中山大學
SUN YAT-SEN UNIVERSITY

Approximation Algorithm

汇报人：肖子立 肖霖畅

中山大學
SUN YAT-SEN UNIVERSITY

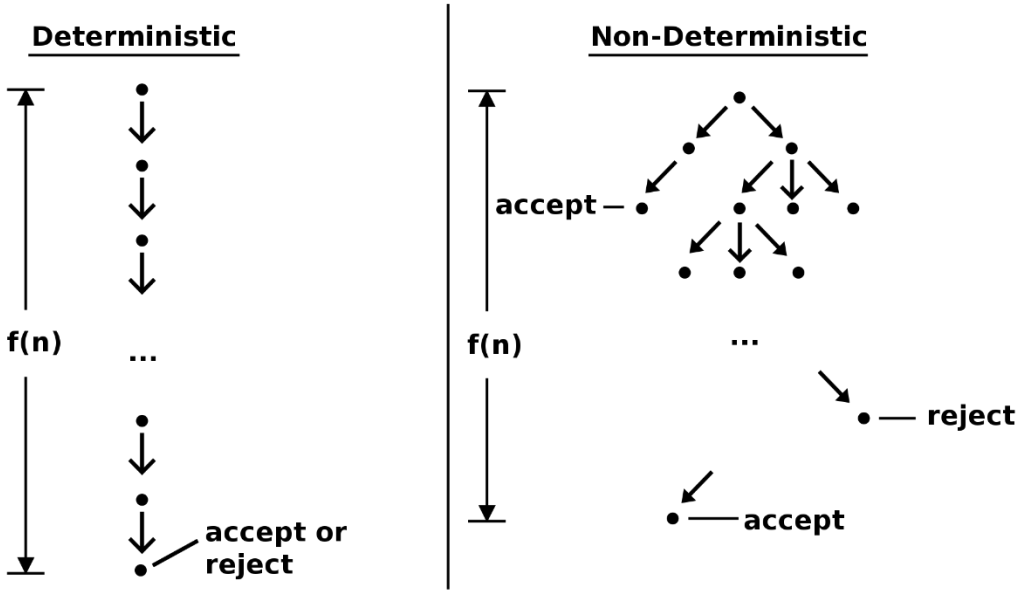
第一部分

大纲

1. **NP-完全性理论**
2. 贪心算法
3. 局部搜索算法
4. 动态规划
5. 科研应用

NP-完全性理论

- 确定型和非确定型图灵机
  - 确定型图灵机（DTM）
    - 根据当前的输入和状态存在**确定**的执行的操作
  - 非确定型图灵机（NTM）
    - 根据当前的输入和状态存在**多种不确定**的执行的操作



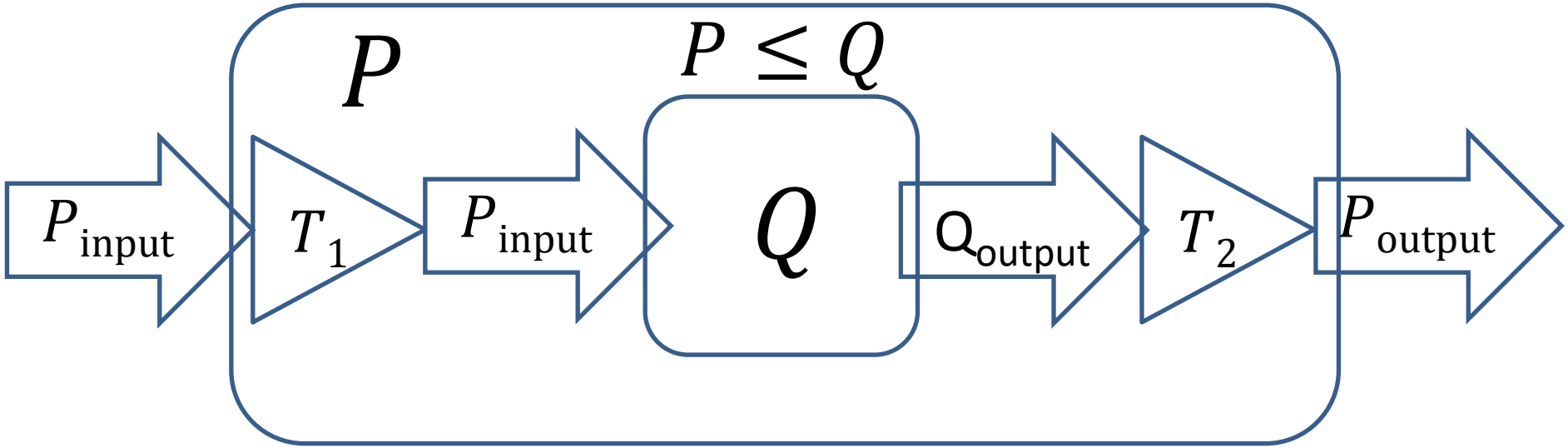
NP-完全性理论

- P和NP问题
  - P类问题：对其中任一问题都存在一个**确定型**图灵机M和一个多项式p，对于该问题的任何长度为n的实例，M都能在p(n)步内给出对该实例的回答。（**多项式时间内可解**）
  - NP类问题：对其中的任一问题都存在一个**非确定型**图灵机N和一个多项式p，对于该问题的任何长度为n的实例，N都能在p(n)步内给出对该实例的回答。（**多项式时间内可验证其解的正确性**）
    - 若NTM在该问题的每一步都存在2种动作可选，则回答实例需考察$2^{p(n)}$种不同的可能性。

P=NP?

NP-完全性理论

- 归约（Reduction）
  - 对于问题P和Q，如果存在一个可计算的函数f，使得对于问题P的任一实例x，都有$\mathrm{P}(x) = Q(f(x))$



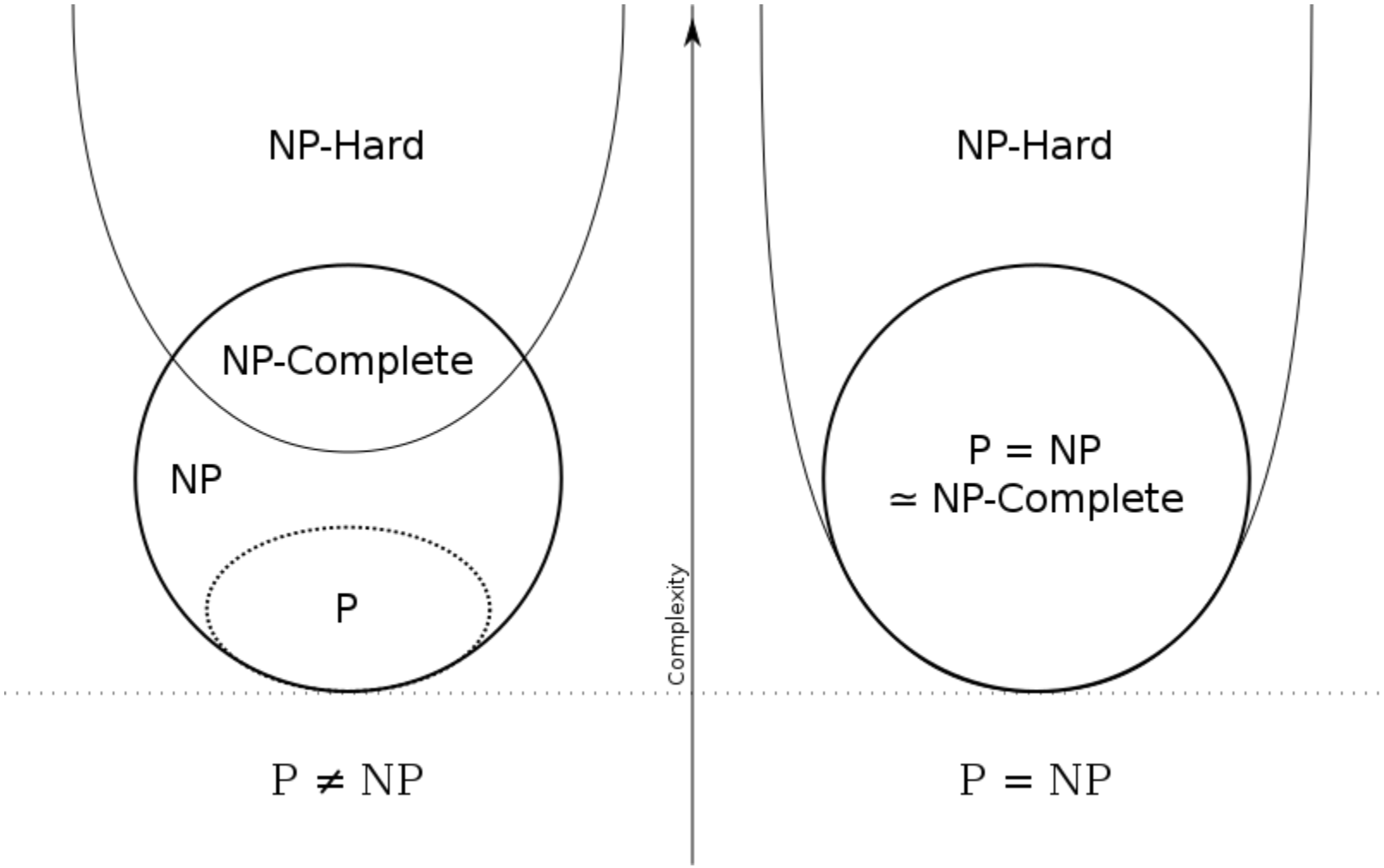
NP-完全性理论

NP及NPC类问题

- NPC问题：
  - 对于一个问题q，若：
    1. q∈NP
    2. NP问题中任一实例均可多项式时间归约到q
  - 则称q为NP-complete（NPC）问题
- NP-hard问题：
  - 若问题q近满足上述条件2而不一定满足条件1，则问题q称为NP-hard（NPH）问题
  - **NP-complete⊆NP-hard**

NP-完全性理论

- P、NP、NPH和NPC之间的关系



NP-完全性理论

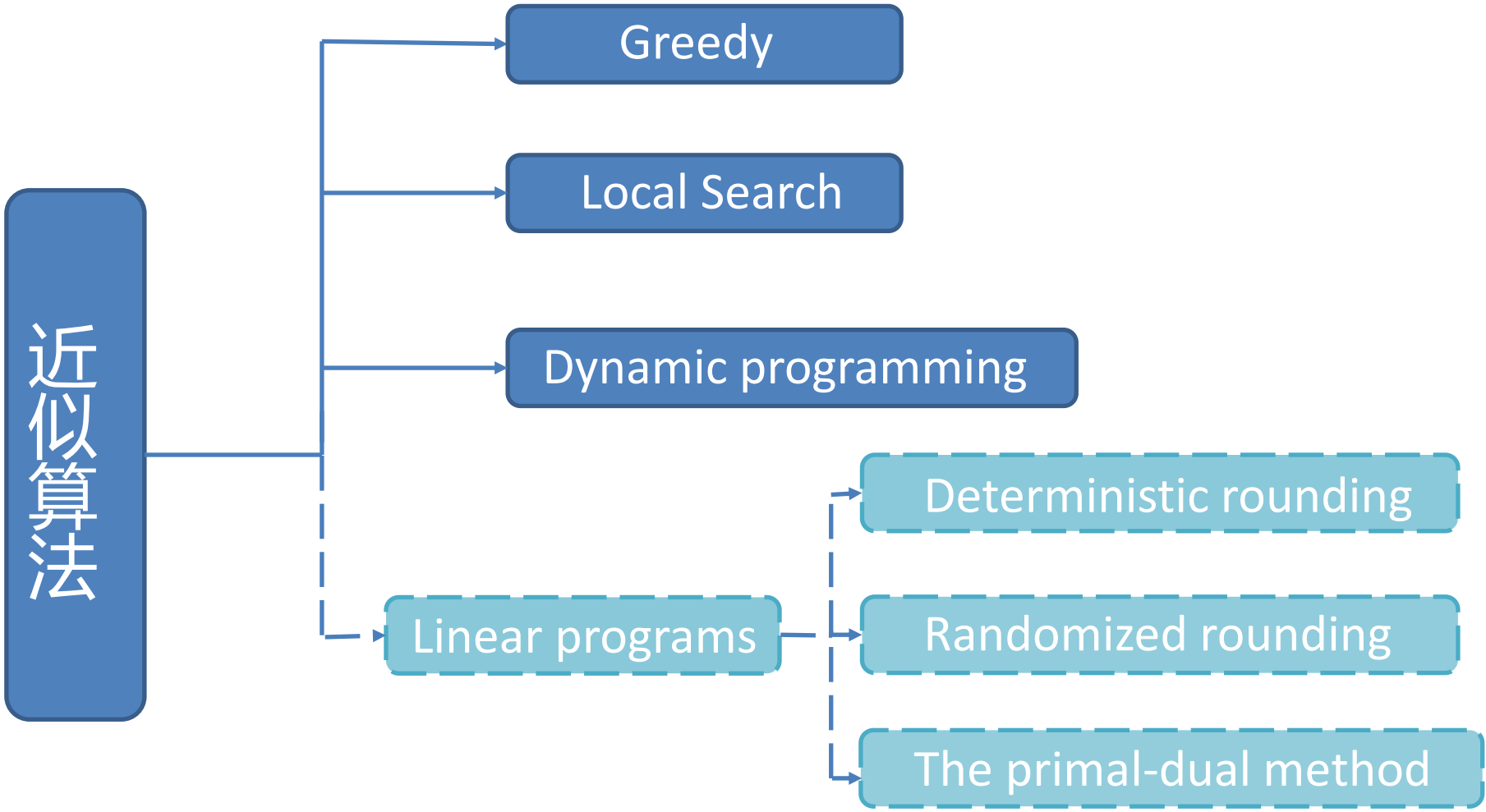
- 解决NPH问题常见算法
  - 启发式算法：
    - 设计多基于研究者直观经验，缺乏理论依据
    - 算法性能不稳定，其可行解与最优解的**偏离程度难以预计**
  - AI类算法
    - 算法产生的可行解与最优解之间的距离难以衡量，缺乏**理论分析**
    - 对**训练数据**要求严格，**计算复杂度**相对较高，对**硬件资源**需求量大
  - 近似算法
    - 算法设计**简单直观**，有严格的**数学理论支撑**
    - 其可行解与最优解之间的距离可以证明，算法性能有**稳定上界**

NP-完全性理论

- 常见优化问题分类：
  - 容易近似：
    - Knapsack，UFLP， Bin Packing等
  - 较难近似：
    - Vertex Cover，Euclidean TSP，Steiner Trees等
  - 难以近似：
    - Graph Coloring，TSP，Clique等

NP-完全性理论

- 常见近似算法分类：

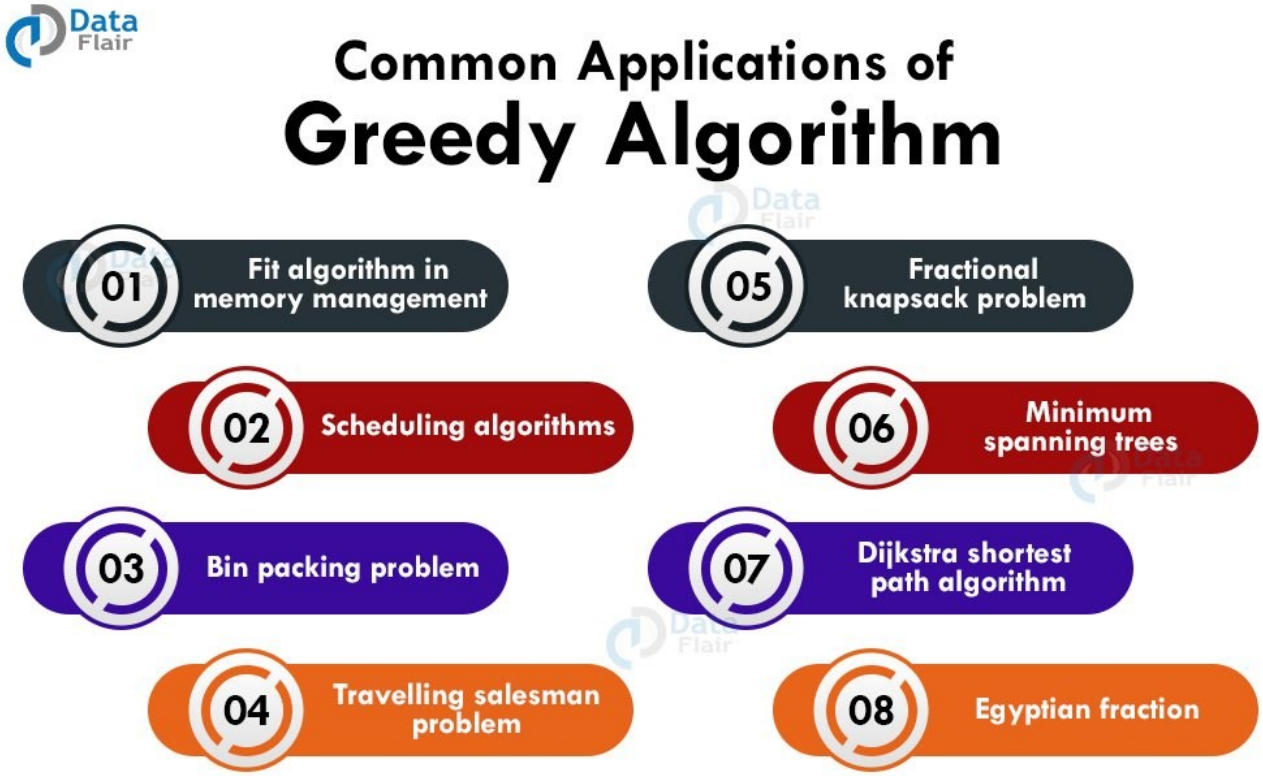


大纲

1. NP-完全性理论
2. **贪心算法**
3. 局部搜索算法
4. 动态规划
5. 科研应用

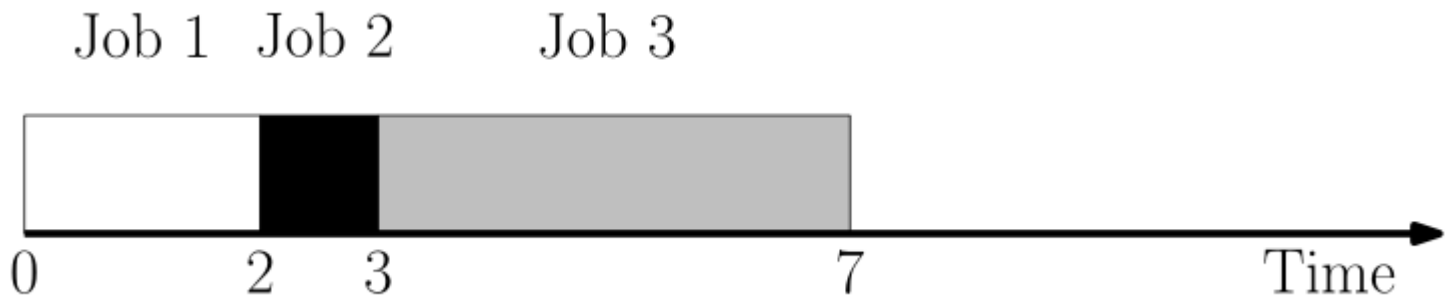
贪心算法

- 贪心算法：
  - 核心思想：总是在当前时刻做出**最优的**选择
  - 目标：从**局部的**最优解最终得到**全局**最优解



贪心算法

- 单机作业调度（Scheduling jobs with deadlines on a single machine）：
  - 每个job具有发布时间r_j，作业执行时间p_j，截止时间d_j
  - 优化目标：最小化作业完成延迟
    - 作业j被完成的时间为C_j
    - 延迟$L_j=C_j-d_j$
    - 目标函数：min $L_{max}=max_{j=1,...,n} L_j$



贪心算法

- 单机作业调度（Scheduling jobs with deadlines on a single machine）：
  - Lemma1:对于job的任何子集S(假设$d(S) < 0$)，都有：

$$L^*_{max} \geq r(S) + p(S) - d(S)$$

其中：L^*_{max}为该优化问题的最优解

$$r(S) = \min_{j \in S} r_j$$

$$p(S) = \sum_{j \in S} p_j$$

$$d(S) = \max_{j \in S} d_j$$

贪心算法

- 单机作业调度（Scheduling jobs with deadlines on a single machine）：
  - Theorem1：earliest due date（EDD）是一个近似比为2的算法:

$$t \le C_j, r(S) = t$$

$$p(S) = C_j - t = C_j - r(S), C_j \le r(S) + p(S)$$

Lemma1和 $d(S) < 0$

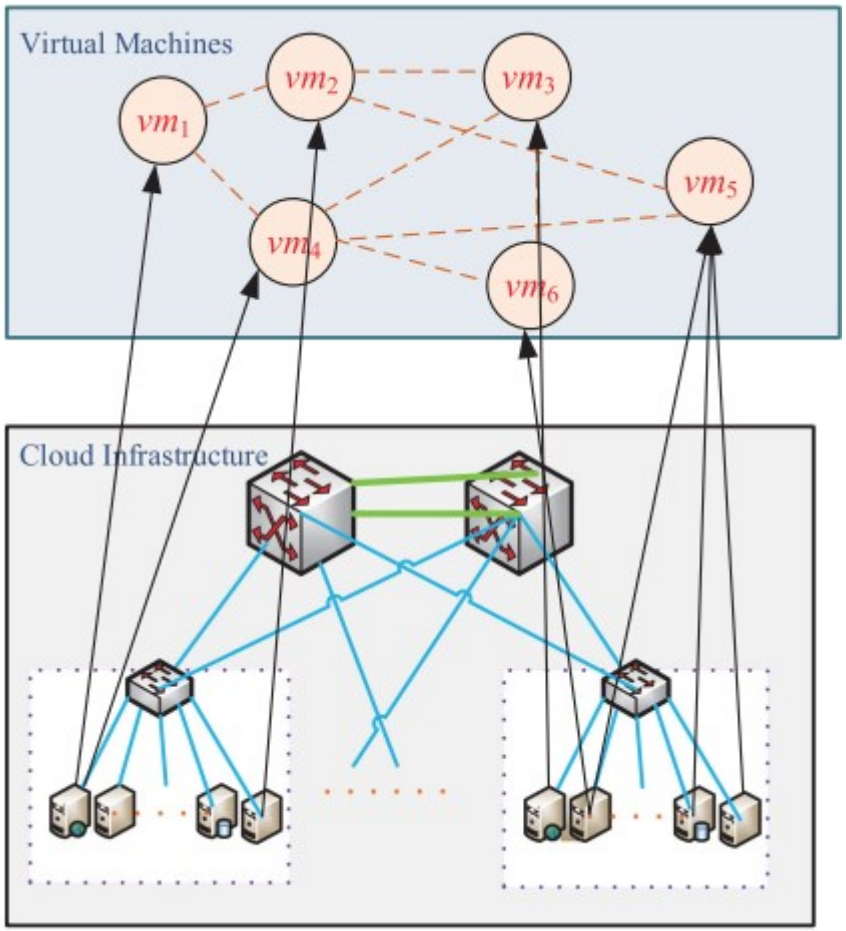
$$\boldsymbol{L^*_{max}} \ge r(S) + p(S) - d(S) \ge r(S) + p(S) \ge \boldsymbol{C_j}$$

$$\boldsymbol{L^*_{max}} \ge r_j + p_j - d_j \ge \boldsymbol{-d_j}$$

$$\boldsymbol{L_{max}} = C_j - d_j \le \boldsymbol{2L^*_{max}}$$

贪心算法

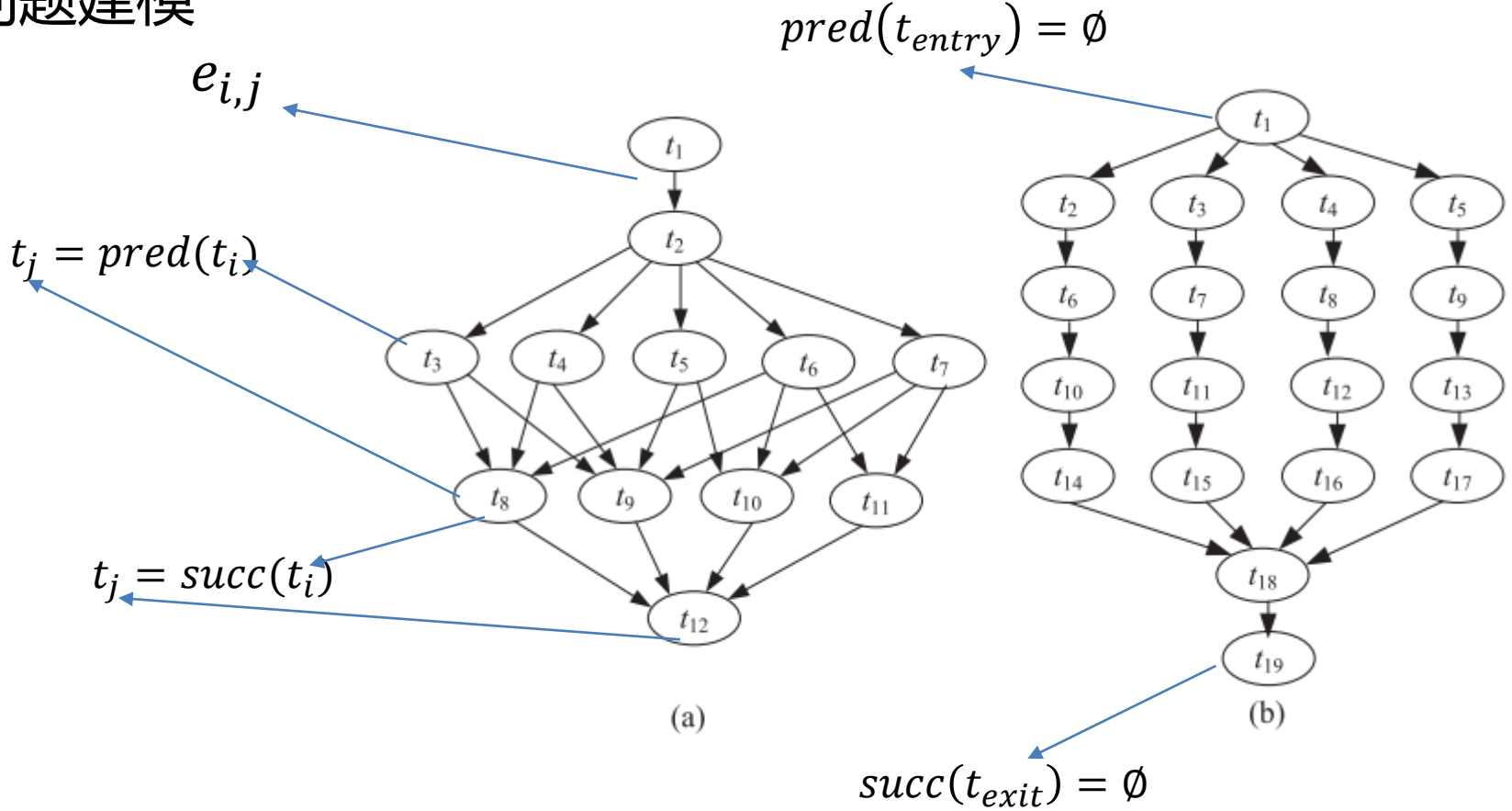
- 问题场景
  - 云工作流任务调度
  - 限制：
    - **数据读写**
    - **任务截止日期**
    - **租用云资源成本**
  - 目标：
    - 满足任务截止日期的情况下，**最小化租用金额**



[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

- 问题建模



[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

问题建模

任务开始时间：

$$EST(t_{entry}, vm_j) = 0.$$

任务执行时间：

$$ET(t_i, vm_j) = d_i^I / GSR(vm_j) + MI_i / w(vm_j) + d_i^O / GSW(vm_j).$$

任务完成时间：

$$\begin{aligned} EFT(t_i, vm_j) &= EST(t_i, vm_j) + ET(t_i, vm_j) \\ &= EST(t_i, vm_j) + d_i^I / GSR(vm_j) \\ &\quad + MI_i / w(vm_j) + d_i^O / GSW(vm_j). \end{aligned}$$

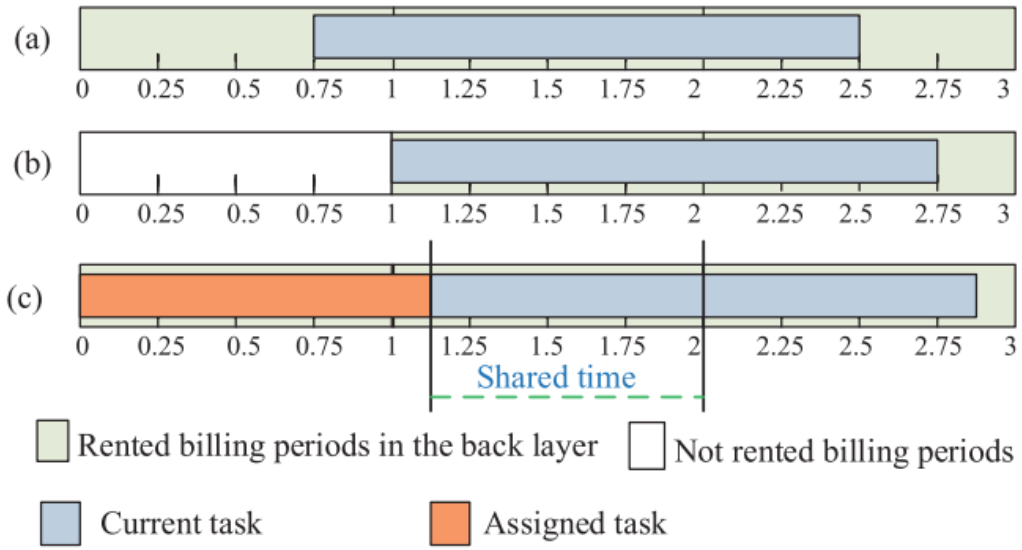
任务开始时间限制：

$$\begin{aligned} EST(t_i, vm_j) \geq & Max\{Available(vm_j), \\ & EFT_{t_k \in pred(t_i)}(t_k, vm_s)\}. \end{aligned}$$

[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

- 问题建模



$$c(t_i, vm_j) = \begin{cases} (\lceil ET(t_i, vm_j)/\tau \rceil + 1) \times c(vm_j), \\ \lceil ET(t_i, vm_j)/\tau \rceil \times c(vm_j), \\ (\lceil ET(t_i, vm_j)/\tau \rceil - 1) \times c(vm_j). \end{cases}$$

[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

问题形式化定义

优化目标：最小化

$$Cost = \sum_{t_i \in T, j=1,\cdots,m} X_{i,j} c(t_i, vm_j).$$

约束条件：

$$\begin{cases} makespan \leq d, & \\ EST(t_i, vm_j) \geq Max\{Available(vm_j), EFT_{t_k \in pred(t_i)}(t_k, vm_s)\} & \forall t_i, \\ \sum_{j=1,\cdots,m} X_{i,j} = 1 & \forall i, \\ X_{i,j} \in \{0,1\} & \forall i, j. \end{cases}$$

[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

- 问题复杂度分析（即证明问题是NP-hard）

Cloud workflow scheduling

$$Cost = \sum_{t_i \in T, j=1,\cdots,m} X_{i,j} c(t_i, vm_j).$$

$$\begin{cases} makespan \leq d, \\ EST(t_i, vm_j) \geq Max\{Available(vm_j), EFT_{t_k \in pred(t_i)}(t_k, vm_s)\} & \forall t_i, \\ \sum_{j=1,\cdots,m} X_{i,j} = 1 & \forall i, \\ X_{i,j} \in \{0,1\} & \forall i,j. \end{cases}$$

reduce

MKP

$$\begin{cases} Max \Sigma_{i=1}^{q} \Sigma_{j \in Q_i} \alpha_{i,j} Z_{i,j} \\ \textbf{s.t.} \\ \Sigma_{i=1}^{q} \Sigma_{j \in Q_i} \beta_{i,j}^{k} Z_{i,j} \leq b^k, \\ \sum_{j \in Q_i} Z_{i,j} = 1 & \forall i, \\ Z_{i,j} \in \{0,1\} & \forall i,j. \end{cases}$$

[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

贪心算法

- 算法思想：

对所有**父任务**进行调度

计算每个任务在每种机器上的**执行时间及相应价格**

对所有任务花费**排序**

优先调度**花费最少**任务

更新DAG及任务调度集合

Algorithm 2. Greedy workflow scheduling algorithm

Input: Workflow DDAG, cloud VMs
Output: The schedule of task-VM pairs
1: Put t_{entry} into schedulable tasks set ω
2: **while** Schedulable tasks ω is not empty **do**
3: **for** each tasks t_i in ω **do**
4: **for** each VMs **do**
5: Calculate task $EFT(t_i, vm_j)$ by Eq. (3)
6: Calculate task execution cost $c(t_i, vm_j)$ by Eq. (7)
7: **end**
8: **end**
9: Sort these task-VM pairs according to $c(t_i, vm_j)$ by increasing order
10: Find the first task-VM pair that satisfies $EFT(t_i, vm_j) \leq d_i$
11: Assign task t_i to corresponding VM
12: Update VM and task status
13: **for** each tasks t_x in $succ(t_i)$ **do**
14: Remove task t_i from task t_x's $pred(t_x)$
15: **if** $pred(t_x)$ is empty **then**
16: Put task t_x into schedulable tasks ω
17: **end**
18: **end**
19: Remove task t_i from schedulable tasks ω
20: **if** level group $EFT(T_k) \leq EFT(t_i, vm_j)$ **then**
21: Adjust level group subdeadline according to Eq. (14)
22: Update unscheduled tasks $d_i = d(T_k)$
23: **end**
24: **end**

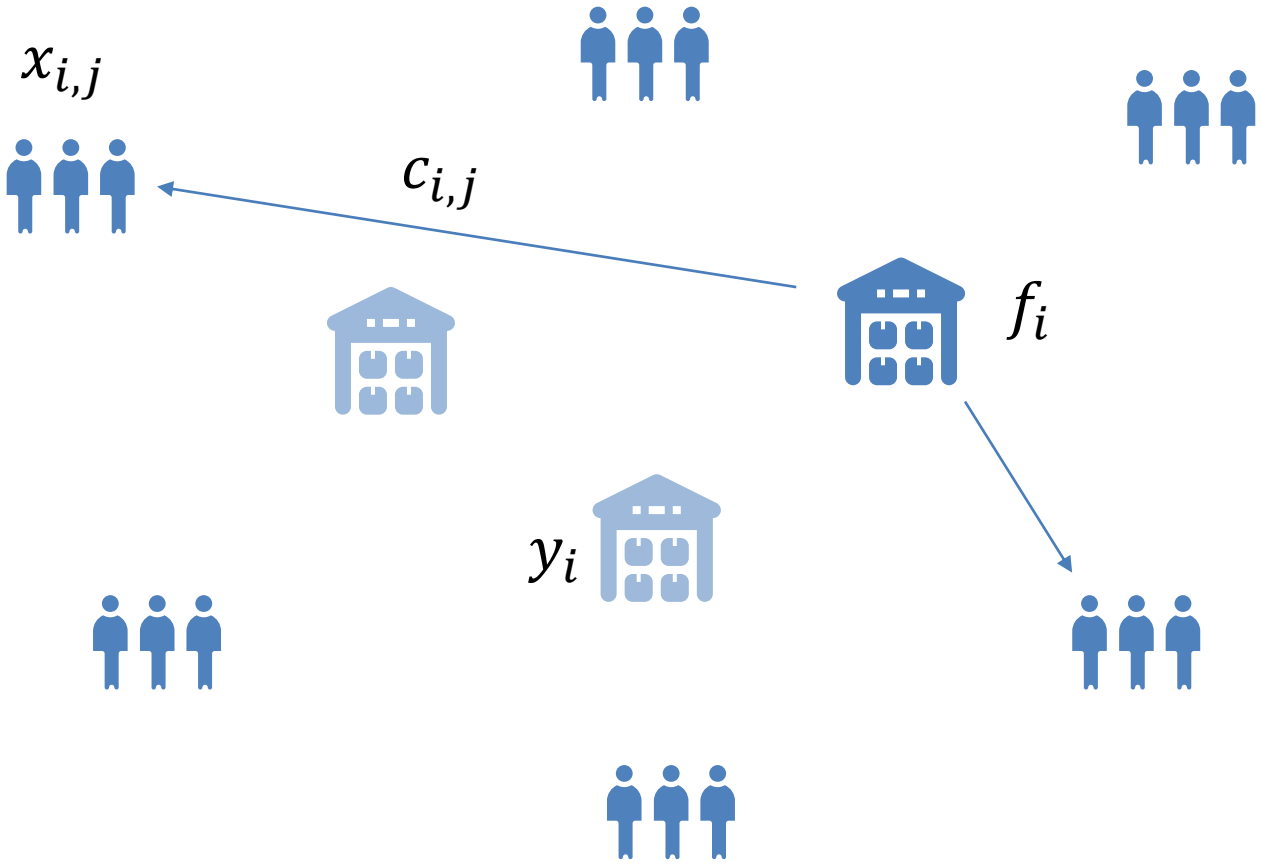
[1]Tang X, Cao W, Tang H, et al. Cost-efficient workflow scheduling algorithm for applications with deadline constraint on heterogeneous clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(9): 2079-2092.

大纲

1. NP-完全性理论
2. 贪心算法
3. **局部搜索算法**
4. 动态规划
5. 科研应用

局部搜索算法

- 无容量限制的设备选址问题（uncapacitated facility location problem, UFLP）



局部搜索算法

- 无容量限制的设备选址问题（uncapacitated facility location problem, UFLP）

$$\text{minimize} \quad \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij}$$

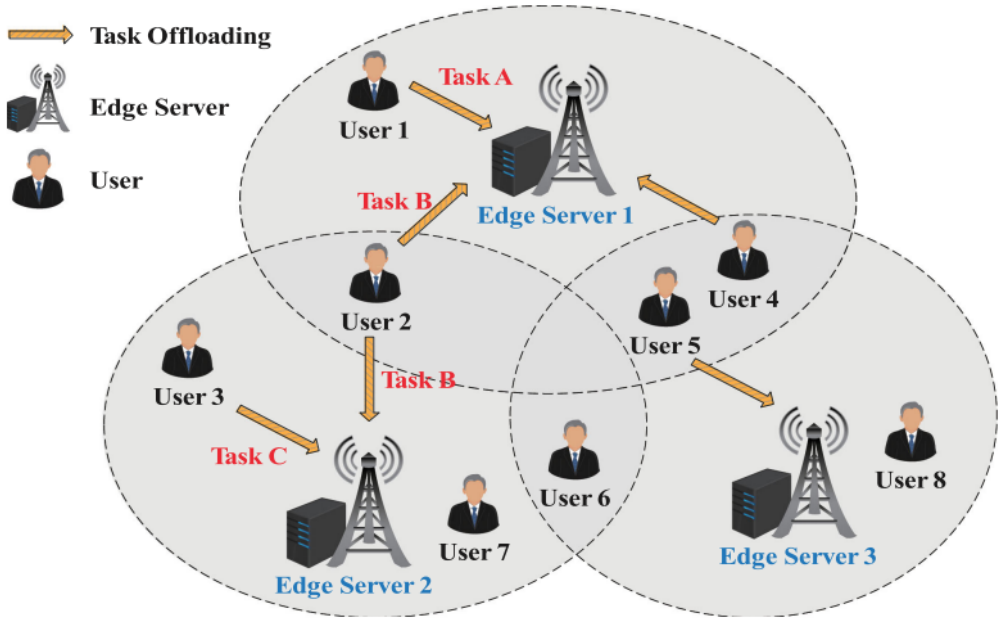
$$\begin{aligned}
\text{subject to} \quad & \sum_{i \in F} x_{ij} = 1, && \forall j \in D, \\
& x_{ij} \leq y_i, && \forall i \in F, j \in D, \\
& x_{ij} \in \{0, 1\}, && \forall i \in F, j \in D, \\
& y_i \in \{0, 1\}, && \forall i \in F.
\end{aligned}$$

局部搜索算法

- 应用场景
  - 设备选址问题（UFLP）
  - K-media问题
  - Minimum-degree spanning trees问题
  - Edge coloring问题

局部搜索算法

- 问题场景
  - MEC环境下的任务卸载
  - 限制：
    - **边缘端计算资源**
    - **用户请求延迟**
  - 目标：
    - 最小化**任务卸载成本**（服务放置成本、边缘节点计算成本和能源消耗成本）



[2] Chen Y, Zhang S, Jin Y, et al. LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(7): 1581-1592.

局部搜索算法

- 问题建模

$$\mathbb{P}_1 : \min_{\boldsymbol{X},\boldsymbol{Y}} \quad C^p + C^u + C^e$$

$$\text{s.t.} \quad x_{n,m} \in [0,1], \qquad \forall n \in \mathcal{N}, \forall m \in \mathcal{M},$$

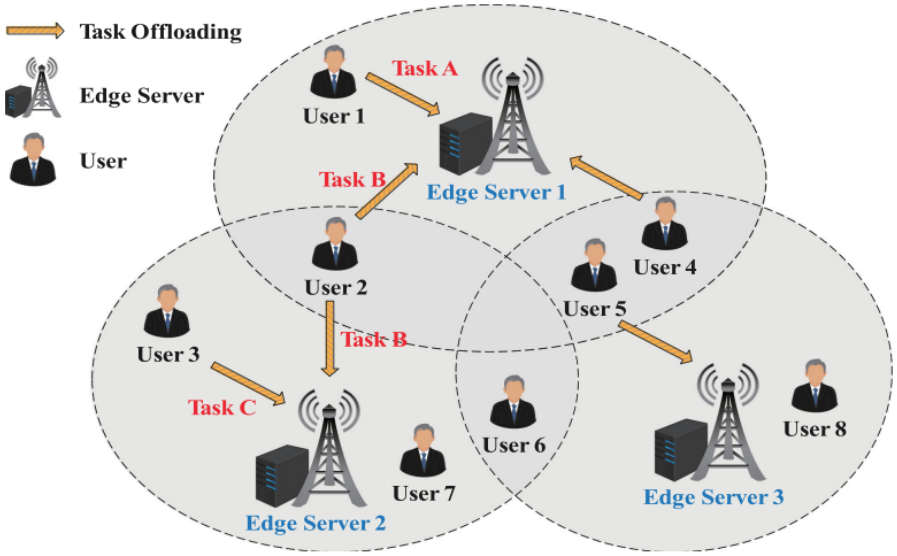
$$y_{m,s} \in \{0,1\}, \qquad \forall m \in \mathcal{M}, \forall s \in \mathcal{S},$$

$$\sum_{m \in \mathcal{M}_n} x_{n,m} = 1, \qquad \forall n \in \mathcal{N},$$

$$x_{n,m} \le y_{m,s}, \qquad \forall n \in \mathcal{N}_s,$$

$$\sum_{n \in \mathcal{N}_m} x_{n,m} l_n \le L_m, \qquad \forall m \in \mathcal{M},$$

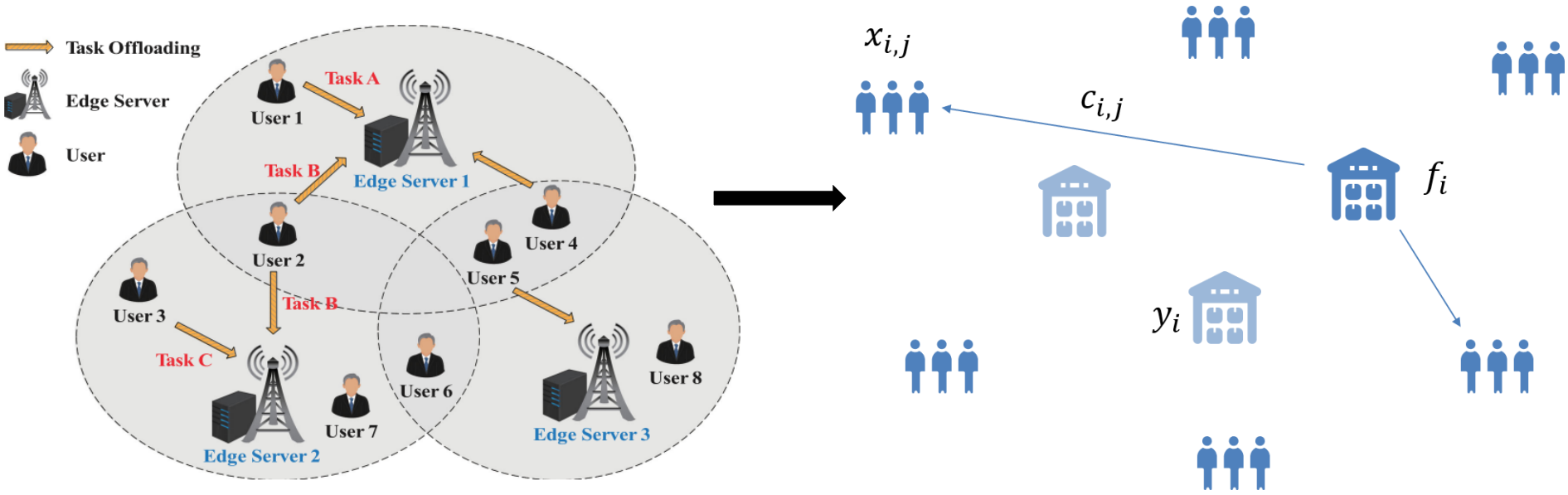
$$D^c_{n,m} + D^t_{n,m} \le d_n, \quad \forall n \in \mathcal{N}, \qquad \forall m \in \mathcal{M}.$$



[2] Chen Y, Zhang S, Jin Y, et al. LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(7): 1581-1592.

局部搜索算法

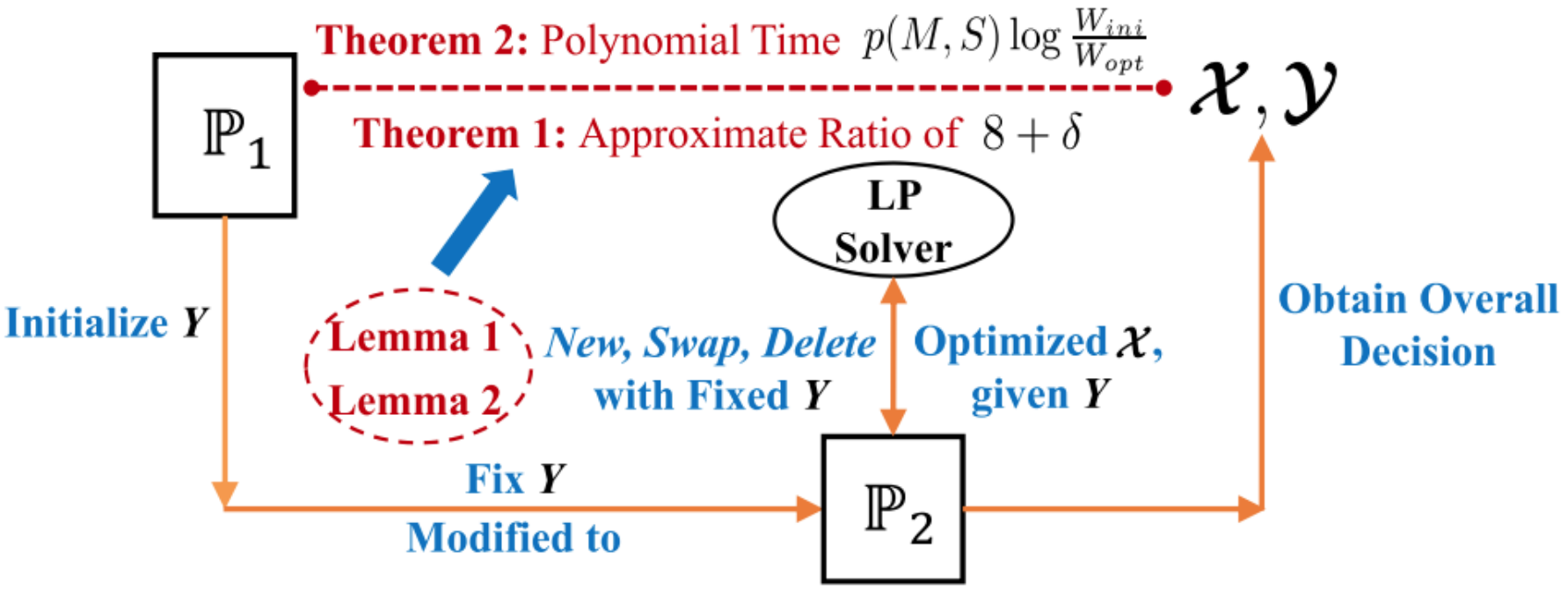
- 问题复杂度分析（即证明问题是NP-hard）



[2] Chen Y, Zhang S, Jin Y, et al. LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(7): 1581-1592.

局部搜索算法

- 算法思想



[2] Chen Y, Zhang S, Jin Y, et al. LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(7): 1581-1592.

局部搜索算法

算法性能理论证明

作者直接从UFLP问题相关研究论文（SODA’ 00）直接引用得到

$$w_e(\boldsymbol{Y}) < w(\boldsymbol{Y}^*) + MSw(\boldsymbol{Y})/p(M,S)$$

$$w_s(\boldsymbol{Y})(1-\frac{(MS)^2}{p(M,S)}) < 5w(\boldsymbol{Y}^*) + 2w_e(\boldsymbol{Y}) + \frac{w(\boldsymbol{Y})}{MS}$$

$$w_s(\boldsymbol{Y})\left(1-\frac{(MS)^2}{p(M,S)}\right) < 7w(\boldsymbol{Y}^*) + \frac{2MSw(\boldsymbol{Y})}{p(M,S)} + \frac{w(\boldsymbol{Y})}{MS}$$

$$w_e(\boldsymbol{Y})(1-\frac{(MS)^2}{p(M,S)}) < w(\boldsymbol{Y}^*) + \frac{MSw(\boldsymbol{Y})}{p(M,S)}$$

$$w(\boldsymbol{Y})\left(1-\frac{(MS)^2}{p(M,S)}\right) \leq 8w(\boldsymbol{Y}^*) + \frac{3MSw(\boldsymbol{Y})}{p(M,S)} + \frac{w(\boldsymbol{Y})}{MS}$$

近似比

$$\frac{w(\boldsymbol{Y})}{w(\boldsymbol{Y}^*)} < 8\frac{1}{(1-\frac{(MS)^2}{p(M,S)}-\frac{3MS}{p(M,S)}-\frac{1}{MS})}$$

[2] Chen Y, Zhang S, Jin Y, et al. LOCUS: User-Perceived Delay-Aware Service Placement and User Allocation in MEC Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(7): 1581-1592.

大纲

1. NP-完全性理论
2. 贪心算法
3. 局部搜索算法
4. **动态规划**
5. 科研应用

动态规划

- 问题：如何用最少的钞票数凑出目标金额

666

✔ 贪心算法

666=6×100+1×50+1×10+1×5+1×1

面值：1　5　10　20　50　100

15

✖ 贪心算法

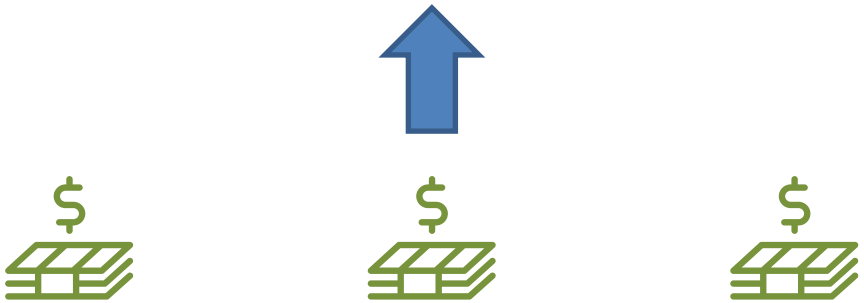
15=1×11+4×1

面值：1　5　11

动态规划

- 问题：如何用最少的钞票数凑出目标金额

15



面值：1　　5　　11

取11：数量=$f(4)+1=4+1=5$
取5：　数量=$f(10)+1=2+1=3$
取1：　数量=$f(14)+1=4+1=5$

$$f(n)=\min\{f(n-1),f(n-5),f(n-11)\}+1$$

动态规划

- 算法思想：
  - 将原问题分解成子问题
  - 构建状态空间
  - 确定初始值
  - 建立状态转移方程
- 适用问题：
  - 最优子结构性
  - 无后效性
  - 子问题重叠性

大纲

1. NP-完全性理论
2. 贪心算法
3. 局部搜索算法
4. **动态规划**
5. 科研应用

科研应用

明确地数学化定义问题场景

根据所定义场景找到可归约的常见问题（如MKP、UFLP等）

阅读相关问题的文献（更多是数学领域相关论文）

整理有近似比证明过程的算法

找到其中最适合本文实际场景定义的算法并进行适当修改应用

推荐参考资料：Williamson D P, Shmoys D B. The design of approximation algorithms[M]. Cambridge university press, 2011.

中山大學
SUN YAT-SEN UNIVERSITY

第二部分

大纲

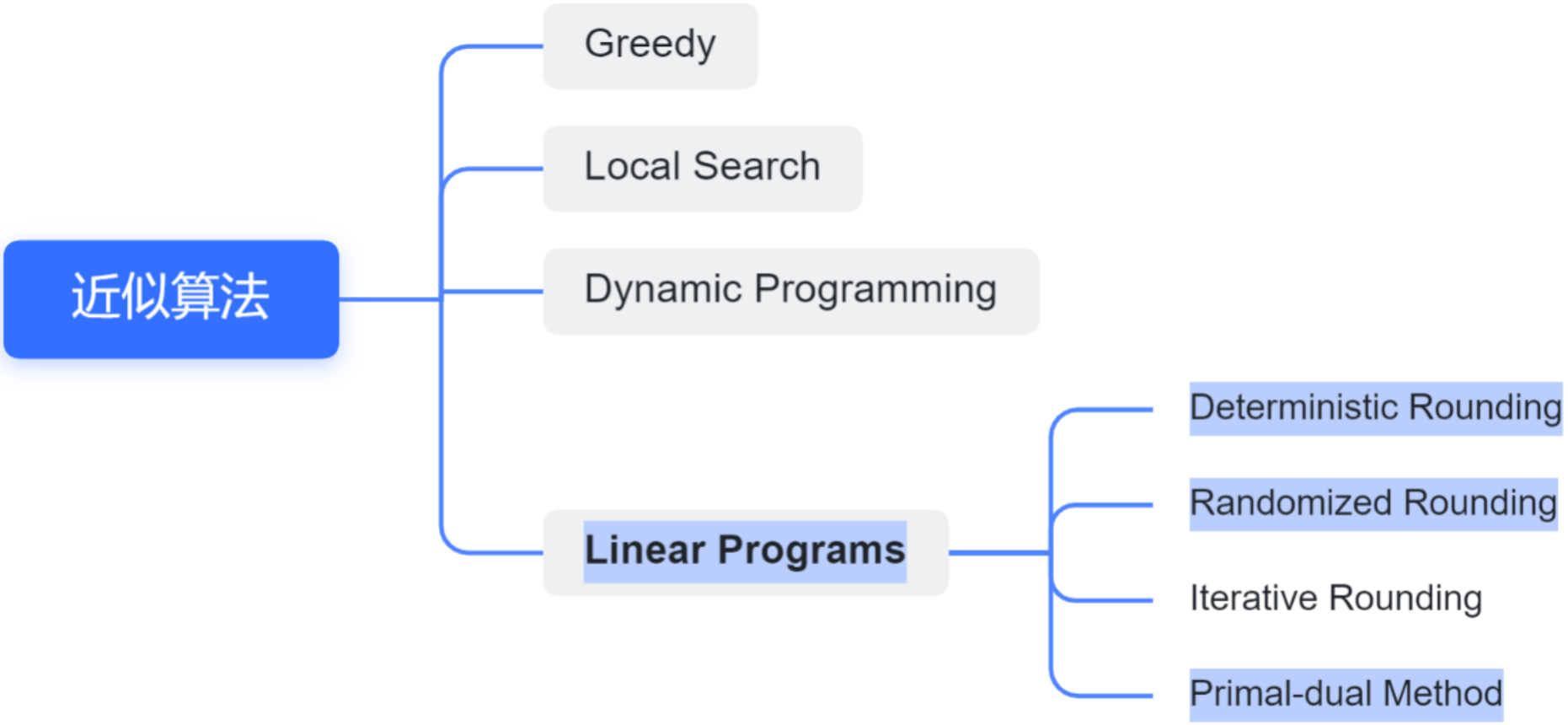
1. **近似算法基本设计步骤**
2. 近似算法中线性规划法的分类和应用
3. 在线算法的简单介绍
4. 未来研究方向和现有问题

近似算法证明的流程

- 假设我们想要证明一个算法 ALG 是一个对某些**最小化cost问题**的 α-近似 算法，通常的证明流程：
  - 对任何的输入实例I，找到OPT cost的下界(Lower Bound，LB)：$LB(I) \leq c\big(OPT(I)\big), \forall I$
  - 证明对任何的输入实例I， 都有：$c\big(ALG(I)\big) \leq \alpha LB(I), \forall I, \alpha \geq 1$
  - 推断出结论：$c(ALG) \leq \alpha LB \leq \alpha c(OPT)$

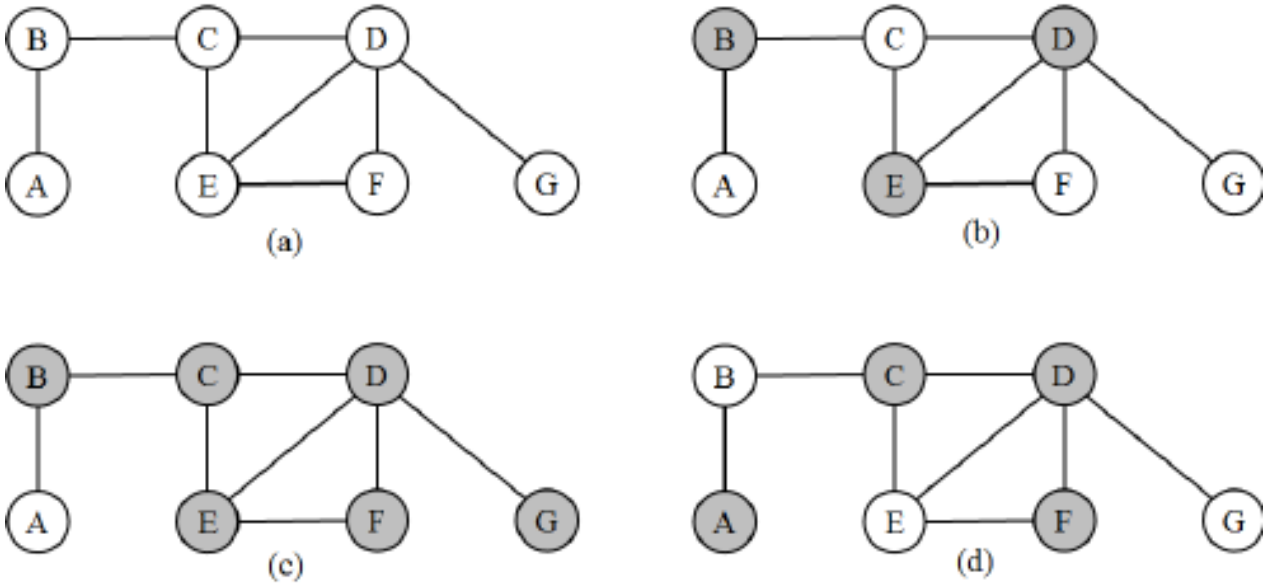
- Issue：
  - Issue 1：难以找到下界LB以及最优解OPT和下界LB的关系
  - Issue 2：找到了问题的下界LB，却找不到算法ALG可以与LB联系
- 解决方案：Linear programming(LP)

常见近似算法分类


近似算法
Greedy
Local Search
Dynamic Programming
Linear Programs
Deterministic Rounding
Randomized Rounding
Iterative Rounding
Primal-dual Method

线性规划——寻找问题下界LB的工具

- Toy Problem —— **Weighted Vertex Cover (WVC)**
  - Vertex Cover：无向图 $G = (V, E)$ 的一个顶点覆盖是一个子集V' ⊆ V，使得如果(u，v)是G的一条边，则$u \in V'$，或者$v \in V'$。一个顶点覆盖的规模是其中所包含的顶点数。
  - Weighted Vertex Cover：每个顶点具有权重
  - 优化目标：获得一个无向图G的具有最小权重和的顶点覆盖。



线性规划——寻找问题下界LB的工具

- Toy Problem —— **Weighted Vertex Cover (WVC)**
  - 步骤1：将问题规约到整数规划问题(Integer Program, IP)
    - Note: 约束1表示对任何一条边，边对应的顶点至少有一个点被算法ALG选中

$$\text{OBJECTIVE FUNCTION:} \quad \min \sum_{v \in V} c(v) x_v$$

$$\begin{aligned} \text{CONSTRAINTS:} \quad & x_u + x_v \geq 1 \qquad \forall \{u, v\} \in E \\ & x_v \in \{0, 1\} \qquad \forall v \in V \end{aligned}$$

    - 可行解(feasible solution)：满足线性约束和整数变量约束的方案
    - 最优解(optimal solution)： 可行解中最小化目标函数的方案

线性规划——寻找问题下界LB的工具

- Toy Problem —— **Weighted Vertex Cover (WVC)**
  - 步骤1：将问题规约到整数规划问题(Integer Program, IP)
  - 步骤2：将整数规划放松(Relax)成线性规划(Linear Program, LP)并获得下界

$$\text{OBJECTIVE FUNCTION:} \quad \min \sum_{v \in V} c(v) x_v$$

$$\begin{aligned} \text{CONSTRAINTS:} \quad & x_u + x_v \geq 1 \qquad \forall \{u, v\} \in E \\ & 0 \leq x_v \leq 1 \qquad \forall v \in V \end{aligned}$$

  - **重要通用性质**：任何一个整数规划(IP)的可行解也是它对应的放松线性规划问题(LP)的可行解，因此，$c(Z_{LP}^*) \leq c(Z_{IP}^*) = OPT$
  - 一个简单的推论：使用IP建模问题，其对应的放松LP问题的最优解就是OPT的一个下界，完成了近似算法构造的第一步：对任何的输入实例I，找到OPT cost的下界(Lower Bound, LB)：$LB(I) \leq c\big(OPT(I)\big), \forall I$

线性规划——利用线性规划做近似算法

- 回顾NP-完全性理论
  - 多项式时间算法：对于规模n的输入，在最坏情况下的运行时间是$O(n^k)$，其中k为某一确定常数。
    - Note: 通常来说指数级别的算法计算量增长非常快，而即使多项式算法也不一定完全满足实时性要求。用计算复杂度O衡量并不是100%合理，因为只考虑最差情况
  - P类问题：**多项式时间内可解**
  - NP类问题：不能在多项式时间内解决或不确定能不能在多项式时间内解决，但**能在多项式时间验证的问题**
  - NP-Complete类问题：所有NP问题**在多项式时间内**都能规约(Reducibility)到它的NP问题，即解决了此NPC问题，所有NP问题也都得到解决
- 整数规划(IP)：NP-Complete问题
- 线性规划(LP)：P类问题 => 适合作为IP问题的近似解法

线性规划——利用线性规划做近似算法

- Toy Problem —— **Weighted Vertex Cover (WVC)**
  - 步骤1：将问题规约到整数规划问题(Integer Program, IP)
  - 步骤2：将整数规划放松(Relax)成线性规划(Linear Program, LP)并获得下界
  - 步骤3：多项式时间内得到LP的最优解Z_{LP}^*
    - Note: 得到的解是分数解，无法直接满足IP问题的整数约束
  - 步骤4：[近似算法: 一个最简单的舍入(Rounding)方法] 如果$x_v \geq 1/2$，舍入到1；反之舍入到0。算法结束。
    - Note: 这一步就是近似算法，我们需要证明满足IP问题的约束，同时通过证明近似比分析其性能。
- 证明近似比的步骤：
  - 步骤1：证明 $c(Z_{LP}^*) = LB(I) \leq c\big(OPT(I)\big), \forall I$，已经完成
  - 步骤2：证明对任何的输入实例I， 都有：$c\big(ALG(I)\big) \leq \alpha c(Z_{LP}^*), \forall\, I$

线性规划——利用线性规划做近似算法

- Toy Problem —— **Weighted Vertex Cover (WVC)**
  - 目标1：证明ALG算法满足原始问题的约束条件
  - 目标2：证明近似算法ALG与原问题下界LB(即线性规划最优解)的不等式：$c\big(ALG(I)\big) \le \alpha c(Z^*_{LP}), \forall\, I$

$$\text{OBJECTIVE FUNCTION:} \quad \min \sum_{v \in V} c(v) x_v$$

$$\begin{aligned} \text{CONSTRAINTS:} \quad & x_u + x_v \ge 1 \qquad \forall \{u, v\} \in E \\ & 0 \le x_v \le 1 \qquad \forall v \in V \end{aligned}$$

  - 分析1：根据约束1可知，对于每条边的两个点，步骤3得到的Z^*_{LP}必然满足$x_u \ge \frac{1}{2}$ 或 $x_v \ge \frac{1}{2}$ ，在ALG中必然存在一个u或v被设置为1，因此ALG满足原始问题的约束条件
  - 分析2：对任务的输入实例I都有
    - $c(ALG) = \sum_v c(v) x_v(ALG) = \sum_{v: x_v(ALG)=1} c(v) = \sum_{v: x_v(Z^*_{LP}) \ge \frac{1}{2}} c(v)$
    - $\sum_{v: x_v(Z^*_{LP}) \ge \frac{1}{2}} c(v) \le 2 \sum_{v: x_v(Z^*_{LP}) \ge \frac{1}{2}} c(v) x_v(Z^*_{LP}) \le 2\, c(Z^*_{LP}) \le 2c(OPT)$

线性规划——Integrality Gap

- Integrality Gap
  - 回顾近似算法的证明目标1：对任何的输入实例I，找到OPT cost的下界(Lower Bound, LB)：$LB(I) \leq c(OPT(I))$ ，$\forall I$
  - 引入IP和LP后的证明目标1：天然存在$c(Z_{LP}^*) \leq c(Z_{\mathrm{IP}}^*) = OPT$
    - 问题：使用LP作为下界，这个下界是否足够好，用Integrality Gap衡量

$$\sup_{instances\ I\ of\ \Pi} \left(\frac{Z_{IP}^*(I)}{Z_{LP}^*(I)} \right)$$

  - 利用LP构造近似算法，其近似比存在一个天然界限：
    - 回顾近似算法的推演路径：$c(ALG) \leq \alpha LB \leq \alpha c(OPT)$
    - 引入IP和LP后的推演路径：$IG \cdot c(ALG) \leq \alpha \cdot IG \cdot \mathrm{c}(Z_{LP}^*) \leq \alpha c(Z_{IP}^*)$
    - 因此通过IP放松成LP而提出的近似算法，最好的近似比就只能是IG。
    - 一些Relax天然会带来一个巨大的IG[见下页]。

线性规划——Integrality Gap

- Integrality Gap
  - 一些Relax天然会带来一个巨大的IG。
    - 上：IP； 下：Relaxed-LP
    - 对m台机器，1个任务，每个任务执行时间都是m的实例：IP得到的c(OPT)=m， 而Relaxed-LP得到的$c(Z_{LP}^*) = 1$。因此IG至少是m，且会随着问题规模的增大而增加。

$$\begin{aligned}
&\text{minimize: } t \\
&\text{subject to: } \sum_{j=1}^{n} x_{i,j} p_{i,j} \le t \qquad 1 \le i \le m \\
&\qquad \sum_{i=1}^{m} x_{i,j} = 1 \qquad 1 \le j \le n \\
&\qquad x_{i,j} \in \{0,1\} \qquad 1 \le i \le m,\ 1 \le j \le n
\end{aligned}$$

$$\begin{aligned}
&\text{minimize: } t \\
&\text{subject to: } \sum_{j=1}^{n} x_{i,j} p_{i,j} \le t \qquad \forall i \in [m] \\
&\qquad \sum_{i=1}^{m} x_{i,j} = 1 \qquad \forall j \in [n] \\
&\qquad x_{i,j} \ge 0 \qquad \forall i \in [m],\ j \in [n]
\end{aligned}$$

解线性规划的多项式时间方法

- Simplex Algorithm 单纯形法
- Ellipsoid Algorithm
- Interior-Point Algorithm 内点法

- 不是本Slide的重点，感兴趣可以去学习最优化理论或运筹学，基本都会涉及到。
- 参考学习资料：
  - 《最优化理论》
  - …

利用线性规划设计近似算法

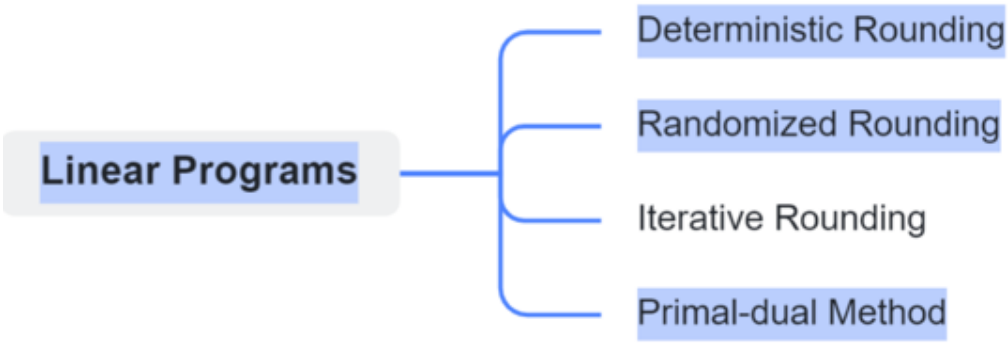
- 步骤1：将问题规约到整数规划问题(Integer Program, IP)
- 步骤2：将整数规划放松(Relax)成线性规划(Linear Program, LP)并获得下界
- 步骤3：多项式时间内得到LP的最优解Z^*_{LP}
- 步骤4：[近似算法]
- 证明近似比

大纲

1. 近似算法基本设计步骤
2. **近似算法中线性规划法的分类和应用**
3. 在线算法的简单介绍
4. 未来研究方向和现有问题

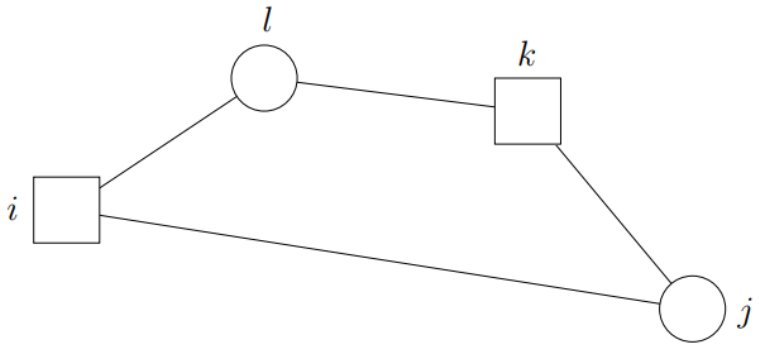
线性规划法的分类

- Deterministic Rounding
  - 从LP的最优分数解舍入到整数解时，舍入方法唯一且确定
- Randomized Rounding
  - 从LP的最优分数解舍入到整数解时，舍入方法的操作过程是随机的
  - 算法的输出结果是随机的，可能正确，可能错误。
- Iterative Rounding
  - 从LP的最优分数解舍入到整数解时，在每轮迭代中先选择部分解，剩余部分再构成子LP问题求解最优分数解，直到获得所有解。
- Primal-dual Method
  - 利用Dual问题的性质进行舍入



线性规划——Deterministic Rounding

- **Problem —— Facility Location[1]**
  - General Facility Location：
    - 输入：一个大小为m的工厂集合V，一个大小为n的客户集合C，工厂i开张成本是$f_i \geq 0$，客户j与工厂i建立链接的链接成本是$c_{ij} \geq 0$
    - 输出：一个拟开放的工厂子集$F \subseteq V$ 使得总成本最小 $\sum_{i \in F} f_i + \sum_{j \in C} \min_{i \in F} c_{ij}$
  - Metric Facility Location：
    - 客户和工厂都存在一个度量空间中
    - 增加约束条件：$c_{ij} \leq c_{il} + c_{kl} + c_{kj}$



  - 其他属性：
    - Capacitated： 客户的需求以分数形式分配给多个工厂
    - **Uncapacitated**：客户的需求以0-1形式分配给一个工厂

1. 自整理FLP问题文档：https://we5lw6jk7r.feishu.cn/wiki/wikcnTyoRXJnmav33KF6QYQH5wh

线性规划——Deterministic Rounding

- **Metric Uncapacitated Facility Location**
  - 步骤1：ILP问题建模
  - 步骤2：放松成LP问题

$$\text{minimize } Z(x,y) = \sum_{i \in V} f_i y_i + \sum_{i,j \in V} c_{ij} x_{ij}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V$$

$$x_{ij} \leq y_i \qquad \forall i,j \in V$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V$$

$$y_i \in \{0,1\} \qquad \forall i \in V$$

$$y_i = \begin{cases} 1 & \text{if } i \in F \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if terminal } j \text{ is assigned to facility } i \\ 0 & \text{otherwise} \end{cases}$$

线性规划——Deterministic Rounding

Metric Uncapacitated Facility Location

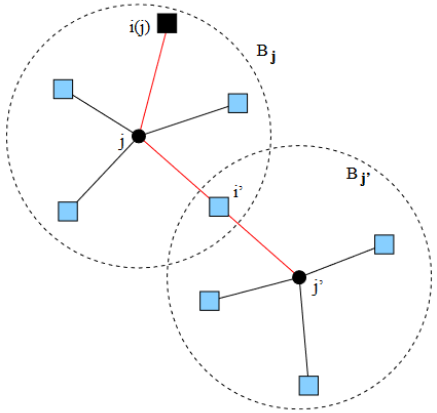
- 步骤1：ILP问题建模
- 步骤2：放松成LP问题
- 步骤3：从分数解到整数解：
  - Filtering：去除在c_{ij}比较大的时候对应的非0的分数解x_{ij}。
    - 步骤3.1：我们定义用户j的分数链接成本：$\Delta_j = \sum_{i \in V} c_{ij} x_{ij}$ 。
    - 步骤3.2：证明对于该LP问题得到的分数最优解(x, y)，必然存在另一个可行分数解(x', y')使得$F(x', y') \leq 2F(x, y)$, $C(x', y') \leq 2C(x, y)$且$\text{If } x'_{ij} > 0, \text{then } c_{ij} < 2\Delta_j$。说明这些可行分数解$(x', y')$ 确定的工厂更加“靠近”用户j。【证明略】
    - 步骤3.3：选择“靠近”用户j的工厂集合 $B_j = \{i : c_{ij} \leq 2\Delta_j\}$，将部分分数解转为0，同时满足原始问题的x约束进行归一化。

$$x'_{ij} = \begin{cases} \frac{x_{ij}}{\sum_{i \in B_j} x_{ij}} & \text{if } i \in B_j \\ 0 & \text{if } i \notin B_j \end{cases}$$

线性规划——Deterministic Rounding

Metric Uncapacitated Facility Location

- 步骤1：ILP问题建模
- 步骤2：放松成LP问题
- 步骤3：从分数解到整数解：
  - Filtering：对于分数解而言，去除在c_{ij}比较大的时候对应的非0的解 x_{ij}。
  - Rounding：尽量减少工厂的开张成本
    - 步骤3.4：首先排序用户分数链接成本$\{\Delta_j\}(j \in 1,\dots,n)$，选择最低成本的用户$j'$，再从Filtering中得到的工厂集合 $B_{j'} = \{i: c_{ij'} \leq 2\Delta_{j'}\}$ 中选取最小开张成本的工厂$i(j')$，如果这个工厂i(j')在另一个已经分配的任务的临近工厂圈B_j内，则不打开这个工厂$i(j')$，而是让用户j'与$i(j)$建立链接；
- 步骤4：证明近似比 $Z(\hat{x},\hat{y}) = F(\hat{x},\hat{y}) + C(\hat{x},\hat{y}) \leq 2F(x,y) + 6C(x,y) < 6Z(x,y).$



$$
\begin{aligned}
c_{j'i(j)} &\leq c_{j'i'} + c_{i'j} + c_{ji(j)} \\
&\leq 2\Delta_{j'} + 2\Delta_j + 2\Delta_j \\
&\leq 6\Delta_{j'}
\end{aligned}
$$

$$
\begin{aligned}
C(\hat{x},\hat{y}) &= \sum_{j \in V} c_{i(j)j} \\
&\leq \sum_{j \in V} 6\Delta_j \\
&\leq 6C(x,y)
\end{aligned}
$$

$$F(\hat{x},\hat{y}) \leq F(x',y') \leq 2F(x,y).$$

线性规划——Deterministic Rounding

- **Metric Uncapacitated Facility Location**
  - 只要将B_j 的定义做一些更改，可以得到4-approximation的近似算法【证明略】
  - 该问题的应用场景：
    - Edge Offload
    - 数据中心选址问题, 公共交通枢纽选址问题, 投票站选址问题
- 其他可以用Deterministic Rounding提出近似算法的问题
  - Minimum Cost Bipartite Matching
  - Scheduling on Unrelated Parallel Machines
  - Non-preemptive $1|r_j|\sum C_j$ Scheduling
  - …

线性规划——Randomized Rounding

- **Problem —— Max SAT**
  - 输入：n个Bool变量$x_1, \dots, x_i, \dots x_n$，m个从句$C_1, \dots, C_j, \dots C_m$，每个从句都是多个Bool变量和其反义变量的或式（如$x_3 \vee \overline{x_5} \vee x_{11}$），每个从句有权重$w_j$ 和长度l_j，从句满足（SAT）的条件是其中的一个子句为真
  - 输出：设置n个Bool变量的True/False，获得**最大**权重和的满足从句
- **Max SAT —— 等概率赋值近似算法**
  - 算法1：以1/2的概率为Bool变量赋值
  - 分析近似比：[非LP场景: 找到ALG优化目标的期望值和OPT的关系]
    - 定义一个随机变量Y_j 表示从句j是否SAT，则优化目标为$W = \sum_{j=1}^{m} w_j Y_j$
    - 优化目标的期望：$E[W] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[clause\ C_j\ satisfied]$
    - $\Pr[clause\ C_j\ satisfied] = \left(1 - \left(\frac{1}{2}\right)^{l_j}\right) \geq \frac{1}{2}$; $E[W] \geq \frac{1}{2}\sum_{j=1}^{m} w_j \geq \frac{1}{2} OPT$
    - l_j 越大，近似算法的近似比越好！

线性规划——Randomized Rounding

- **Max SAT —— 不等概率赋值近似算法**
  - 算法2：以p > 1/2的概率为Bool变量赋值为True
  - 分析近似比：[非LP场景: 找到ALG优化目标的期望值和OPT的关系]
    - 优化目标的期望：$E[W] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[clause\ C_j\ satisfied]$
    - 从句（$a + b = l_j \geq 2$）：$\Pr[clause\ C_j\ satisfied] = (1 - p^a(1-p)^b)$；因为 $p > \frac{1}{2} > 1 - p$，$\Pr[clause\ C_j\ satisfied] \geq 1 - p^{a+b} \geq 1 - p^{l_j} \geq 1 - p^2$
    - 从句（$l_j = 1$）：$\Pr[clause\ C_j\ satisfied] = \mathrm{p}$
    - 归纳：$E[W] \geq \min(p, 1-p^2) \sum_{j=1}^{m} w_j \geq \min(p, 1-p^2) OPT \geq 0.618 \cdot OPT$

线性规划——Randomized Rounding

Max SAT —— Randomized Rounding

- 算法3：
  - 核心思想：为每一个变量设置不同的概率
  - ILP构造：

$$\text{maximize} \quad \sum_{j=1}^{m} w_j z_j$$

$$\text{subject to} \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, \qquad \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i,$$

$$y_i \in \{0, 1\}, \qquad i = 1, \ldots, n,$$

$$z_j \in \{0, 1\}, \qquad j = 1, \ldots, m.$$

  - 放松到LP：

$$\text{maximize} \quad \sum_{j=1}^{m} w_j z_j$$

$$\text{subject to} \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j, \qquad \forall C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i,$$

$$0 \leq y_i \leq 1, \qquad i = 1, \ldots, n,$$

$$0 \leq z_j \leq 1, \qquad j = 1, \ldots, m.$$

线性规划——Randomized Rounding

Max SAT —— Randomized Rounding

- 分析近似比：[LP场景: 找到ALG优化目标的期望和LP最优解的关系]
  - 优化目标的期望：$E[W] = \sum_{j=1}^{m} w_j E[Y_j] = \sum_{j=1}^{m} w_j \Pr[clause\ C_j\ satisfied]$
  - $\Pr[clause\ C_j\ not\ satisfied] = \prod_{i \in P_j}(1 - y_i^*) \prod_{i \in N_j} y_i^*$ （算术-几何均值不等式）
  - $\leq \left[\frac{1}{l_j}\left(\sum_{i \in P_j}(1 - y_i^*) + \sum_{i \in N_j} y_i^*\right)\right]^{l_j} = \left[1 - \frac{1}{l_j}\left(\sum_{i \in P_j} y_i^* + \sum_{i \in N_j}(1 - y_i^*)\right)\right]^{l_j}$
  - $\leq \left[1 - \frac{z_j^*}{l_j}\right]^{l_j}$
  - $\Pr[clause\ C_j\ satisfied] \geq 1 - \left[1 - \frac{z_j^*}{l_j}\right]^{l_j} \geq \left[1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}\right] z_j^*$
  - $E[W] = \sum_{j=1}^{m} w_j \Pr[clause\ C_j\ satisfied] \geq \sum_{j=1}^{m} w_j z_j^* \left[1 - \left(1 - \frac{z_j^*}{l_j}\right)^{l_j}\right]$
  - $\geq \min_{k \geq 1}\left[1 - \left(1 - \frac{1}{k}\right)^k\right] \sum_{j=1}^{m} w_j z_j^* \geq \left(1 - \frac{1}{e}\right) OPT$

线性规划——Randomized Rounding

- **Max SAT —— Choosing better one**
  - 算法4：在算法1和算法3中选最好的一个结果执行
  - 分析近似比： [LP场景: 找到ALG优化目标的期望和LP最优解的关系]
    - $E[W] = E[\max(W_1, W_2)] \geq E\left[\frac{1}{2}W_1 + \frac{1}{2}W_2\right] = \frac{1}{2}E[W_1] + \frac{1}{2}E[W_2]$
    - $\geq \frac{1}{2}\sum_{j=1}^{m} w_j z_j^* \left[1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right] + \frac{1}{2}\sum_{j=1}^{m}\left(1 - \left(\frac{1}{2}\right)^{l_j}\right)$
    - $\geq \sum_{j=1}^{m} w_j z_j^* [\frac{1}{2}\left(1 - \left(1 - \frac{1}{l_j}\right)^{l_j}\right) + \frac{1}{2}(1 - 2^{-l_j})]$
    - $\geq \frac{3}{4} Z_{LP}^* \geq \frac{3}{4} OPT$
- **Max SAT —— Integrality Gap**
  - 假定场景：$x_1 \vee x_2$，$x_1 \vee \overline{x_2}$，$\overline{x_1} \vee x_2$，$\overline{x_1} \vee \overline{x_2}$，权重均为1
  - 利用LP可以得到一个可行解：$y_1 = y_2 = \frac{1}{2}$；$z_j = 1$；得到LP的优化目标为4，而实际问题最优解为3，IG为3/4。
  - 因此利用LP Relax无法获得比3/4-approx.更好的算法。

线性规划——Randomized Rounding

- **概率论在Randomized Rounding应用的重要理论**
  - 问题：对于给定的随机变量，和期望值相差给定距离的取值发生的概率是多少?
  - 三个结论：马尔可夫不等式、切比雪夫不等式和切诺夫界。
  - 特点：随着对随机变量的独立性的要求提高，这三个结论对于这个概率的估计也愈加准确。
- **马尔可夫不等式**
  - 若Z是非负、整数随机变量，则：$\forall\, a \in R^{+}: \Pr[Z \geq a] \leq \frac{E[Z]}{a}$
  - 基于马尔科夫不等式，可以得到和期望相差一定距离的随机变量的取值发生的概率的上限：$\Pr\left[Z \geq (1+\delta)E[Z]\right] \leq \frac{1}{1+\delta}$
  - 一个直观的例子：如果Z是工资，那么E(Z)就是平均工资，假设a=n*E(X)，即平均工资的n倍。那么根据马尔可夫不等式，不超过1/n的人会有超过平均工资的n倍的工资。

线性规划——Randomized Rounding

切比雪夫不等式

- 若有一组随机变量，包括：$X_1, \ldots, X_n \in \{0,1\}$ ，令$p_i = E[X_i] = \Pr[X_i = 1]$，$X = \sum_i X_i$ ，定义$\mu = E[X] = \sum_i p_i$ ， $\sigma = \sqrt{Var(X)}$ ，则有：

$$\Pr[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2}, k > 0$$

- 特点：比马尔可夫不等式更加精确

切诺夫界

- 特点：比前两个不等式更加精确
- 存在各种变形，应用广泛

$$\Pr[X > (1+\delta)\mu] \leq \frac{e^{(e^t-1)\mu}}{e^{(1+\delta)t\mu}}$$

$$\Pr[X > (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{3}} \quad (0 < \delta < 1)$$

$$\Pr[X < (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2}} \quad (0 < \delta < 1)$$

$$\Pr[X > (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2+\delta}} \quad (\delta \geq 0)$$

$$\Pr[X < (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu}{2+\delta}} \quad (\delta \geq 0)$$

线性规划——Randomized Rounding

- **Problem —— Network Congestion**
  - 输入：一个无向图$G=(V,E)$，给定一个顶点对集合$D=\{(s_i,t_i)\}_{i=1\ldots k}$，问题需要我们为每个顶点对流$(s_i,t_i)$确定连接链路$P_i$，且使得没有链路的负载过高。
  - 输出：使得最大拥塞链路的拥塞程度最小的分配方案。
- **切诺夫界在Randomized Rounding中的应用**
  - ILP构造：

$$\begin{aligned}
\text{minimize} \quad & C \\
\text{subject to} \quad & \sum_{P\in\mathcal{P}_i} f_P^i = 1 \qquad \forall i \\
& \sum_i \sum_{e\in P} f_P^i \le C \quad \forall e \in E, \\
& f_P^i \in \{0,1\} \quad \forall P \in \mathcal{P}_i, i = 1,\ldots,k
\end{aligned}$$

f_P^i 用于决策是否在流i中选择路径P；对于每条边e，计算经过这条边的路径P的数量，这个数量要比优化目标小

  - 放松到LP问题，在多项式时间内得到LP最优分数解：$f_P^{i*} \in [0,1]$
  - Randomized Rounding: 以LP最优分数解f_P^{i*}作为概率分布，从连接每个顶点对的可行链路中，随机选取一条链路$P \in \mathcal{P}_i$

线性规划——Randomized Rounding

切诺夫界在Randomized Rounding中的应用

- 分析近似比步骤1[获得每条边e的期望拥塞程度]：任何一个流i选择一条边e的概率为 $\sum_{\{P\in\mathcal{P}_i, e\in P\}} f_P^{i*}$ ，定义一个随机变量X_i^e，当流i选择了边e则该变量为1，那么对任何一个边e的拥塞程度表示为：$X^e = \sum_i X_i^e$，可以得到：
  - $E[X_i^e] = \sum_{\{P\in\mathcal{P}_i, e\in P\}} f_P^{i*}$；
  - $E[X^e] = E[\sum_i X_i^e] = \sum_i E[X_i^e] = \sum_i \sum_{\{P\in\mathcal{P}_i, e\in P\}} f_P^{i*} \le C_{LP}^* \le C_{IP}^*$
- 分析近似比步骤2 [得到ALG期望目标和LP最优分数解的关系]：
  - 核心思想：利用切诺夫界得到X^e 超过LP最优解一段距离的概率上界
  - 证明方式1：令$k = 1+\epsilon, \epsilon \in (0,1)$，则$\Pr[X_e > kC_{IP}^*] \le e^{-\frac{\epsilon^2 C}{3}}$，当$C \gg \log n$，可以直接证明ALG是(1+ε)-approx.
  - 证明方式2：令$k = 1+\epsilon, \epsilon \in (0,+\infty)$，则$\Pr[X_e > kC_{\mathrm{IP}}^*] \le e^{-\frac{\epsilon^2 C}{2+\epsilon}} \approx e^{-\epsilon C} (\epsilon \gg 2)$，想得到常数约束，需要保证$\epsilon = O(\log n)$，则可证明ALG是O(log n)-approx.
  - 证明方式3：使用另一种切诺夫界表示，可以证明ALG是$O\left(\frac{\log n}{\log\log n}\right)$ -approx

线性规划——Primal-Dual Method

- **用线性规划构造近似算法的难点：**
  - 证明对任何的输入实例I， 都有：$c\big(ALG(I)\big) \le \alpha c(LP^*), \forall\, I, \alpha \ge 1$
  - 很多时候很难直接找到ALG和LP最优整数解的关系
- **LP对偶问题：为LP原始问题提供一个下界**
  - 考虑一个最小化原始问题

$$\begin{array}{lll} \min \sum_j c_j x_j & & \text{subject to} \\ \sum_j a_{ij} x_j \ge b_i & \forall i \in [m] & \\ x_j \ge 0 & \forall j \in [n] & \end{array} \qquad \textbf{(Primal)}$$

  - 目标：找到 $c(Z_{LP}) = \sum_j c_j x_j \ge LB$
  - 从约束入手：$\sum_i y_i\big(\sum_j a_{ij} x_j\big) \ge \sum_i y_i b_i\,,\ y_i \ge 0$
  - 联系新约束和目标函数：
    - $\sum_i y_i b_i \le \sum_i y_i(\sum_j a_{ij} x_j) = \sum_j\big(\sum_i y_i a_{ij}\big)x_j \le \sum_j c_j x_j$
    - 需要引入新约束: $\sum_i y_i a_{ij} \le c_j\,, \forall\, j$

线性规划——Primal-Dual Method

- **LP对偶问题：为LP原始问题提供一个下界**
  - 得到对偶问题：

$$\begin{array}{lll} \max \sum_i y_i b_i & & \text{subject to} \\ \sum_i y_i a_{ij} \leq c_j & \forall j \in [n] & \\ y_i \geq 0 & \forall i \in [m] & \end{array} \qquad \textbf{(Dual)}$$

  - 对偶问题的一些性质
    - LP Monogamy: 原始问题和对偶问题相互对偶
    - Weak Duality: 对任何的原始对偶问题的解(x, y)，都有$y^T b < c^T x$ [下界]
    - Strong Duality: 若原问题和对偶问题中有任意一个存在有界的最优解，则另外一个存在相同的最优解。若原问题和对偶问题中有任意的最优解趋于无穷大，则另外一个不存在可行解。
    - Complementary Slackness: 原始对偶问题具有有界最优解(x, y)，当且仅当：
      - $x_j = 0 \; or \; \sum_i y_i a_{ij} = c_{ij} \; \forall j$ 且 $y_i = 0 \; or \; \sum_j x_j a_{ij} = b_{ij} \; \forall i$

线性规划——Primal-Dual Method

- **写LP对偶问题的一个简单例子**
  - 可能的复杂情况：各种求和，展开即可

① $\min\ -50x_1+20x_2$

$s.t.\quad 2x_1-x_2\geq -5 \quad y_1$

$3x_1+x_2\geq 3 \quad y_2$

$2x_1-3x_2\leq 12 \quad y_3$

$x_1,x_2\geq 0$

② $2y_1x_1-y_1x_2 \geq -5y_1$

$3y_2x_1+y_2x_2 \geq 3y_2$

$-2y_3x_1+3y_3x_2 \geq -12y_3$

$\max\ -5y_1+3y_2-12y_3$

③ $2y_1x_1-y_1x_2 \geq \cdots$

$3y_2x_1+y_2x_2 \geq \cdots$

$-2y_3x_1+3y_3x_2 \geq \cdots$

$2y_1+3y_2-2y_3\leq -50$

$-y_1+y_2+3y_3\leq 20$

④ $\max\ -5y_1+3y_2-12y_3$

$s.t.\quad 2y_1+3y_2-2y_3\leq -50$

$-y_1+y_2+3y_3\leq 20$

$y_1,y_2,y_3\geq 0$

线性规划——Primal-Dual Method

- **Metric Uncapacitated Facility Location**
  - 步骤1：ILP问题建模
  - 步骤2：放松成LP问题
  - 步骤3：获得对偶问题[用上页方法可以轻松做到]并在多项式时间内解这两个LP问题

$$\text{minimize} \quad \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{i \in F} x_{ij} = 1, && \forall j \in D, \\
& x_{ij} \leq y_i, && \forall i \in F, j \in D, \\
& x_{ij} \in \{0,1\}, && \forall i \in F, j \in D, \\
& y_i \in \{0,1\}, && \forall i \in F.
\end{aligned}$$

$$\text{maximize} \quad \sum_{j \in D} v_j$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{j \in D} w_{ij} \leq f_i, && \forall i \in F, \\
& v_j - w_{ij} \leq c_{ij}, && \forall i \in F, j \in D, \\
& w_{ij} \geq 0, && \forall i \in F, j \in D.
\end{aligned}$$

线性规划——Primal-Dual Method

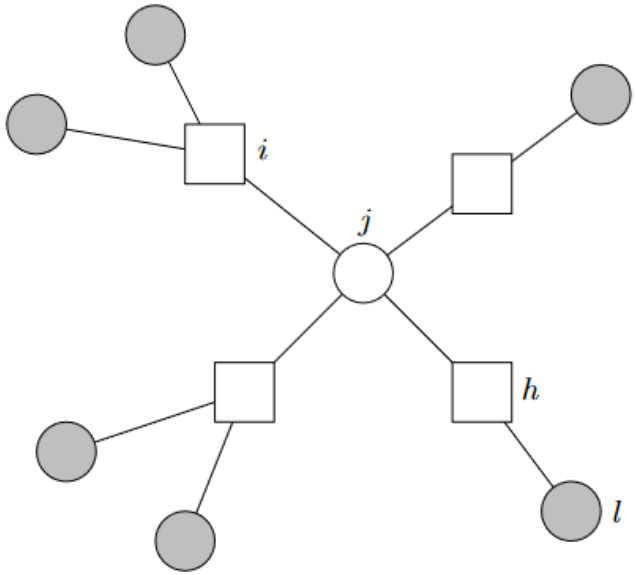
Metric Uncapacitated Facility Location

- 步骤4：[算法1——Deterministic Rounding]
  - 步骤4.1：定义客户j临近工厂集合$N(j) = \{i \in F: x_{ij}^* > 0\}$
  - 步骤4.2：
    - 根据Complementary Slackness性质，可以得到当$x_{ij}^* > 0$，则必然存在$v_j^* - w_j^* = c_{ij}$，此时$c_{ij} \leq v_j^*$；
    - 根据Weak Duality性质，可以得到$\sum_{j \in D} v_j^* \leq Z^* \leq OPT$ [完成为OPT找LB]
  - 步骤4.3：我们先研究单个客户j_k，如果打开临近工厂中最便宜的工厂i_k，可以获得下界：$f_{i_k} = f_{i_k} \sum_{i \in N(j_k)} x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i \, x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i \, y_i^*$，那么对需要开展的k个客户，可得：$\sum_k f_{i_k} \leq \sum_k \sum_{i \in N(j_k)} f_i y_i^* \leq \sum_{i \in F} f_i y_i^*$ [完成为ALG找与LP最优解的关系1]
  - 步骤4.4：定义客户j临近的所有工厂$N(j)$的临近客户端$N^2(j) = \{k \in D: client\ k\ neighbors\ some\ facility\ i \in N(j)\}$

线性规划——Primal-Dual Method

Metric Uncapacitated Facility Location

- 步骤4： [算法1——Deterministic Rounding]
  - 步骤4.5：为了进一步降低工厂开张的成本，可以尽量让客户端j_k选中的工厂i_k可以为该客户端临近的所有工厂$N(j_k)$的临近客户端$N^2(j_k)$服务，同时给一个性能界：$c_{il} \le c_{ij} + c_{hj} + c_{hl} \le v_j^* + v_j^* + v_l^*$。在迭代中，我们每次都优先选中$v^*$最小的客户。假如$j$比$l$先处理，则必然存在$v_j^* \le v_l^*$，故：$c_{il} \le 3v_l^*$；又因为：$\sum_{i\in F, j\in D} c_{ij}x_{ij} \le \sum_{l\in D} c_{il} \sum_{i\in F} x_{ij} \le \sum_{l\in D} 3v_l^* \le 3OPT$ [完成为ALG找与LP最优解的关系2]
- 步骤5：分析得到近似比为1+3=4



线性规划——Primal-Dual Method

- **Metric Uncapacitated Facility Location**
  - 步骤4： [算法1——Deterministic Rounding]
  - 步骤5：分析得到近似比为4

Solve LP, get optimal primal solution (x^*, y^*) and dual solution (v^*, w^*)
$C \leftarrow D$
$k \leftarrow 0$
while $C \neq \emptyset$ **do**
  $k \leftarrow k + 1$
  Choose $j_k \in C$ that minimizes v_j^* over all $j \in C$
  Choose $i_k \in N(j_k)$ to be the cheapest facility in $N(j_k)$
  Assign j_k and all unassigned clients in $N^2(j_k)$ to i_k
  $C \leftarrow C - \{j_k\} - N^2(j_k)$

线性规划——Primal-Dual Method

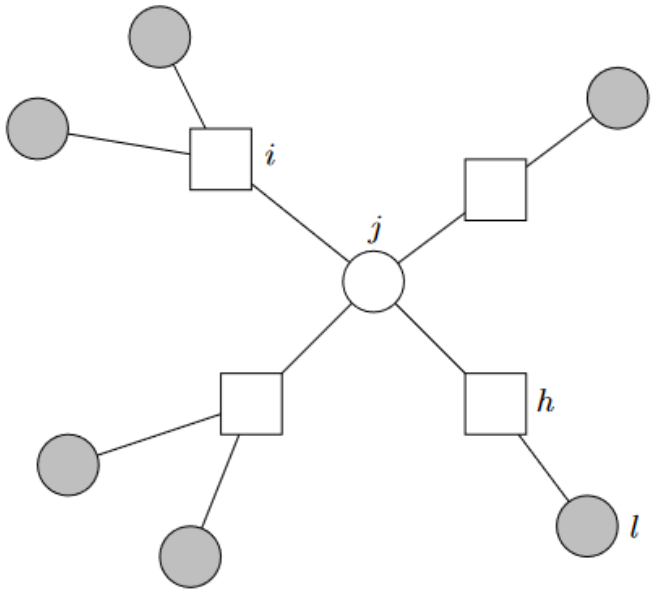
Metric Uncapacitated Facility Location

- 步骤4：[算法2——Randomized Rounding]
  - 步骤4.1：定义客户j临近工厂集合$N(j) = \{i \in F: x_{ij}^* > 0\}$
  - 步骤4.2：根据Complementary Slackness性质，可以得到当$x_{ij}^* > 0$，则必然存在$v_j^* - w_j^* = c_{ij}$，此时$c_{ij} \le v_j^*$；根据Weak Duality性质，可以得到$\sum_{j \in D} v_j^* \le Z^* \le OPT$ [完成为OPT找LB]
  - 步骤4.3：我们先研究单个客户j_k，如果以$x_{ij_k}^*$的概率打开临近的工厂i_k，可以获得工厂的期望开张成本：$\mathrm{E}[f_{i_k}] = \sum_{i \in N(j_k)} f_i \, x_{ij_k}^* \le \sum_{i \in N(j_k)} f_i \, y_i^*$，那么对需要开张的k个工厂，可得：$\mathrm{E}[\sum_k f_{i_k}] \le \sum_k \sum_{i \in N(j_k)} f_i y_i^* \le \sum_{i \in F} f_i y_i^*$ [完成为ALG找与LP最优解的关系1]
  - 步骤4.4：定义客户j临近的所有工厂$N(j)$的临近客户端$N^2(j) = \{k \in D: client\ k\ neighbors\ some\ facility\ i \in N(j)\}$；定义客户j连接到任意临近工厂的期望连接成本是$C_j^* = \sum_{i \in N(j)} c_{ij} x_{ij}^*$

线性规划——Primal-Dual Method

Metric Uncapacitated Facility Location

- 步骤4：[算法2——Randomized Rounding]
  - 步骤4.5：为了进一步降低工厂开张的成本，可以尽量让客户端j_k选中的工厂i_k可以为该客户端临近的所有工厂$N(j_k)$的临近客户端$N^2(j_k)$服务，同时给一个性能界：$c_{il} \le c_{ij} + c_{hj} + c_{hl} = \sum_{i \in N(j)} c_{ij} x_{ij}^* + c_{hj} + c_{hl} \le C_j^* + v_j^* + v_l^*$ 。在循环中，我们每次都优先选中$v^* + C^*$最小的客户。假如j比l先处理，则必然存在$v_j^* + C_j^* \le v_l^* + C_l^*$，故：$E\left[\sum_{i \in F, j \in D} c_{ij} x_{ij}\right] \le \sum_{j \in D} (2v_j^* + C_j^*)$
- 步骤5：分析近似比
  - $\mathrm{E}\left(\sum_{i \in F} f_i y_i^* + \sum_{i \in F, j \in D} c_{ij} x_{ij}\right)$
  - $\le \sum_{i \in F} f_i y_i^* + \sum_{j \in D} (2v_j^* + C_j^*)$
  - $\le \sum_{i \in F} f_i y_i^* + 2\sum_{j \in D} v_j^* + \sum_{i \in F, j \in D} c_{ij} x_{ij}^*$
  - $\le \ OPT + 2OPT = 3OPT$



线性规划——Primal-Dual Method

- **Metric Uncapacitated Facility Location**
  - 步骤4：[算法2——Randomized Rounding]
  - 步骤5：分析近似比

Solve LP, get optimal primal solution (x^*, y^*) and dual solution (v^*, w^*)
$C \leftarrow D$
$k \leftarrow 0$
while $C \neq \emptyset$ **do**
    $k \leftarrow k + 1$
    Choose $j_k \in C$ that minimizes $v_j^* + C_j^*$ over all $j \in C$
    Choose $i_k \in N(j_k)$ according to the probability distribution $x_{ij_k}^*$
    Assign j_k and all unassigned clients in $N^2(j_k)$ to i_k
    $C \leftarrow C - \{j_k\} - N^2(j_k)$

线性规划——Primal-Dual Method

- **Metric Uncapacitated Facility Location**
  - 步骤4：[算法3——Primal-Dual Method] [证明略]
  - 步骤5：分析近似比为3

$v \leftarrow 0, w \leftarrow 0$
$S \leftarrow D$
$T \leftarrow \emptyset$
while $S \neq \emptyset$ **do** // While not all clients neighbor a facility in T
  Increase v_j for all $j \in S$ and w_{ij} for all $i \in N(j), j \in S$ uniformly until some $j \in S$ neighbors some $i \in T$ or some $i \notin T$ has a tight dual inequality
  if some $j \in S$ neighbors some $i \in T$ **then**
    $S \leftarrow S - \{j\}$
  if $i \notin T$ has a tight dual inequality **then**
    // facility i is added to T
    $T \leftarrow T \cup \{i\}$
    $S \leftarrow S - N(i)$
$T' \leftarrow \emptyset$
while $T \neq \emptyset$ **do**
  Pick $i \in T$; $T' \leftarrow T' \cup \{i\}$
  // remove all facilities h if some client j contributes to h and i
  $T \leftarrow T - \{h \in T : \exists j \in D, w_{ij} > 0 \text{ and } w_{hj} > 0\}$

大纲

1. 近似算法基本设计步骤
2. 近似算法中线性规划法的分类和应用
3. **在线算法的简单介绍**
4. 未来研究方向和现有问题

Online Algorithm

- 在线问题 vs. 离线问题
  - 在线问题：决策时未掌握全部实例信息，已做的决策在更多信息呈现后不可更改。
  - 离线问题：实例在决策前全部已知的问题。
- 在线算法
  - 以序列化的形式一个个处理输入，被迫做出的选择可能会被证明不是最优的，即使每一步做出局部最优选择，也有可能累计陷入非最优解。
  - 例子：插入排序
- 竞争比
  - 记S是在线问题的输入序列，算法ALG应对S产生的成本记为c(ALG)，假设有一个全知全能的离线算法，它应对S产生的成本记为c(OPT)，对于任何的S，加入 $c(ALG) \leq \alpha\ c(OPT)$，则称ALG为$\alpha$竞争算法。

Online Algorithm

- 固有下界
  - 下界通常是在线问题的固有属性，与解决它的具体算法无关，在线问题存在下界的主要原因是实例信息的不完全性。
  - 存在（或已经找到）非平凡下界的离线算法只有一部分，但是在线算法的下界普遍存在，因此通常存在竞争比的界
  - 研究在线问题的目标：
    - 找到更大的下界，使得下界更靠近OPT
    - 找到更好的算法，使得竞争比更靠近下界约束的竞争比的界
- 证明竞争比的步骤[和证明近似比的步骤基本一致]
  - 对任何的输入实例I，找到OPT cost的下界(Lower Bound，LB)：$LB(I) \leq c\big(OPT(I)\big), \forall I$
  - 证明对任何的输入实例I，都有：$c\big(ALG(I)\big) \leq \alpha LB(I), \forall I, \alpha \geq 1$
  - 推断出结论：$c(ALG) \leq \alpha LB \leq \alpha c(OPT)$

大纲

1. 近似算法基本设计步骤
2. 近似算法中线性规划法的分类和应用
3. 在线算法的简单介绍
4. **未来研究方向和现有问题**

未来研究方向

- 在线优化算法 & 在线学习
  - 在任务调度的实际系统中，信息很难完全已知，是一个在线问题。
  - 传统的离线学习难以处理迅速增长的数据量和特征数量，可以在线迭代调整算法，减少训练需要的数据量
- 现有近似算法/在线优化算法在复杂新场景中的应用
  - 在任务调度领域，近似算法/在线优化算法仍然是很常见的算法
- 近似算法/在线优化算法作为论文中的一部分，增加工作量
  - 很多启发式算法的某些部分需要一些数学做点缀，可以考虑用近似算法
- 找到或整理目前现有的近似算法/在线优化算法的详细整理文档
  - 这个工作很耗时但很重要

现有问题

- 近似算法只能保证最差时候的性能下界，但是这个保证也是一个低层次的保证
  - 在Relaxed-LP场景中，Integrality Gap限制了性能
  - 近似算法可能带来n倍于最优解的性能差距，这在实际系统中通常难以被接受
- 近似算法的泛化能力差
  - 近似算法严重依赖场景的设计和问题的构造
  - 为了证明近似比，通常近似算法的改造能力很差，很难迁移到其他领域

现有问题

- “提出一个理论证明完善的近似算法/在线优化算法”就是一个NP类问题
  - NP类问题：不能在多项式时间内解决或不确定能不能在多项式时间内解决，但能在多项式时间验证的问题
  - 看论文验证证明过程：啊对对对 vs. 自己解决写证明：？？？？
  - 自己写完证明：啊对对对 vs. 跑起实验：？？？？
- 学习内容非常多且应用时不成体系
  - 凸优化
  - 组合优化
  - 在线优化
  - ……

学习资料

- Approximation Algorithms
  - CMU 15-854: Approximation Algorithms
  - CMU 15-854(B): Advanced Approximation Algorithms
  - Umich IOE 2713: Approximation & Online Algorithms
  - Williamson D P, Shmoys D B. The design of approximation algorithms[M]. Cambridge university press, 2011.
- Online Algorithms
  - UofT CSC2421: Topics in Algorithms: Online and other Myopic Algorithms
  - Nguyen T K. Primal-Dual Approaches in Online Algorithms, Algorithmic Game Theory and Online Learning[D]. Université Paris Sorbonne, 2019.

学习资料

Optimization Online

- Categories – Optimization Online (optimization-online.org)

- Applications – OR and Management Sciences (1,556)
  - Airline Optimization (31)
  - Finance and Economics (185)
  - Marketing (14)
  - Production and Logistics (151)
  - Scheduling (215)
  - Supply Chain Management (82)
  - Telecommunications (109)
  - Transportation (295)
  - Yield Management (16)
- Applications – Science and Engineering (1,209)
  - Basic Sciences Applications (79)
  - Biomedical Applications (99)
  - Chemical Engineering (29)
  - Civil and Environmental Engineering (28)
  - Control Applications (142)
  - Data-Mining (155)
  - Facility Planning and Design (78)
  - Mechanical Engineering (43)
  - Multidisciplinary Design Optimization (33)
  - Optimization of Systems modeled by PDEs (61)
  - Smart Grids (42)
  - Statistics (179)
  - VLSI layout (10)
- Integer Programming (1,692)
  - (Mixed) Integer Linear Programming (591)
  - (Mixed) Integer Nonlinear Programming (483)
  - 0-1 Programming (265)
  - Cutting Plane Approaches (280)
- Linear, Cone and Semidefinite Programming (1,409)
  - Cone Programming (5)
  - Linear Programming (290)
  - Second-Order Cone Programming (107)
  - Semi-definite Programming (554)
- Network Optimization (271)
- Nonlinear Optimization (2,089)
  - Bound-constrained Optimization (78)
  - Constrained Nonlinear Optimization (658)
  - Nonlinear Systems and Least-Squares (104)
  - Quadratic Programming (261)
  - Systems governed by Differential Equations Optimization (121)
  - Unconstrained Optimization (329)
- Optimization in Data Science (14)
  - Data Science Algorithms (1)
  - Data Science Applications (1)
  - Data Science Theory (3)

学习资料

Hailiang Zhao’ s Blog

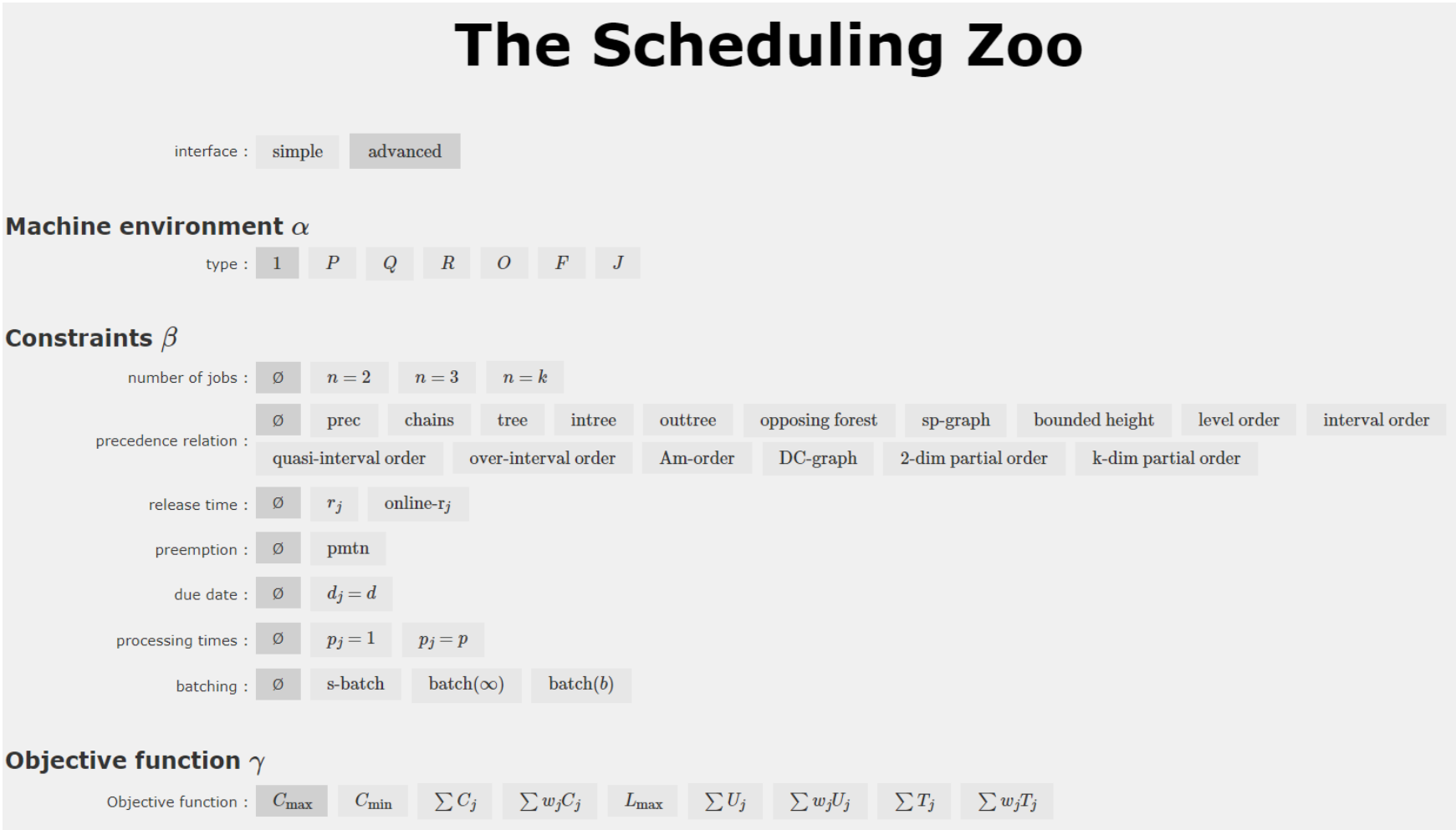
- Hailiang Zhao @ ZJU.CS.CCNT (hliangzhao.me)

  - **ADMM for Stochastic Optimization** [ADMM part 8]
    This slide demonstrates how to use ADMM to solve stochastic optimization problems, Oct 2022.
  - **ADMM for Distributed Optimization** [ADMM part 7]
    This slide demonstrates how to use ADMM to solve centralized and decentralized distributed optimization problems, Oct 2022.
  - **ADMM for Nonconvex Optimization** [ADMM part 6]
    This slide demonstrates how to use ADMM to solve nonconvex problems, Oct 2022.
  - **ADMM: The Variational Inequality Perspective** [ADMM part 5]
    This slide demonstrates how to cast ADMM into the framework of variational inequality, Oct 2022.
  - **ADMM for Nonlinear Convex Problems** [ADMM part 4]
    This slide demonstrates how to solve general nonlinear convex problems with ADMM, Oct 2022.
  - **ADMM for Multi-Block Problems** [ADMM part 3]
    This slide demonstrates how to solve multi-block problems with ADMM, Oct 2022.
  - **ADMM, Linearized ADMM, Accelerated Linearized ADMM and Their Convergence Analysis** [ADMM part 2]
    This slide summaries the iteration steps of ADMM, Linearized ADMM, Accelerated Linearized ADMM, and their convergen rate, Oct 2022.
  - **Distributed Systems in One Slide**
    This slide summaries the key contents of an interesting online book, *Distributed Systems for Fun and Profit*, Oct 2022.
  - **From Dual Descent to ADMM** [ADMM part 1]
    This slide demonstrates the design details of Dual Descent, Method of Multipliers, and the vanilla version of ADMM, Oct 2022.
  - **Preliminaries for Optimization Algorithm Design and Analysis**
    This slide demonstrates the mathematical preliminaries of optimization algorithms that should be kept in mind firmly, Oct 2022.

学习资料

Scheduling Zoo

- [The scheduling zoo (lip6.fr)]



中山大學
SUN YAT-SEN UNIVERSITY

谢 谢 各 位 老 师 指 正

中山大学计算机学院