le \alpha\beta \cdot OPT_D \le \alpha\beta \cdot OPT_P$$

其中最后一步使用到了弱对偶定理。这说明 $\boldsymbol{x}$ 是 (P) 的 $\alpha\beta$ 近似解，$\boldsymbol{y}$ 同理。 □

于是，对于某个 ILP 和其松弛得到的 (P)，如果存在某个算法能找到整数原始可行解 $\boldsymbol{x}$ 和对偶可行解 $\boldsymbol{y}$ 满足上述" 放松的互补松驰条件"，则该算法就是一个 $\alpha\beta$-近似算法。

集合覆盖问题

回忆一下，（加权的）集合覆盖问题是指：设 E 是有限的基本元素集，$\mathcal{U} \subseteq 2^E$。设有一个费用（或者说权重）函数 $c : \mathcal{U} \to \mathbb{R}^+$。不妨设 $\mathcal{U} = \{U_1, U_2, \ldots, U_k\}$，那么集合覆盖问题的目标是：寻找 $\mathcal{U}$ 中费用最小的集合，使得 E 中所有元素均在其中，即寻找一个

$$\{U_{i_1}, U_{i_2}, \ldots, U_{i_l}\} \subseteq \mathcal{U},\ \text{使得} \bigcup_{p=1}^{l} U_{i_p} = E,\ \text{且} \sum_{p=1}^{l} c(U_{i_p})\ \text{最小}$$

记 $x_U \in \{0, 1\}$ 代表是否选择集合 U，那么加权集合覆盖问题可以写作

$$\begin{aligned}
\min \quad & \sum_{U \in \mathcal{U}} c(U) x_U \\
\text{s.t.} \quad & \sum_{U \supseteq e} x_U \geq 1, \quad \forall e \in E \\
& x_U \in \{0, 1\}, \quad \forall U \in \mathcal{U}
\end{aligned}$$

集合覆盖问题的 LP 松弛及对偶

显见最优解中 x_U 不会大于 1，因此我们可以把 $x_U \in \{0,1\}$ 松弛成 $x_U \geq 0$，随之写出对偶：

$$
\begin{aligned}
\max \quad & \sum_{e\in E} y_e \\
\text{s.t.} \quad & \sum_{e\in U} y_e \leq c(U), \quad \forall U \in \mathcal{U} \\
& y_e \geq 0, \quad \forall e \in E
\end{aligned}
$$

令 f 是 $\forall e \in E$ 在 $\mathcal{U}$ 中出现的最高频次，即

$$
f = \max_{e\in E} \sum_{\substack{U\in\mathcal{U} \\ U\ni e}} 1
$$

原始对偶算法的互补松弛条件

并定义下面的互补松弛条件：

$$\begin{cases} \text{对偶问题条件: } \forall U \in \mathcal{U},\ x_U \neq 0 \Rightarrow \sum_{e \in U} y_e = c(U) \\ \text{原始问题条件: } \forall e \in E,\ y_e \neq 0 \Rightarrow \sum_{U \supseteq e} x_U \leq f \end{cases}$$

换言之，在前文的”放松的互补松弛条件”中令 $\alpha = 1$、$\beta = f$。若该条件成立，那么我们得到的就是一个 f-近似算法。接下来，通过如下步骤使得该条件成立：

原始对偶算法步骤

1. 初始步：令 $\boldsymbol{y} = \mathbf{0}$，$\boldsymbol{x} = \mathbf{0}$，集合 $F = \emptyset$ 表示被覆盖的元素；
2. 迭代步：重复下述两个步骤，直至所有元素被覆盖（即 $F = E$）：
   1. 选择未被覆盖的 $e \in E \setminus F$，提升 y_e，直至某个（或某些）约束 $\sum_{e \in U} y_e = c(U)$ 变紧；
   2. 对所有约束为紧的集合 U，令 $x_U = 1$，并将这些集合中所有元素并入 F。

算法正确性分析

上面每一次迭代中至少有一个集合被选中，因此一定在多项式步骤内可结束。由于已经被覆盖的元素 $e \in F$ 所对应之 y_e 都不会再被提升，因此所有取紧约束都不会再发生变化，保证对偶可行，即：若 $x_U = 1$ 则 $\sum_{e \in U} y_e = c(U)$，对偶问题条件满足。

另一方面，由于至多有 f 个包含 e 的集合被选择，于是 $\sum_{U \supseteq e} x_U \leq f$ 自然成立，原始问题条件满足。按照这种方法，我们就得到了原 ILP 问题的一个可行解，且近似比至多为 f。

近似比紧性分析

接下来构造一个例子证明 f 的上界是紧的。如下：

$$\begin{cases} E = \{e_1, \ldots, e_{n+1}\} \\ U = \{U_1, U_2, \ldots, U_n\} = \{\{e_1, e_n\}, \{e_2, e_n\}, \ldots, \{e_{n-1}, e_n\}, E\} \\ c(U_1) = c(U_2) = \cdots = c(U_{n-1}) = 1,\ c(U_n) = 1 + \varepsilon,\ \text{其中}\varepsilon > 0 \end{cases}$$

对这个实例使用上述算法，那么算法可以让 $y_1, y_2, \ldots, y_n$ 都会被提升至 1，使得 $\{e_1, e_2, \ldots, e_n\}$ 被覆盖，但还有一个 e_{n+1} 未被覆盖，故 y_{n+1} 需要被提升至 ε；因此该算法选中了所有的集合，总费用为 $(n-1) + (1+\varepsilon) = n + \varepsilon$，而其最优解显然是仅选择 $U_n = E$，对应最优的费用为 $1 + \varepsilon$，于是近似比是 $\frac{n+\varepsilon}{1+\varepsilon} \to n = f$。

无容量限制设施选址问题

简而言之，无容量限制设施选址（Uncapacitated Facility Location, UFL）问题是：从没有限定容量大小的设施位置集合中选择要开放的设施，使其以最小的代价服务于给定的所有客户。

该问题的输入为：

- F 是候选的设施点（Facility）集合；
- 对任意的 $i \in F$ 都对应一个建设费用（Facility costs）$f_i \in \mathbb{R}_+$；
- D 是用户（clients）集合；
- $c_{ij} \in \mathbb{R}_+$ 是 $j \in D$ 到设施 $i \in F$ 的连接费用（路费）或称距离，由度量距离函数（Metric distance function）$c : (F \cup D) \times (F \cup D) \to \mathbb{R}_+$ 确定。

度量距离函数的三角不等式

我们来解释一下最后 c_{ij} 的定义。这个意思是说：假设构造一个无向完全二分图，其中每个用户 D 和每个设施 F 都是该图中的顶点，同时连接 $j \in D$ 到设施 $i \in F$ 的边被赋予正权值 c_{ij}；我们要求该图满足度量要求，即所谓的三角不等式：在下图中的情形里，所有蓝色边对应费用之和不应小于红色边。

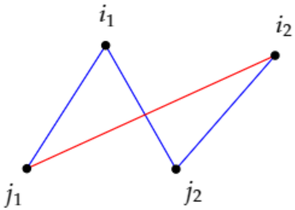


图: 示意图，用 mathcha 绘制

该性质会在最终算法分析时起到作用。

UFL 问题的决策变量

我们将" 用户 j 选择设施 i" 简写作 $\phi(j) = i$，ϕ 代表确定分配方案的函数；引入变量

$$y_i = \begin{cases} 1, & \text{若设施} i \text{ 开放} \\ 0, & \text{otherwise} \end{cases} \quad i \in F, \qquad x_{ij} = \begin{cases} 1, & \phi(j) = i \\ 0, & \text{otherwise} \end{cases} \quad i \in F, j \in D$$

这些决策变量可以松弛成 $y_i \geq 0$、$x_{ij} \geq 0$（最优解中二者都不会超过 1）。那么松弛 UFL 得到的 LP 问题为：

$$\begin{aligned} & \min && \sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij} && \\ (P) \quad & \text{s.t.} && \sum_{i \in F} x_{ij} = 1, && \forall j \in D \\ & && y_i - x_{ij} \geq 0, && \forall i \in F, j \in D \\ & && x_{ij}, y_i \geq 0, && \forall i \in F, j \in D \end{aligned}$$

UFL 问题的 LP 形式解释

其中，目标函数是两项之和，分别对应两种费用；第一个约束代表每位用户都有且仅有一个设施选择，第二个约束表示用户选择的设施必定是开放的。其对偶为

$$
(D) \quad \begin{aligned}
\max \quad & \sum_{j \in D} v_j \\
\text{s.t.} \quad & v_j - w_{ij} \leq c_{ij}, \quad i \in F, j \in D \\
& \sum_{j \in D} w_{ij} \leq f_i, \quad i \in F \\
& w_{ij} \geq 0
\end{aligned}
$$

UFL 问题的 LP 形式解释

某种程度上可以这样考虑对偶问题：F 是待建设施的选址点，让每个用户都当做投资者，自己出资建设设施；那么用户 $j \in D$ 的开销 v_j 包含他去往设施 $i \in F$ 的路费 c_{ij}、以及他个人为设施 i 投入的资金 w_{ij};

(D) 的第一条约束是说：用户 j 的开销不超过建设投资 w_{ij} 与路费 c_{ij} 之和；

(D) 的第二条约束是说所有来到第 i 个选址点的用户投入的价格之和不超过该设施建设费用 f_i;

在这些约束下（可以视作保障了用户本身利益）我们希望最大化所有人的开销。

UFL 问题的互补松弛条件

互补松弛条件为：

$$\begin{cases} 1.\, \forall i \in F, j \in D,\ x_{ij} > 0 \Rightarrow v_i - w_{ij} = c_{ij} \\ 2.\, \forall i \in F,\ y_i > 0 \Rightarrow \sum\limits_{j \in D} w_{ij} = f_i \\ 3.\, \forall i \in F, j \in D,\ w_{ij} > 0 \Rightarrow y_i = x_{ij} \end{cases}$$

互补松弛条件也有很好的解释：

1. 如果用户 j 选择去往第 i 个设施（$x_{ij} > 0$），那么他的路费一定要留够；
2. 如果第 i 个设施开放（$y_i > 0$），那么所有来到该地址的用户之投资一定是凑够 f_i 的；
3. 若用户 j 没有被设施 i 服务，那么他就不会往设施 i 投入资金，即 $y_i \neq x_{ij} \Rightarrow w_{ij} = 0$。

1. 算法一

考虑 (P) 和 (D)，求解二者得到最优解 $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ 和 $(\boldsymbol{v}^*, \boldsymbol{w}^*)$，我们设计如下算法：

1. 在所有未与设施连接（unconnected）的客户中，选择 $j \in D$ 使得 v_j^* 最小；
    1. 令 $N_j = \{i : x_{ij}^* > 0\}$，选择 N_j 中建设费用最少的设施开放；
    2. 对于任意未连接的 $j' \in D$，如果 $N_j \cap N_{j'} \neq \emptyset$，将用户 j' 连接到设施 i；
2. 重复上面的步骤，直至所有客户都与某个设施连接。

设 OPT_P 和 OPT_D 是 (P) 和 (D) 的最优函数值，有如下两个结论：

结论 1：设施开放费用部分不超过 OPT_P

定理 (结论 1)

该算法设施开放费用部分的花费不超过 OPT_P。

证明.

设 j 是算法第 1 步选出的用户，令 $f_{\min}$ 是 N_j 中建设费用最小的设施所对应费用。那么

$$f_{\min} = f_{\min}\sum_{i\in N_j} x_{ij} \le f_{\min}\sum_{i\in N_j} y_i = \sum_{i\in N_j} f_{\min} y_i \le \sum_{i\in N_j} f_i y_i$$

注意到，对于算法第 1 步选出的所有 j，N_j 都两两不交，因此设施建设部分一定不超过 $\sum_{i\in F} f_i y_i \le OPT_P$。 □

结论 2：连接费用部分不超过 $3OPT_D$

定理 (结论 2)

该算法连接费用部分的花费不超过 $3OPT_D$。

证明.

考虑算法第 1 步选出的 j，令 $i \in N_j$ 是 N_j 中费用最少的设施；记 $i' \in N_j$ 是任意的其它设施，j' 是某个其它用户满足 $x_{i'j'} > 0$。由于 $x_{ij} > 0$、$x_{i'j} > 0$、$x_{i'j'} > 0$，因此根据互补松弛条件的第 1 条，

$$
\begin{aligned}
v_j^* &= c_{ij} + w_{ij}^* \geq c_{ij} \\
v_j^* &= c_{i'j} + w_{i'j} \geq c_{i'j} \\
v_{j'}^* &= c_{i'j'} + w_{i'j'} \geq c_{i'j'}
\end{aligned}
$$

□

结论 2 的证明（续）

证明（续）.

由于算法选择的是未连接客户里 v_j^* 最小的 j，因此 $v_j^* \le v_{j'}^*$；再根据度量性质，

$$c_{ij'} \le c_{ij} + c_{i'j} + c_{i'j'} \le v_j^* + v_j^* + v_{j'}^* \le 3v_{j'}^*$$

那么总共的连接费用至多为 $\sum_{j \in D} 3v_j^* = 3OPT_D$。 □

根据强对偶定理，$OPT_P = OPT_D$，于是

$$A_{IP} \le OPT_P + 3OPT_D = 4OPT_P \le 4OPT_{IP}$$

该算法是一个 4-近似算法。

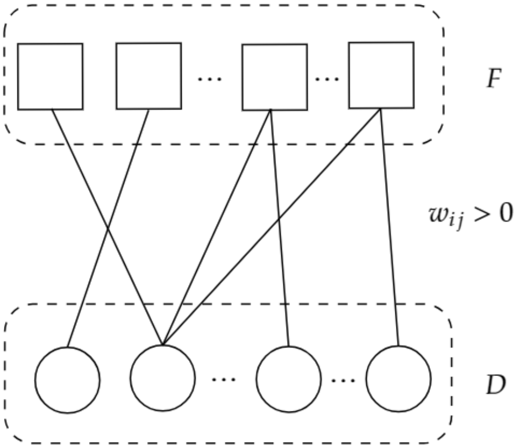
2. 算法二——第一阶段

1. 初始化，令 $v_j = 0$、$w_{ij} = 0$，此时所有用户都未连接到设施；
2. 等速提升所有 v_j；如果出现某个（或某些）$v_j = c_{ij}$，对这样的 (i, j) 我们称作" 紧边（tight）"，并把 w_{ij} 也加入等速提升的队伍中；
3. 如果出现某个（或某些）设施 i 满足 $\sum_{j \in D} w_{ij} = f_i$，对这样的设施我们称作" 暂时开放（temporarily open）"，并不再提升与之相连的紧边的 w_{ij} 和 v_j；对于" 暂时开放" 的设施 i，称与其有紧边相连（即 $v_j - w_{ij} = c_{ij}$）的用户 j 为" 连上（connected）" 了该设施。
4. 如果所有的 v_j 都不再能提升，第一阶段算法结束。

可以看出：第一阶段算法结束后，每一个用户 j 都将连上至少一个" 暂时开放" 的设施。此时，可能会出现某个用户 j 对多个" 暂时开放" 的设施投入了 $w_{ij} > 0$ 的资金，我们希望做些调整，使得每一位用户只对其最终连接的设施做贡献，于是有了第二阶段。

第二阶段

首先构造一张二分图 T，一侧是设施，一侧是用户，且我们仅对所有 $w_{ij} > 0$ 连边。



第二阶段（续）

考虑图 T^2，即顶点与 T 相同、将 T 中相邻或由一个共同邻点的一对顶点都连上边。设暂时开放的设施集合为 F_t，T^2 中由 F_t 导出的子图为 H，$I \subseteq F_t$ 是 H 中的极大独立集。也就是说：I 是" 暂时开放" 的设施集合的子集：这些设施在 T 中没有共同用户邻点（独立），且不在 I 中的" 暂时开放" 设施一定与 I 中某设施有共同用户邻点（极大）。令 I 为" 正式开放" 的设施。

记下来分配用户到 I 中的设施，在算法实现上我们理应采用就近分配，但为了后续证明方便这里构造一种分配方式。

- 若存在开放设施 $i \in I$ 使得用户 j 连上了 i，那么称 j 为直连（directly connected）用户。可以直接选择用设施 i 服务用户 j，即令 $\phi(j) = i$。
- 否则，不存在开放设施 $i \in I$ 使得 j 与 i 相连，称 j 为旁连（indirectly connected）用户；设 i' 为 j 第一个连上的设施，且 $i \in I$ 与 i' 在 H 中相邻，那么令 $\phi(j) = i$。

第二阶段（续）

上述这种方案保证了：不存在用户对两个不同的正式开放设施作出贡献。同时，若一个用户对某开放设施投入了资金（他一定是直连用户），则他一定被该设施服务，即保证了互补松驰条件里的第 3 项。

引理：旁连用户的连接费用

引理

设有旁连用户 j，$\phi(j) = i$，那么 $c_{ij} \leq 3v_j$。

证明.

设 j 连上了某个暂时开放但未正式开放的设施 i'，由于 i 与 i' 相冲突，因此存在 j' 使得 $w_{ij'} > 0$、$w_{i'j'} > 0$，即 j' 对 i 和 i' 都做了投资。那么 $v_j \geq c_{i'j}$、$v_{j'} \geq c_{ij'}$、$v_{j'} \geq c_{i'j'}$。注意到：当 v_j 不能提升的时候，$v_{j'}$ 一定也不能提升了，因为 v_j 与 i' 相连，此时设施 i' 已经暂时开放且 (i', j') 是紧边，因此 $v_{j'} \leq v_j$。根据度量性质

$$c_{ij} \leq c_{i'j} + c_{ij'} + c_{i'j'} \leq v_j + v_j + v_{j'} \leq 3v_j$$

□

定理：3-近似算法

定理

上述基于原始-对偶的 *UFL* 近似算法是 *3*-近似算法，且数字 *3* 是紧的。

证明.

我们根据该算法构造出一个 UFL 的整数解：若 $i \in I$ 则 $y_i = 1$，否则 $y_i = 0$；若 $\phi(j) = i$ 则 $x_{ij} = 1$，否则 $x_{ij} = 0$。我们假设算法先假设算法结束后用户 j 的开销 v_j 可以分为两部分：$v_j = v_j^{(f)} + v_j^{(c)}$，其中 $v_j^{(f)}$ 表示投入给设施建设的费用，$v_j^{(c)}$ 表示在路费上的开销。注意到所有开放设施的贡献都来自直连用户，而旁连用户的开销仅在路费上，因此

- 对于直连的用户，$v_j^{(c)} = c_{ij}$，$v_j^{(f)} = w_{ij}$；
- 对于旁连的用户，$v_j^{(f)} = w_{ij} = 0$，那么 $v_j = v_j^{(c)}$。

□

定理证明（续）

证明（续）.

且有 $\sum_{i\in I} f_i = \sum_{j\in D} v_j^{(f)}$ 成立。根据引理以及直连时 $c_{ij} = v_j^{(c)}$，有

$$\sum_{i\in F}\sum_{j\in D} c_{ij}x_{ij} \leq 3\sum_{j\in D} v_j^{(c)} \text{ 成立。}$$

两式相加即得到

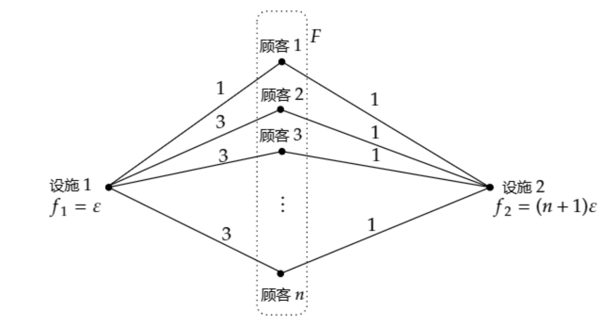
$$3\sum_{j\in D} v_j = 3\sum_{j\in D} v_j^{(f)} + 3\sum_{j\in D} v_j^{(c)} \geq 3\sum_{i\in F} f_i y_i + \sum_{i\in F, j\in D} c_{ij}x_{ij} \geq \sum_{i\in F} f_i y_i + \sum_{i\in F, j\in D} c_{ij}x_{ij}$$

上式右侧是算法给出的 UFL 目标函数值 A_{IP}，左侧是对偶可行解目标函数值的三倍，根据弱对偶定理，左侧项不超过 $3OPT_P \leq 3OPT_{IP}$。因此该算法是一个 3-近似算法。 $\square$

3-近似算法紧性证明

紧性证明.

容易构造一个例子说明 3 是紧的，考虑如下问题实例：



3-近似算法紧性证明

紧性证明（续）.

容易构造一个例子说明 3 是紧的，考虑如下问题实例：

其中 ε 是个足够小的值，那么最优解应该是 $OPT = (n+1)\varepsilon + n$，而我们的算法会令设施 1 和 2 都是" 暂时开放" 设施，用户 1 与两个设施都连上，而其余用户与设施 2 连上，而其余用户与设施 2 连上；H 中仅有 f_1 与 f_2 相连，算法可以仅选择设施 1 " 正式开放"，那么所有用户都会连到设施 1（而且仅有用户 1 直连，其余用户都是旁连），$A_{IP} = 3n - 2 + \varepsilon$，而优解是选择设施 2 开放，即 $OPT_{IP} = n + (n+1)\varepsilon$，令 $n \to \infty$ 即可得到 3 的近似比。 □

The End