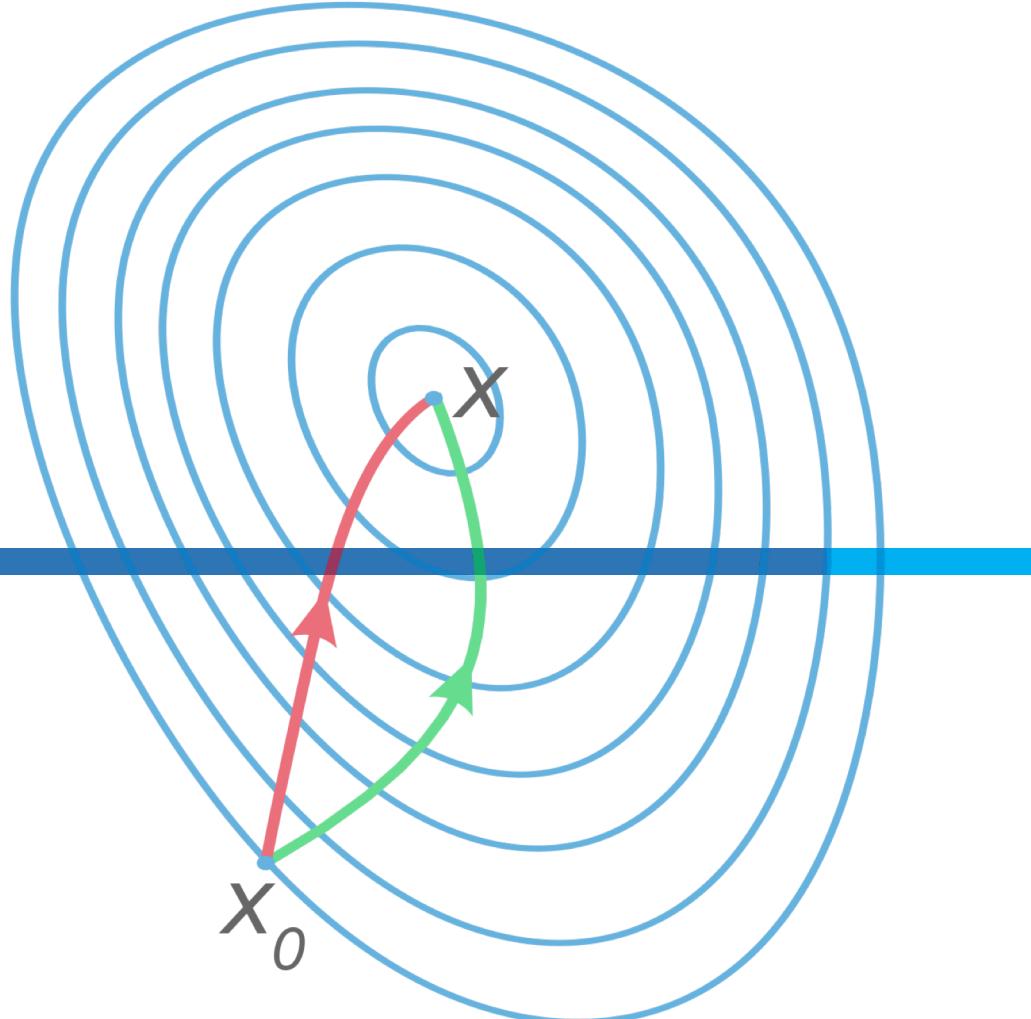


最优化方法

第四周

计算机学院
余皓然

2023/3/16



一种简单的内点法（续）

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、**内点法**；

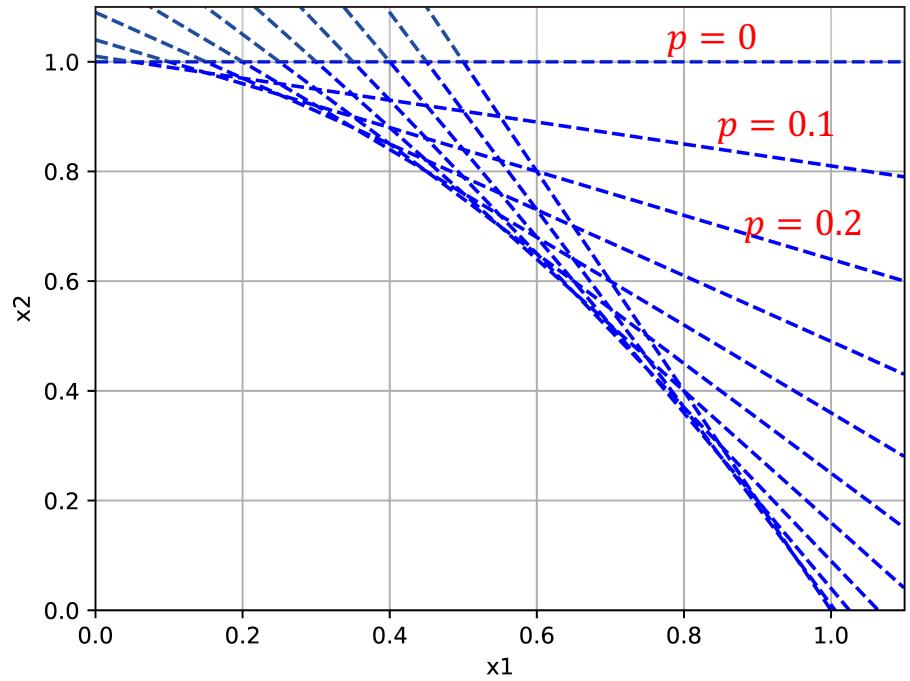
有约束凸优化问题、KKT条件、内点法

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

例子

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.t.} \quad & 2px_1 + x_2 \leq p^2 + 1, \forall p \in [0, 0.1, \dots, 1], \\ \text{var.} \quad & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$



例子

内点法（牛顿法计算每个 t 下的解）

$$\begin{aligned} \min \quad & t \mathbf{c}^T \mathbf{x} - \sum_{i=1}^m \log(-\sum_{j=1}^n A_{ij} x_j + b_i) \\ \text{var. } & \mathbf{x}. \end{aligned}$$

$$x^{(k)} = x^{(k-1)} - (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}),$$

设置：

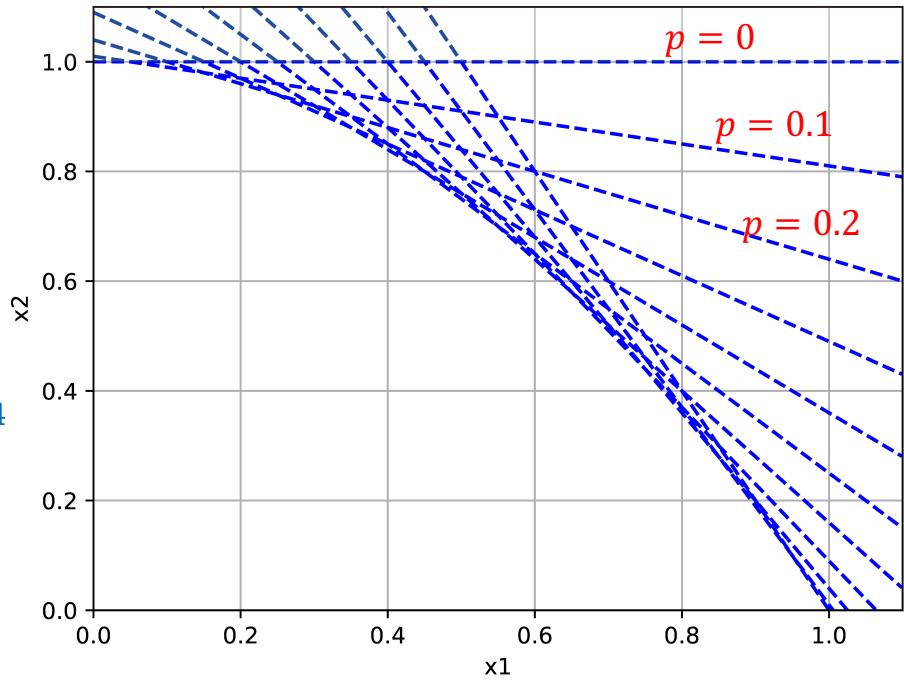
初始解 $(10^{-4}, 10^{-4})$

初始 $t=1$

参数 $\mu=15$ t更新规则改为 $t \leftarrow t + \mu$

外迭代终止条件 相邻 t 的两个解距离 $< 10^{-4}$

内迭代终止条件 相邻两个解距离 $< 10^{-4}$



例子

内点法（牛顿法计算每个 t 下的解）

$$\begin{aligned} \min \quad & t \mathbf{c}^T \mathbf{x} - \sum_{i=1}^m \log(-\sum_{j=1}^n A_{ij} x_j + b_i) \\ \text{var. } \quad & \mathbf{x}. \end{aligned}$$

$$x^{(k)} = x^{(k-1)} - (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}),$$

设置：

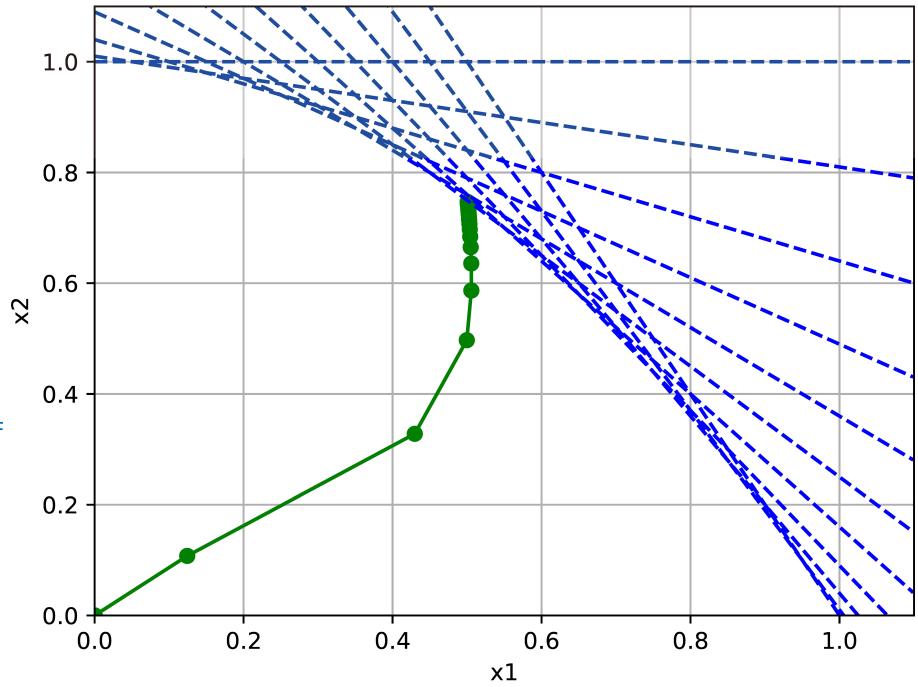
初始解 $(10^{-4}, 10^{-4})$

初始 $t=1$

参数 $\mu=15$ t 更新规则改为 $t \leftarrow t + \mu$

外迭代终止条件 相邻 t 的两个解距离 $< 10^{-4}$

内迭代终止条件 相邻两个解距离 $< 10^{-4}$



例子

内点法（梯度下降法计算每个 t 下的解）

$$\begin{aligned} \min \quad & t \mathbf{c}^T \mathbf{x} - \sum_{i=1}^m \log(-\sum_{j=1}^n A_{ij} x_j + b_i) \\ \text{var.} \quad & \mathbf{x}. \end{aligned}$$

$$x^{(k)} = x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}),$$

设置：

初始解 $(10^{-4}, 10^{-4})$

初始 $t=1$

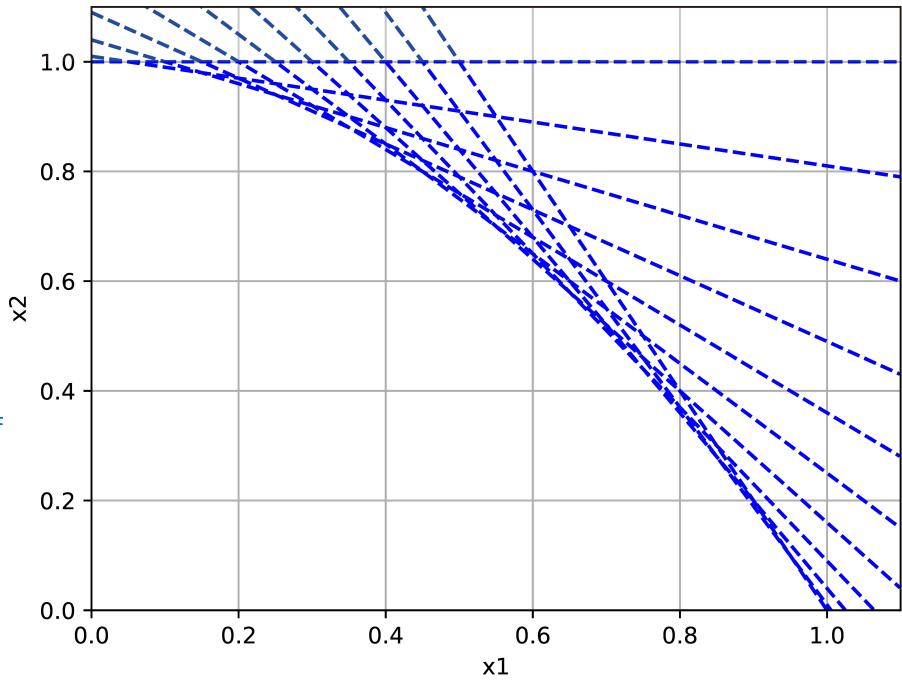
参数 $\mu=15$ t更新规则改为 $t \leftarrow t + \mu$

外迭代终止条件 相邻 t 的两个解距离 $< 10^{-4}$

内迭代终止条件 相邻两个解距离 $< 10^{-4}$

或超5000次

参数 $\alpha=10^{-5}$



例子

内点法（梯度下降法计算每个 t 下的解）

$$\begin{aligned} \min \quad & t \mathbf{c}^T \mathbf{x} - \sum_{i=1}^m \log\left(-\sum_{j=1}^n A_{ij} x_j + b_i\right) \\ \text{var. } & \mathbf{x}. \end{aligned}$$

$$x^{(k)} = x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}),$$

设置：

初始解 $(10^{-4}, 10^{-4})$

初始 $t=1$

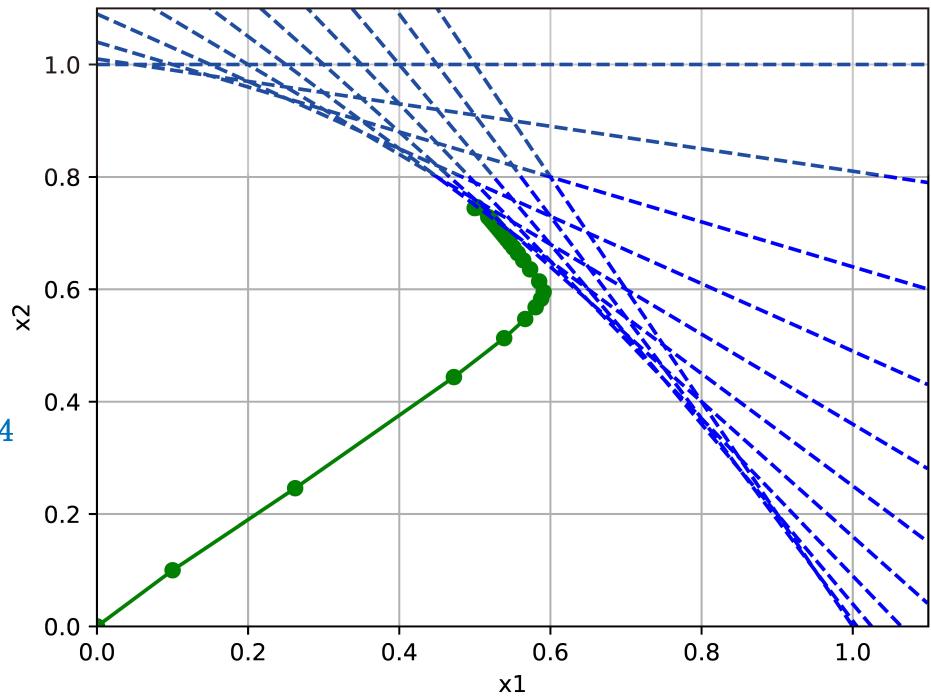
参数 $\mu=15$ t更新规则改为 $t \leftarrow t + \mu$

外迭代终止条件 相邻t的两个解距离 $< 10^{-4}$

内迭代终止条件 相邻两个解距离 $< 10^{-4}$

或超5000次

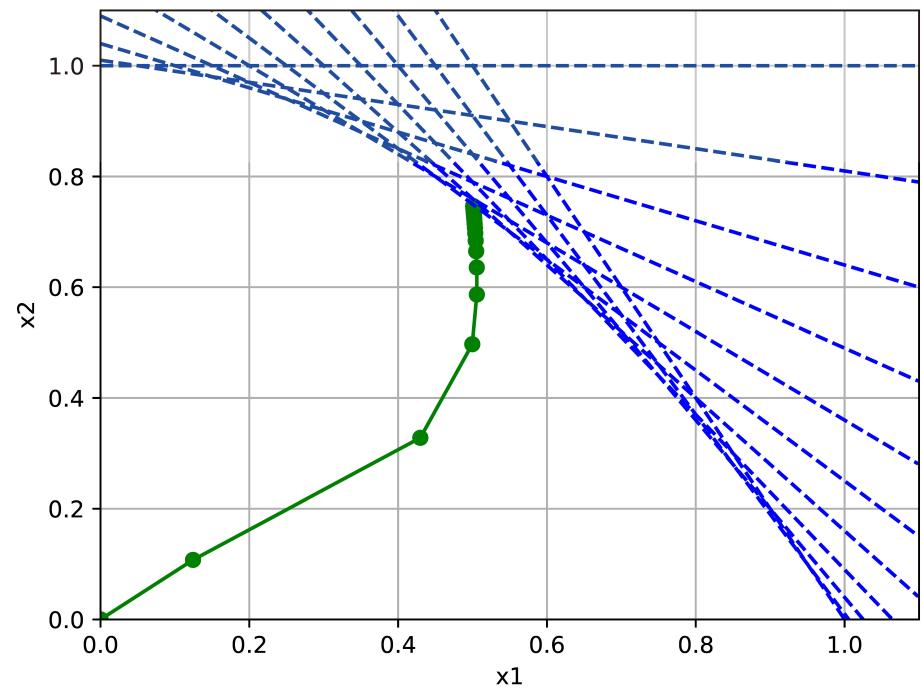
参数 $\alpha=10^{-5}$



例子

内点法（牛顿法计算每个 t 下的解）

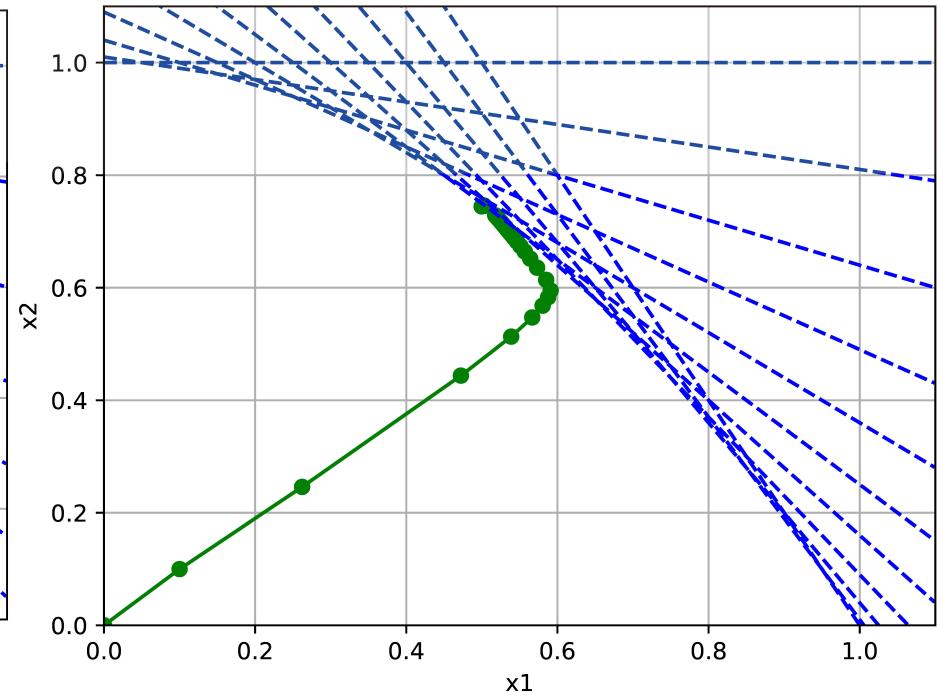
$$x^{(k)} = x^{(k-1)} - (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}),$$



0.011s

内点法（梯度下降法计算每个 t 下的解）

$$x^{(k)} = x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}),$$



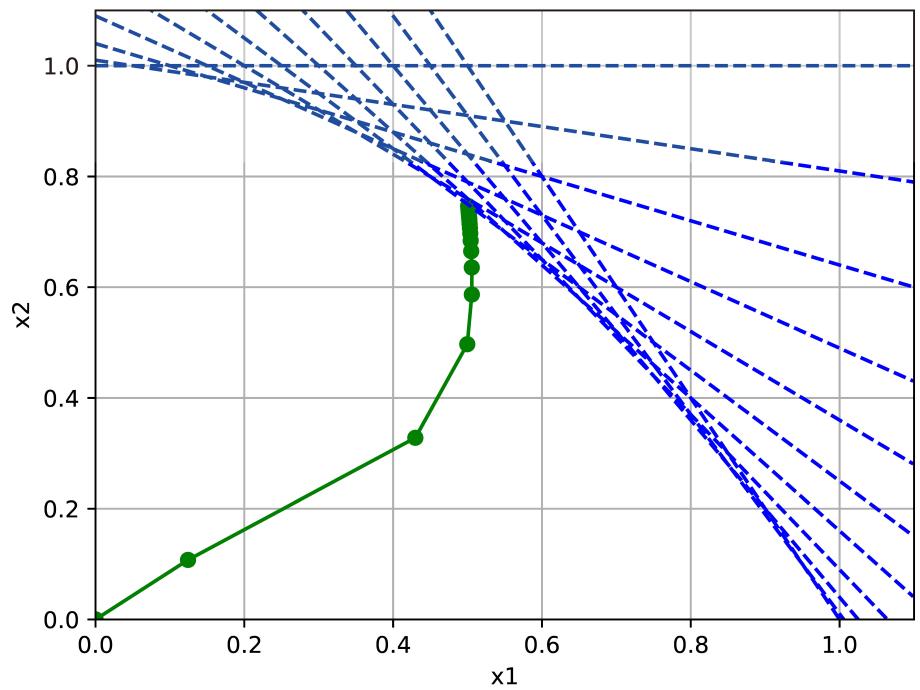
1.707s

同样 t 值下 $x^*(t)$ 应该相等，因为内迭代终止条件的影响，左右两图结果不同
若将内迭代终止条件改为“相邻两个解距离 $< 10^{-5}$ ”，左右图结果将更相似

例子

内点法（牛顿法计算每个 t 下的解）

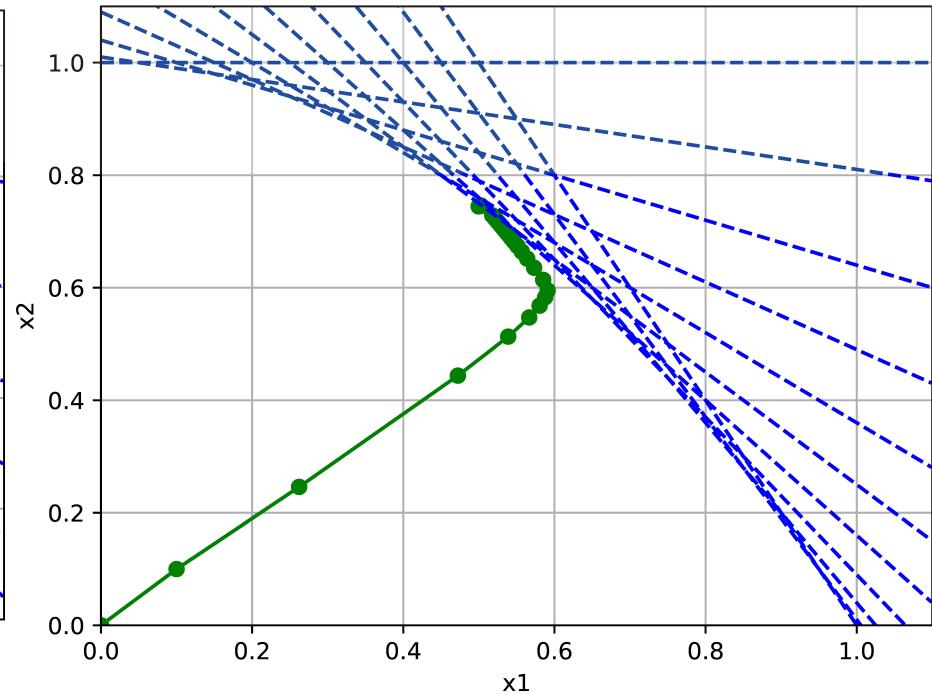
$$x^{(k)} = x^{(k-1)} - (\nabla^2 f(x^{(k-1)}))^{-1} \nabla f(x^{(k-1)}),$$



0.011s

内点法（梯度下降法计算每个 t 下的解）

$$x^{(k)} = x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}),$$

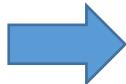


1.707s

线性规划

最优化问题的一般形式

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{D}. \end{aligned}$$



线性规划问题

$$\begin{aligned} \min \quad & \boldsymbol{a}_0^T \boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{a}_i^T \boldsymbol{x} + b_i \leq 0, i = 1, \dots, m, \\ & \boldsymbol{d}_j^T \boldsymbol{x} + c_j = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x}. \end{aligned}$$

当目标和约束都是线性时

很多有约束的最优化问题不属于线性规划

有约束凸优化问题

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、内点法；

有约束凸优化问题、KKT条件、内点法

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

有约束凸优化

将约束条件写入定义域后，定义域是凸集

无约束凸优化问题

$$\begin{aligned} & \min f(\boldsymbol{x}) \\ & \text{var. } \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中， $f(\boldsymbol{x})$ 是凸函数

当 $f(\boldsymbol{x})$ 一阶/二阶连续可微时，
可用梯度下降/牛顿法等求解

有约束凸优化问题

$$\begin{aligned} & \min f(\boldsymbol{x}) \\ & \text{var. } \boldsymbol{x} \in \mathcal{C}. \end{aligned}$$

其中， $f(\boldsymbol{x})$ 是凸函数， \mathcal{C} 是凸集合

有约束凸优化

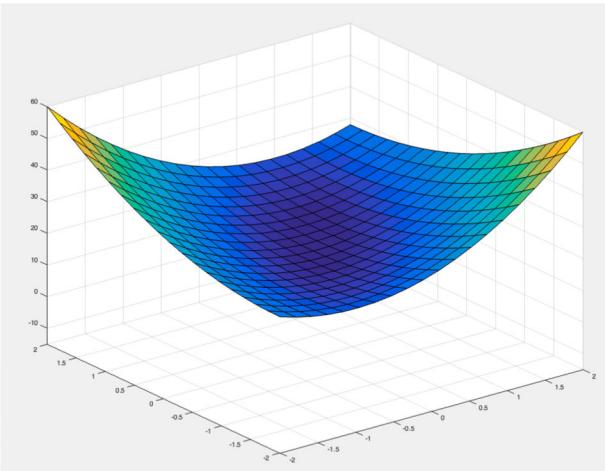
将约束条件写入定义域后，定义域是凸集

无约束凸优化问题

$$\begin{aligned} & \min f(x) \\ & \text{var. } x \in \mathcal{R}^n. \end{aligned}$$

其中， $f(x)$ 是凸函数

当 $f(x)$ 一阶/二阶连续可微时，
可用梯度下降/牛顿法等求解

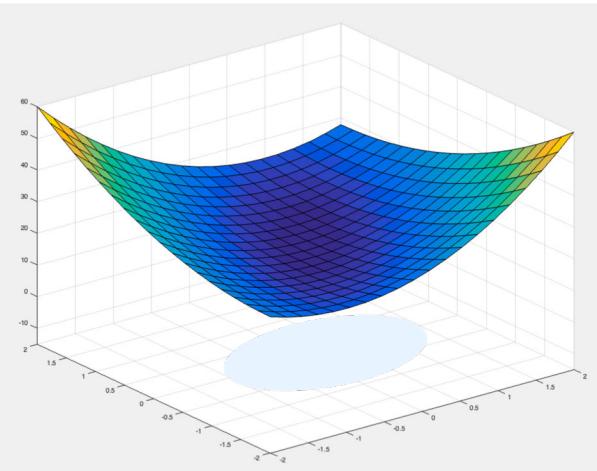


对 x 取值范围没有限制

有约束凸优化问题

$$\begin{aligned} & \min f(x) \\ & \text{var. } x \in \mathcal{C}. \end{aligned}$$

其中， $f(x)$ 是凸函数， \mathcal{C} 是凸集合



限制 x 从蓝色区域取值

有约束凸优化

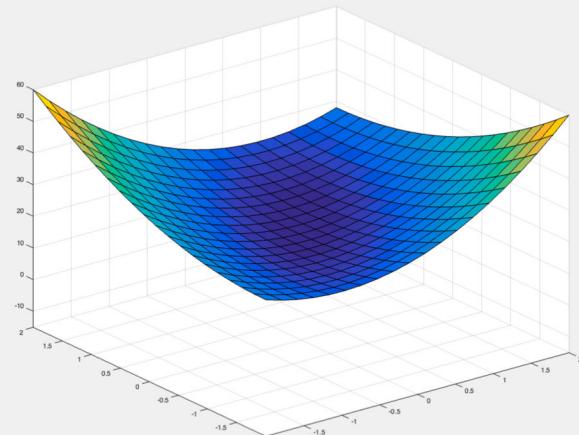
将约束条件写入定义域后，定义域是凸集

无约束凸优化问题

$$\begin{aligned} \min \quad & f(x) \\ \text{var. } \quad & x \in \mathcal{R}^n. \end{aligned}$$

其中， $f(x)$ 是凸函数

当 $f(x)$ 一阶/二阶连续可微时，
可用梯度下降/牛顿法等求解

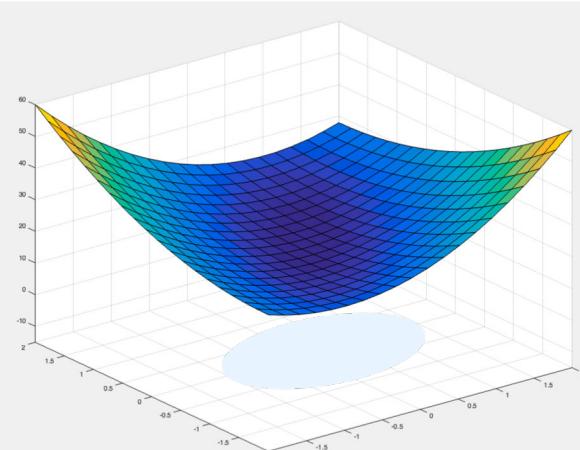


对 x 取值范围没有限制

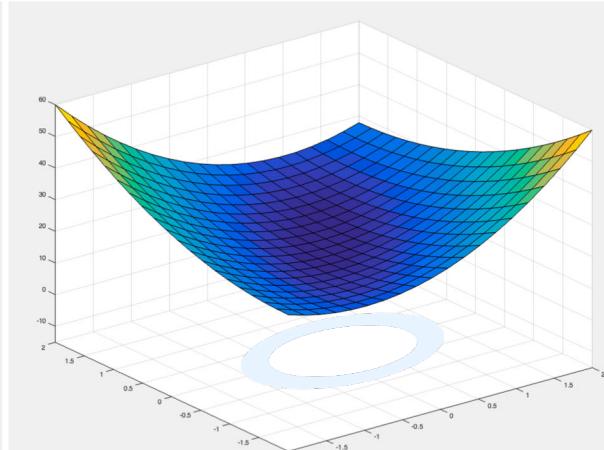
有约束凸优化问题

$$\begin{aligned} \min \quad & f(x) \\ \text{var. } \quad & x \in \mathcal{C}. \end{aligned}$$

其中， $f(x)$ 是凸函数， \mathcal{C} 是凸集合



限制 x 从蓝色区域取值



非凸集合时

有约束凸优化

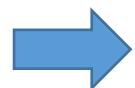
一般可整理为

有约束凸优化问题

$$\min f(\mathbf{x})$$

$$\text{var. } \mathbf{x} \in \mathcal{C}.$$

其中, $f(\mathbf{x})$ 是凸函数, \mathcal{C} 是凸集合



有约束凸优化问题的标准形式

$$\min f(\mathbf{x})$$

$$\text{s.t. } g_i(\mathbf{x}) \leq 0, i = 1, \dots, m, \\ h_j(\mathbf{x}) = 0, j = 1, \dots, p,$$

$$\text{var. } \mathbf{x} \in \mathbb{R}^n.$$

其中, $f(\mathbf{x}), g_i(\mathbf{x})$ 是凸函数, $h_j(\mathbf{x})$ 是仿射函数

仿射函数: 可写成 $\mathbf{a}_j^T \mathbf{x} + b_j$ 的形式

有约束凸优化

一般可整理为

有约束凸优化问题

$$\min f(\mathbf{x})$$

$$\text{var. } \mathbf{x} \in \mathcal{C}.$$

其中, $f(\mathbf{x})$ 是凸函数, \mathcal{C} 是凸集合



有约束凸优化问题的标准形式

$$\min f(\mathbf{x})$$

$$\text{s.t. } g_i(\mathbf{x}) \leq 0, i = 1, \dots, m, \\ h_j(\mathbf{x}) = 0, j = 1, \dots, p,$$

$$\text{var. } \mathbf{x} \in \mathbb{R}^n.$$

其中, $f(\mathbf{x}), g_i(\mathbf{x})$ 是凸函数, $h_j(\mathbf{x})$ 是仿射函数

仿射函数: 可写成 $\mathbf{a}_j^T \mathbf{x} + b_j$ 的形式

容易证明 满足 $g_i(\mathbf{x}) \leq 0$ 及 $h_j(\mathbf{x}) = 0$ 的集合是凸集 (构造 $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}$, 再根据凸集定义证明)

注意 $h_j(\mathbf{x})$ 需要是仿射函数, 若仅是凸函数, 约束条件不一定构成凸集

(考虑 $h_j(\mathbf{x}) = 0 \Leftrightarrow h_j(\mathbf{x}) \leq 0, -h_j(\mathbf{x}) \leq 0$)

有约束凸优化

有约束凸优化问题的标准形式

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathbb{R}^n. \end{aligned}$$

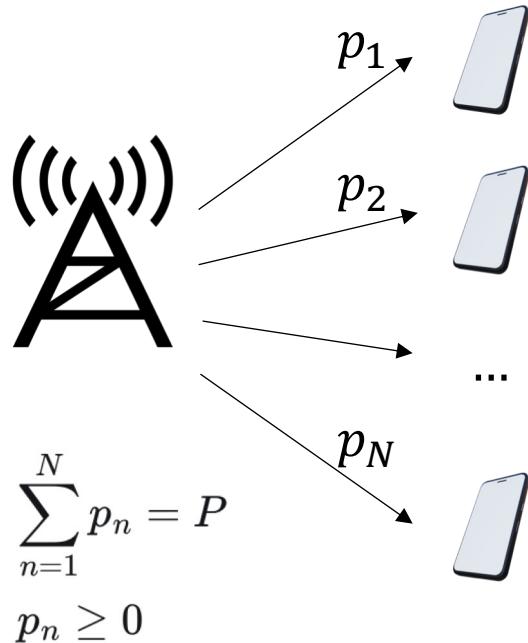
其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是凸函数, $h_j(\boldsymbol{x})$ 是仿射函数

包含各种特例: 无约束凸优化问题、线性规划问题等

对于这些特例, 已有多项式时间复杂度的求解算法

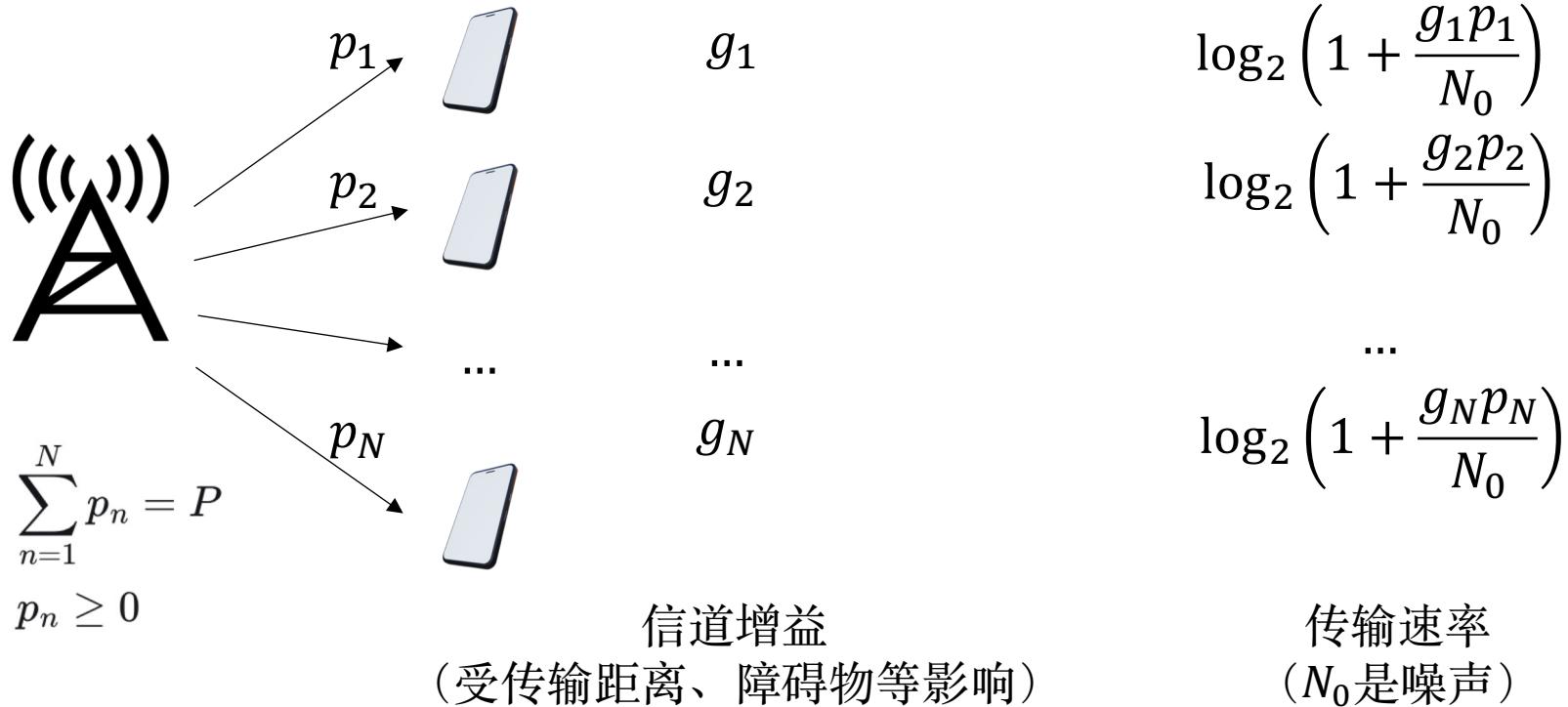
发射功率分配问题

移动通信基站如何分配用于传输数据的功率，从而最大化总传输速率



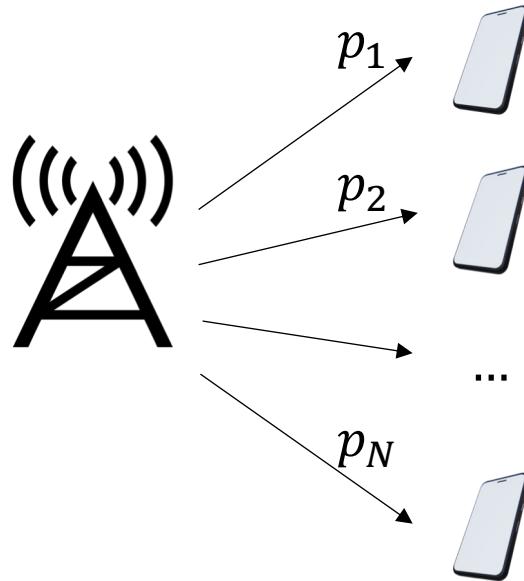
发射功率分配问题

移动通信基站如何分配用于传输数据的功率，从而最大化总传输速率



发射功率分配问题

移动通信基站如何分配用于传输数据的功率，从而最大化总传输速率



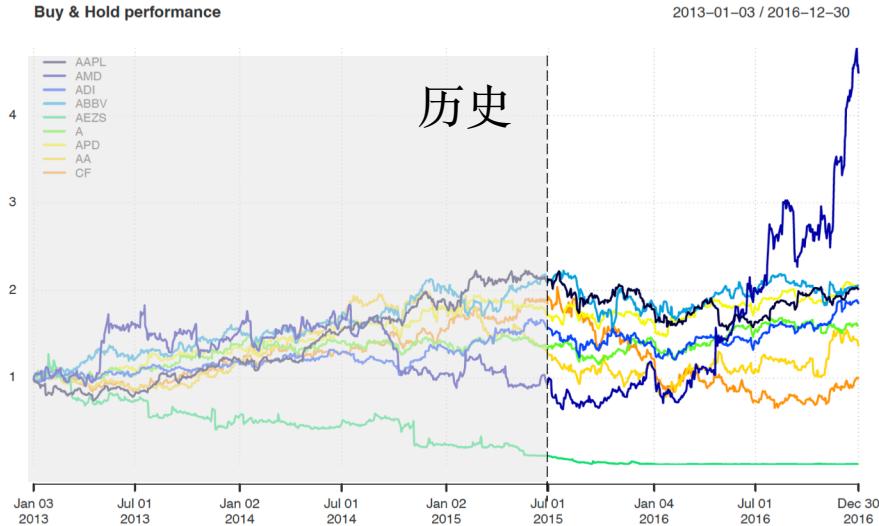
$$\begin{aligned} \max_{p_n} \quad & \sum_{n=1}^N \log_2 \left(1 + \frac{g_n p_n}{N_0} \right) \\ \text{s.t.} \quad & \sum_{n=1}^N p_n = P \\ & p_n \geq 0 \end{aligned}$$

是有约束的凸优化问题
(改写成最小化问题、分析海森矩阵)

* 在实际通信领域，考虑时延约束下的动态功率分配、多基站通信等

投资组合优化问题

如何对不同资产进行投资，从而最大化收益的期望、最小化风险（方差）



投资策略记为 $w = (w_1, w_2, \dots, w_N)^T$

将资产投资收益向量建模成随机向量，向量的期望和协方差矩阵记为 μ 和 Σ （根据历史得到）

投资组合优化问题

如何对不同资产进行投资，从而最大化收益的期望、最小化风险（方差）

The idea of **Markowitz's mean-variance portfolio (MVP)** (Markowitz 1952) is to find a trade-off between the expected return $\mathbf{w}^T \boldsymbol{\mu}$ and the risk of the portfolio measured by the variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$:

$$\begin{aligned} & \underset{\mathbf{w}}{\text{maximize}} && \mathbf{w}^T \boldsymbol{\mu} - \lambda \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w} \\ & \text{subject to} && \mathbf{1}^T \mathbf{w} = 1 \quad \boxed{\mathbf{w} \geq 0} \end{aligned}$$

where $\mathbf{w}^T \mathbf{1} = 1$ is the capital budget constraint and λ is a parameter that controls how risk-averse the investor is.

是有约束的凸优化问题
(改写成最小化问题、随机向量的协方差矩阵一定是半正定)

$$\boldsymbol{\Sigma} = \text{var}(\mathbf{r}) = \mathbb{E} \left[(\mathbf{r} - \mathbb{E}(\mathbf{r})) (\mathbf{r} - \mathbb{E}(\mathbf{r}))^T \right] \geq 0$$

KKT条件

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、内点法；

有约束凸优化问题、**KKT条件**、内点法

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

KKT条件

(一定条件下) 凸优化问题的全局最优解的充要条件：KKT条件

KKT (Karush-Kuhn-Tucker) 条件



William Karush , 1987



Harold Kuhn and Albert Tucker, 1980

The KKT conditions were originally named after [Harold W. Kuhn](#) and [Albert W. Tucker](#), who first published the conditions in 1951.^[2] Later scholars discovered that the necessary conditions for this problem had been stated by [William Karush](#) in his master's thesis in 1939.^{[3][4]}

KKT条件

$$\begin{array}{ll}\min & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} & \boldsymbol{x} \in \mathcal{R}^n.\end{array}$$

其中， $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是凸函数， $h_j(\boldsymbol{x})$ 是仿射函数

定理

对前述凸优化问题，若 $f(\boldsymbol{x})$ 和 $g_i(\boldsymbol{x})$ 都是连续可微函数且Slater条件成立，那么 \boldsymbol{x}^* 是全局最优解的充要条件是 \boldsymbol{x}^* 满足：

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\boldsymbol{x}^*) = \mathbf{0},$$

$$g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\boldsymbol{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, i = 1, \dots, m.$$

其中， λ_i^*, ν_j^* 是对偶问题的最优解。

KKT条件

$$\begin{array}{ll}\min & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} & \boldsymbol{x} \in \mathcal{R}^n.\end{array}$$

其中， $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是凸函数， $h_j(\boldsymbol{x})$ 是仿射函数

定理

对前述凸优化问题，若 $f(\boldsymbol{x})$ 和 $g_i(\boldsymbol{x})$ 都是连续可微函数且Slater条件成立，那么 \boldsymbol{x}^* 是全局最优解的充要条件是 \boldsymbol{x}^* 满足：



$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\boldsymbol{x}^*) = \mathbf{0}, \quad \text{稳定性条件}$$

$$g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\boldsymbol{x}^*) = 0, j = 1, \dots, p, \quad \text{原问题可行条件}$$

$$\lambda_i^* \geq 0, i = 1, \dots, m, \quad \text{对偶问题可行条件}$$

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, i = 1, \dots, m. \quad \text{互补松弛条件}$$

其中， λ_i^*, ν_j^* 是对偶问题的最优解。

类似于《高等数学》中拉格朗日乘子法中的乘子

KKT条件

$$\begin{array}{ll}\min & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} & \boldsymbol{x} \in \mathcal{R}^n.\end{array}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是凸函数, $h_j(\boldsymbol{x})$ 是仿射函数

定理

对前述凸优化问题, 若 $f(\boldsymbol{x})$ 和 $g_i(\boldsymbol{x})$ 都是连续可微函数且Slater条件成立,
那么 \boldsymbol{x}^* 是全局最优解的充要条件是 \boldsymbol{x}^* 满足:

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\boldsymbol{x}^*) = \mathbf{0},$$

对比无约束下的一阶条件
额外考虑约束条件的影响

$$g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\boldsymbol{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

考慮 $\min f(\boldsymbol{x}) + \lambda_i g_i(\boldsymbol{x})$

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, i = 1, \dots, m.$$

考慮 $g_i(\boldsymbol{x}^*) < 0$ 和 $g_i(\boldsymbol{x}^*) = 0$ 两种情况

其中, λ_i^*, ν_j^* 是对偶问题的最优解。

Slater条件

$$\begin{aligned} & \min f(\boldsymbol{x}) \\ \text{s.t. } & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var. } & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中， $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是凸函数， $h_j(\boldsymbol{x})$ 是仿射函数

Slater条件成立：存在 \boldsymbol{x} 使 $g_i(\boldsymbol{x}) < 0, i = 1, \dots, m$ 且 $h_j(\boldsymbol{x}) = 0, j = 1, \dots, p$.

(即存在 \boldsymbol{x} 处于可行域的相对内部)

以下凸优化问题不满足Slater条件，最优解不满足KKT条件：

$$\begin{aligned} & \min \boldsymbol{x} \\ \text{s.t. } & \boldsymbol{x}^2 \leq 0, \\ \text{var. } & \boldsymbol{x} \in \mathcal{R}. \end{aligned}$$

易见 $\boldsymbol{x}^* = 0$ ，而KKT条件（稳定性条件）要求 $1 + \lambda^* \cdot 2 \cdot 0 = 0$

例子

$$\begin{aligned} \min \quad & x_2 - x_1 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \end{aligned}$$

var. $\mathbf{x} \in \mathcal{R}$.

(1) 判断是凸优化问题且Slater条件成立

(2) 列出KKT条件

(3) 分析 $\lambda_i^* > 0$ 或 $\lambda_i^* = 0$ 的可能性

a. 若 $\lambda_1^* = \lambda_2^* = 0$, 稳定性条件不满足

b. 若 $\lambda_1^* = 0, \lambda_2^* \neq 0$, 稳定性条件不满足

c. 若 $\lambda_1^* \neq 0, \lambda_2^* = 0$, 推出 $x_1^* = \sqrt{2}, x_2^* = -\sqrt{2}$, 原问题可行条件不满足

d. 若 $\lambda_1^* \neq 0, \lambda_2^* \neq 0$, 推出 $x_1^* = \sqrt{3}, x_2^* = -1, \lambda_1^* = \frac{\sqrt{3}+1}{8}, \lambda_2^* = \frac{\sqrt{3}-1}{4}$, 满足KKT条件

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) = \mathbf{0},$$

$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

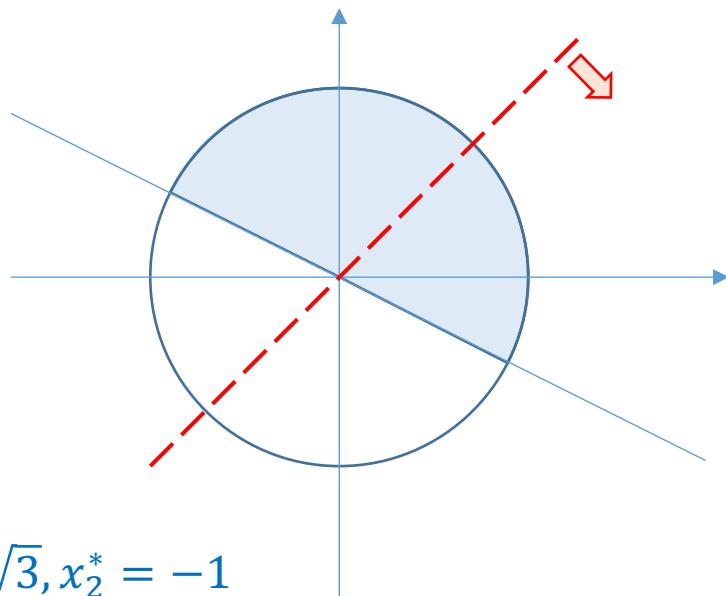
$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, m.$$

例子

$$\begin{aligned} \min \quad & x_2 - x_1 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \\ \text{var.} \quad & \mathbf{x} \in \mathcal{R}. \end{aligned}$$

$$\text{稳定性条件 } \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} = \mathbf{0} \text{ 即 } \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} \quad \lambda_1^* \geq 0, \lambda_2^* \geq 0$$

可视化:



$$x_1^* = \sqrt{3}, x_2^* = -1$$

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) = \mathbf{0},$$

$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

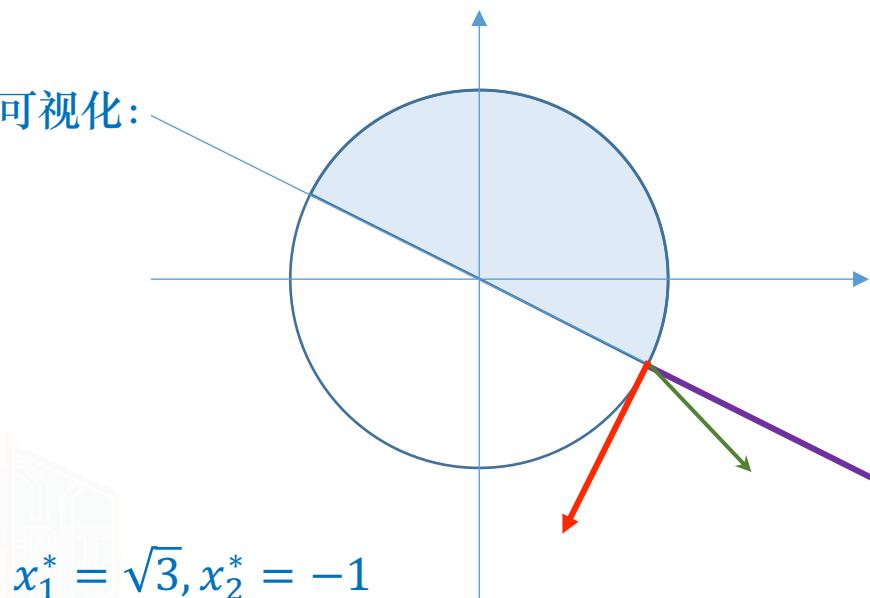
$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, m.$$

例子

$$\begin{aligned} \min \quad & x_2 - x_1 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \\ \text{var.} \quad & \mathbf{x} \in \mathcal{R}. \end{aligned}$$

稳定性条件 $\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} = \mathbf{0}$ 即 $\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix}$ $\lambda_1^* \geq 0, \lambda_2^* \geq 0$

可视化:



该点同时激活了两个不等式约束，
有 $\lambda_1^* \geq 0, \lambda_2^* \geq 0$ ，即绿色向量的方向
需要在红色和紫色之间

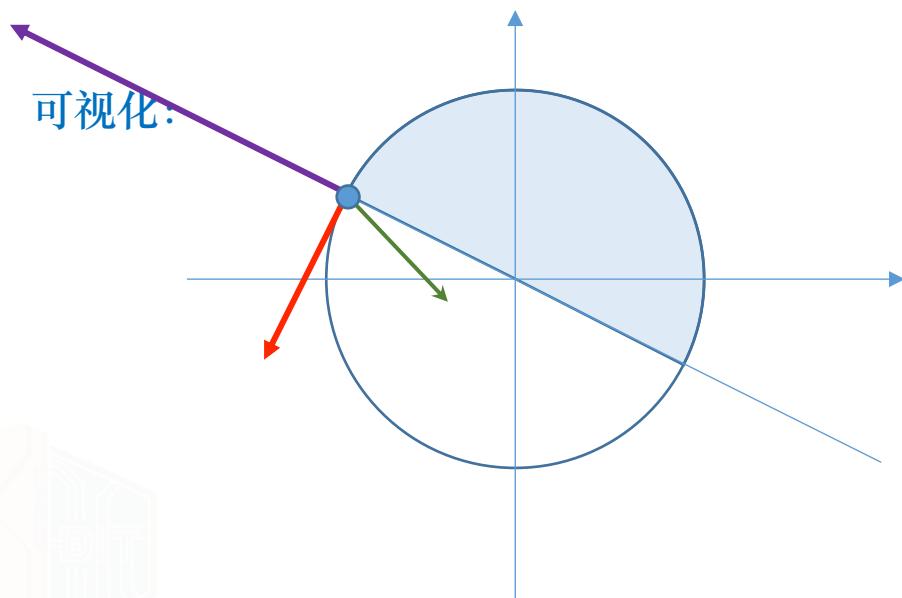
绿色向量: 目标函数 $f(\mathbf{x})$ 下降方向
紫色向量: $g_1(\mathbf{x})$ 上升方向
红色向量: $g_2(\mathbf{x})$ 上升方向

例子

$$\begin{aligned} \min \quad & x_2 - x_1 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \\ \text{var.} \quad & \mathbf{x} \in \mathcal{R}. \end{aligned}$$

$$\begin{aligned} \nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) &= \mathbf{0}, \\ g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) &= 0, j = 1, \dots, p, \\ \lambda_i^* \geq 0, i = 1, \dots, m, \\ \lambda_i^* g_i(\mathbf{x}^*) &= 0, i = 1, \dots, m. \end{aligned}$$

稳定性条件 $\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} = \mathbf{0}$ 即 $\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix}$ $\lambda_1^* \geq 0, \lambda_2^* \geq 0$



考虑蓝色点，绿色向量的方向无法在红色和紫色之间

故该点不是最优解

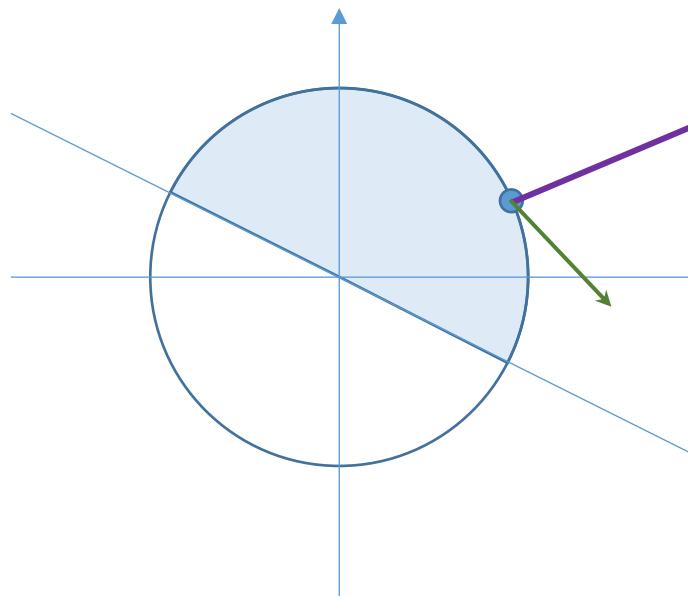
例子

$$\begin{array}{ll}\min & x_2 - x_1 \\ \text{s.t.} & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \\ \text{var.} & \mathbf{x} \in \mathcal{R}.\end{array}$$

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) = \mathbf{0},$$
$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p,$$
$$\lambda_i^* \geq 0, i = 1, \dots, m,$$
$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, m.$$

稳定性条件 $\begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} = \mathbf{0}$ 即 $\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix}$ $\lambda_1^* \geq 0, \lambda_2^* \geq 0$

可视化：



若蓝色点是最优解： $\lambda_2^* = 0$, 绿色向量和紫色向量的方向无法相同

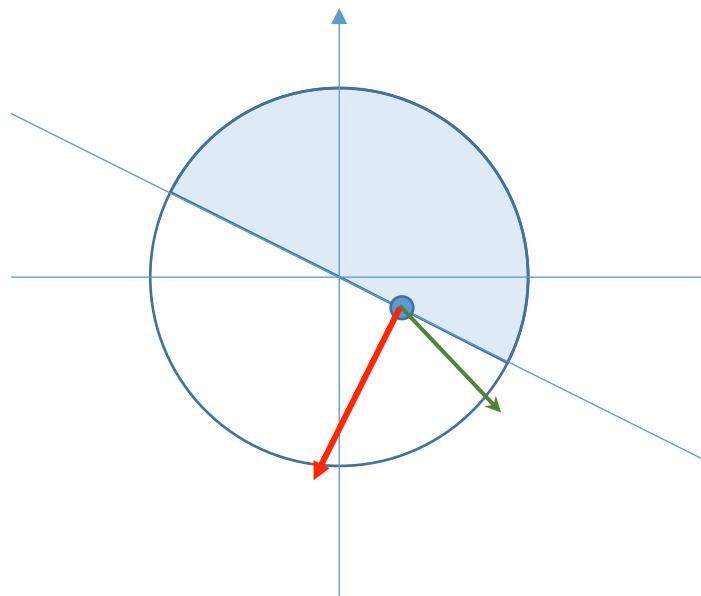
故该点不是最优解

例子

$$\begin{array}{ll}\min & x_2 - x_1 \\ \text{s.t.} & x_1^2 + x_2^2 - 4 \leq 0, \\ & -x_1 - \sqrt{3}x_2 \leq 0, \\ \text{var.} & \mathbf{x} \in \mathcal{R}.\end{array}$$

$$\text{稳定性条件 } \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} = \mathbf{0} \text{ 即 } \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \lambda_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + \lambda_2^* \begin{bmatrix} -1 \\ -\sqrt{3} \end{bmatrix} \quad \lambda_1^* \geq 0, \lambda_2^* \geq 0$$

可视化:



$$\begin{aligned}\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) &= \mathbf{0}, \\ g_i(\mathbf{x}^*) \leq 0, i &= 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p, \\ \lambda_i^* \geq 0, i &= 1, \dots, m, \\ \lambda_i^* g_i(\mathbf{x}^*) &= 0, i = 1, \dots, m.\end{aligned}$$

若蓝色点是最优解: $\lambda_1^* = 0$, 绿色向量和红色向量的方向无法相同

故该点不是最优解

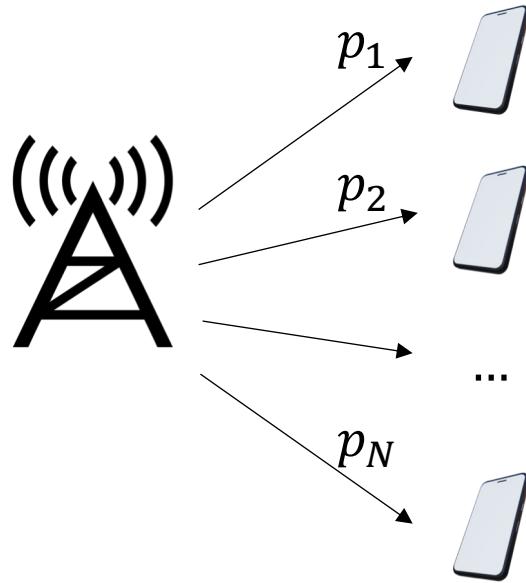
发射功率分配问题

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\boldsymbol{x}^*) = \mathbf{0},$$

$$g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\boldsymbol{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0, i = 1, \dots, m.$$



$$\max_{p_n} \quad \sum_{n=1}^N \log_2 \left(1 + \frac{g_n p_n}{N_0} \right)$$

$$\text{s.t.} \quad \sum_{n=1}^N p_n = P$$

$$p_n \geq 0$$

发射功率分配问题

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) = \mathbf{0},$$

$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, m.$$

KKT条件：

$$-\frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}} - \lambda_n^* + \nu^* = 0, n = 1, \dots, N,$$

$$p_n^* \geq 0, n = 1, \dots, N, \sum_{n=1}^N p_n^* = P,$$

$$\lambda_n^* \geq 0, n = 1, \dots, N,$$

$$-\lambda_n^* p_n^* = 0, n = 1, \dots, N.$$

发射功率分配问题

KKT条件：

$$-\frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}} - \lambda_n^* + \nu^* = 0, \quad n = 1, \dots, N,$$

$$p_n^* \geq 0, \quad n = 1, \dots, N, \quad \sum_{n=1}^N p_n^* = P,$$

$$\lambda_n^* \geq 0, \quad n = 1, \dots, N,$$

$$-\lambda_n^* p_n^* = 0, \quad n = 1, \dots, N.$$

把 λ_n^* 消去

$$p_n^* \geq 0, \quad n = 1, \dots, N, \quad \sum_{n=1}^N p_n^* = P,$$

$$\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}} \geq 0, \quad n = 1, \dots, N,$$

$$-\left(\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}}\right) p_n^* = 0, \quad n = 1, \dots, N.$$

发射功率分配问题

$$p_n^* \geq 0, n = 1, \dots, N, \sum_{n=1}^N p_n^* = P,$$

$$\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}} \geq 0, n = 1, \dots, N,$$

$$-\left(\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}}\right) p_n^* = 0, n = 1, \dots, N.$$

讨论 ν^* 与 $\frac{g_n}{N_0}$ 的关系

若 $\nu^* < \frac{g_n}{N_0}$: p_n^* 需要大于0, 即 $\nu^* = \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}}$

若 $\nu^* \geq \frac{g_n}{N_0}$: p_n^* 需要等于0

总结: $p_n^* = \begin{cases} \frac{1}{\nu^*} - \frac{N_0}{g_n}, & \text{若 } \nu^* < \frac{g_n}{N_0}, \\ 0, & \text{若 } \nu^* \geq \frac{g_n}{N_0}. \end{cases}$

发射功率分配问题

$$p_n^* \geq 0, n = 1, \dots, N, \sum_{n=1}^N p_n^* = P,$$

$$\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}} \geq 0, n = 1, \dots, N,$$

$$-\left(\nu^* - \frac{\frac{g_n}{N_0}}{1 + \frac{g_n p_n^*}{N_0}}\right) p_n^* = 0, n = 1, \dots, N.$$

$$p_n^* = \begin{cases} \frac{1}{\nu^*} - \frac{N_0}{g_n}, & \text{若 } \nu^* < \frac{g_n}{N_0}, \\ 0, & \text{若 } \nu^* \geq \frac{g_n}{N_0}. \end{cases}$$

等价于: $p_n^* = \left(\frac{1}{\nu^*} - \frac{N_0}{g_n}\right)^+$

$$\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n}\right)^+ = P$$

发射功率分配问题

$$\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = P$$

左侧随 ν^* 单调减，且 $\nu^* = \frac{\max_n\{g_n\}}{N_0}$ 时 $\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = 0$

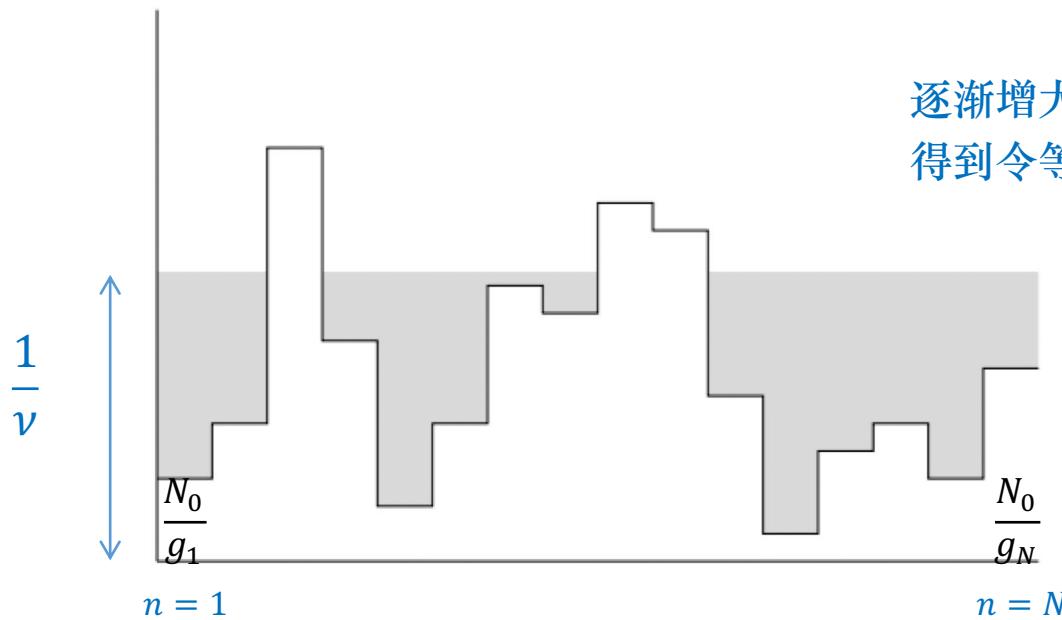
发射功率分配问题

$$\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = P$$

左侧随 ν^* 单调减，且 $\nu^* = \frac{\max_n\{g_n\}}{N_0}$ 时 $\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = 0$

注水算法

逐渐增大 $\frac{1}{\nu}$ ，直到“水”面积等于 P
得到令等式成立的 ν^*

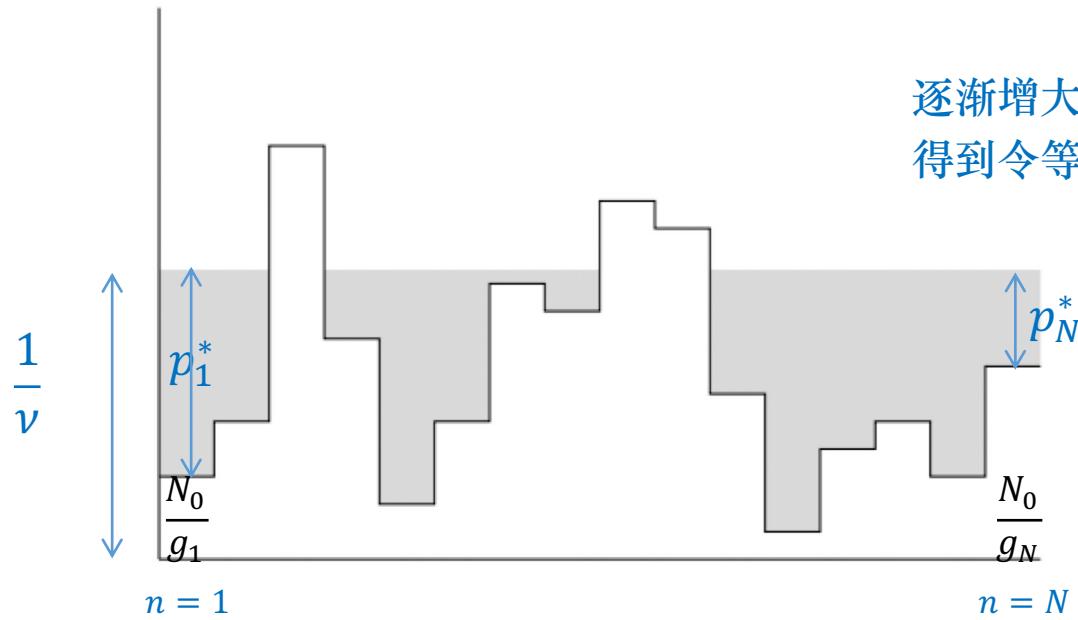


发射功率分配问题

$$\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = P$$

左侧随 ν^* 单调减，且 $\nu^* = \frac{\max_n\{g_n\}}{N_0}$ 时 $\sum_{n=1}^N \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+ = 0$

注水算法

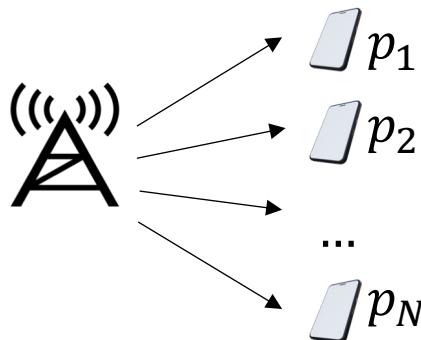


逐渐增大 $\frac{1}{\nu}$ ，直到“水”面积等于 P
得到令等式成立的 ν^*

$$p_n^* = \left(\frac{1}{\nu^*} - \frac{N_0}{g_n} \right)^+$$

每条水柱高度即最优功率

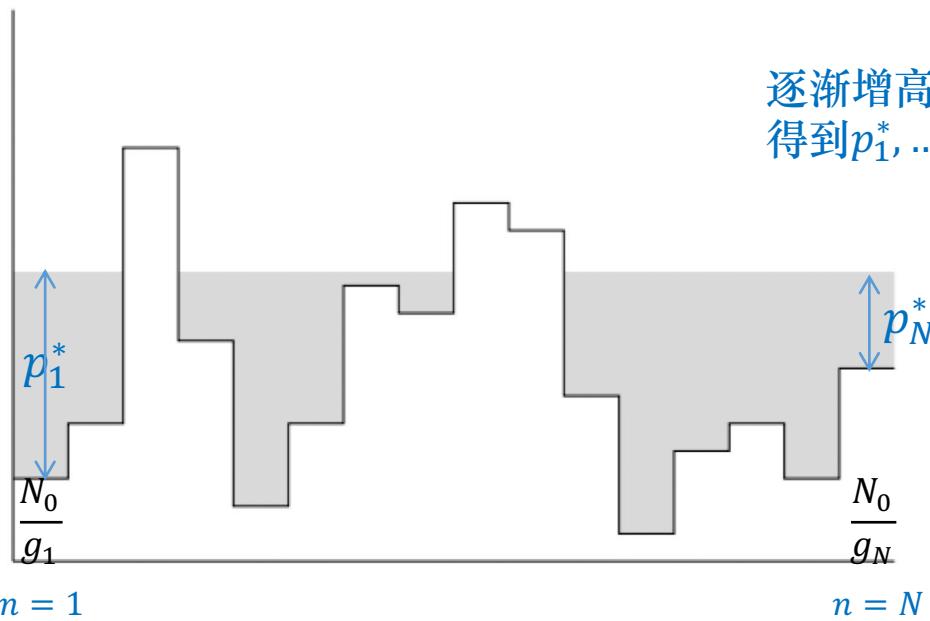
发射功率分配问题



$$\begin{aligned} \max_{p_n} \quad & \sum_{n=1}^N \log_2\left(1 + \frac{g_n p_n}{N_0}\right) \\ \text{s.t.} \quad & \sum_{n=1}^N p_n = P \\ & p_n \geq 0 \end{aligned}$$

注水算法

逐渐增高水面，直到面积等于 P
得到 p_1^*, \dots, p_N^* 等



KKT条件

定理

对前述凸优化问题，若 $f(\mathbf{x})$ 和 $g_i(\mathbf{x})$ 都是连续可微函数且Slater条件成立，那么 \mathbf{x}^* 是全局最优解的充要条件是 \mathbf{x}^* 满足：

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\mathbf{x}^*) = \mathbf{0},$$

$$g_i(\mathbf{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\mathbf{x}^*) = 0, j = 1, \dots, p,$$

$$\lambda_i^* \geq 0, i = 1, \dots, m,$$

$$\lambda_i^* g_i(\mathbf{x}^*) = 0, i = 1, \dots, m.$$

其中， λ_i^*, ν_j^* 是对偶问题的最优解。

对无约束凸优化问题，充要条件可简化为 $\nabla f(\mathbf{x}^*) = \mathbf{0}$

对非凸优化问题，（在一定条件下）KKT条件是全局最优解的必要条件

求解有约束凸优化问题的内点法

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、内点法；

有约束凸优化问题、KKT条件、**内点法**

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

障碍函数

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中， $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数， $h_j(\boldsymbol{x})$ 是仿射函数，Slater条件成立

KKT条件：

$$\begin{aligned} \nabla f(\boldsymbol{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^p \nu_j^* \nabla h_j(\boldsymbol{x}^*) &= \mathbf{0}, \\ g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \quad h_j(\boldsymbol{x}^*) &= 0, j = 1, \dots, p, \\ \lambda_i^* \geq 0, i = 1, \dots, m, \\ \lambda_i^* g_i(\boldsymbol{x}^*) &= 0, i = 1, \dots, m. \end{aligned}$$

互补松弛条件难以处理，需分情况讨论（回顾发射功率分配问题）

障碍函数

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数, $h_j(\boldsymbol{x})$ 是仿射函数, Slater条件成立



$$\begin{aligned} \min \quad & f(\boldsymbol{x}) + \sum_{i=1}^m I_t(g_i(\boldsymbol{x})) \\ \text{s.t.} \quad & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数, $h_j(\boldsymbol{x})$ 是仿射函数, Slater条件成立

$$I_t(u) = -\frac{1}{t} \log(-u)$$

$t > 0$ 控制障碍函数的形状, 当 $t \rightarrow \infty$ 时, 新问题的最优解趋近于原问题最优解

障碍函数

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数, $h_j(\boldsymbol{x})$ 是仿射函数, Slater条件成立



$$\min t f(\boldsymbol{x}) - \sum_{i=1}^m \log(-g_i(\boldsymbol{x}))$$

$$\text{s.t. } \mathbf{A}\boldsymbol{x} = \mathbf{b},$$

$$\text{var. } \boldsymbol{x} \in \mathcal{R}^n.$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数

该问题是不是凸优化问题?

障碍函数

分析新目标函数是否是凸函数: $tf(x) - \sum_{i=1}^m \log(-g_i(x))$

分析 $-\log(-g_i(x))$ 是否是凸函数:

梯度: $-\frac{\nabla g_i(x)}{g_i(x)}$ 海森矩阵: $\boxed{-\frac{1}{g_i(x)} \nabla^2 g_i(x) + \frac{1}{(g_i(x))^2} \nabla g_i(x)(\nabla g_i(x))^T}$

在满足 $g_i(x) \leq 0$ 的区域, 海森矩阵为半正定矩阵

对 $n \times n$ 实对称矩阵 A , 若对任意向量 $u \in \mathbb{R}^n$ 有 $u^T A u \geq 0$, A 为半正定矩阵。

半正定矩阵之和依然是半正定矩阵。

障碍函数

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数, $h_j(\boldsymbol{x})$ 是仿射函数, Slater 条件成立



$$\begin{aligned} \min \quad & tf(\boldsymbol{x}) - \sum_{i=1}^m \log(-g_i(\boldsymbol{x})) \\ \text{s.t.} \quad & \mathbf{A}\boldsymbol{x} = \mathbf{b}, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数

该问题是凸优化问题

如何求解?

障碍函数

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p, \\ \text{var.} \quad & \boldsymbol{x} \in \mathcal{R}^n. \end{aligned}$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数, $h_j(\boldsymbol{x})$ 是仿射函数, Slater条件成立



新问题的KKT条件

$$\min t f(\boldsymbol{x}) - \sum_{i=1}^m \log(-g_i(\boldsymbol{x}))$$

$$\text{s.t. } \mathbf{A}\boldsymbol{x} = \mathbf{b},$$

$$\text{var. } \boldsymbol{x} \in \mathcal{R}^n.$$

其中, $f(\boldsymbol{x}), g_i(\boldsymbol{x})$ 是连续可微凸函数

$$t \nabla f(\boldsymbol{x}^*) - \sum_{i=1}^m \frac{\nabla g_i(\boldsymbol{x}^*)}{g_i(\boldsymbol{x}^*)} + \sum_{j=1}^p v_j^* \mathbf{a}_j = \mathbf{0},$$
$$g_i(\boldsymbol{x}^*) \leq 0, i = 1, \dots, m, \mathbf{A}\boldsymbol{x}^* = \mathbf{b}.$$

\mathbf{a}_j 是 \mathbf{A} 的第 j 行

可以通过消元将原问题化为无约束问题再用牛顿法等, 或直接对KKT条件的方程组进行求解

内点法

内点法

- 0 确定一个初始可行解 \tilde{x} ， 初始参数值 $t > 0$
- 1 由 \tilde{x} 出发，在当前 t 取值下，计算 $x^*(t)$

$$\begin{aligned} \min \quad & tf(x) - \sum_{i=1}^m \log(-g_i(x)) \\ \text{s.t.} \quad & Ax = b, \\ \text{var.} \quad & x \in \mathcal{R}^n. \end{aligned}$$

- 2 将 $x^*(t)$ 赋值给 \tilde{x}
- 3 如果循环结束条件（如 t 大于一定阈值）满足，则结束；
否则，增加 t 取值（如令 $t \leftarrow \mu t$ ），返回1

最优化问题的类别

Part1 如何从复杂度的角度衡量算法好坏？

算法复杂度、P问题与NP问题与NP难问题

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、内点法；

有约束凸优化问题、KKT条件、内点法

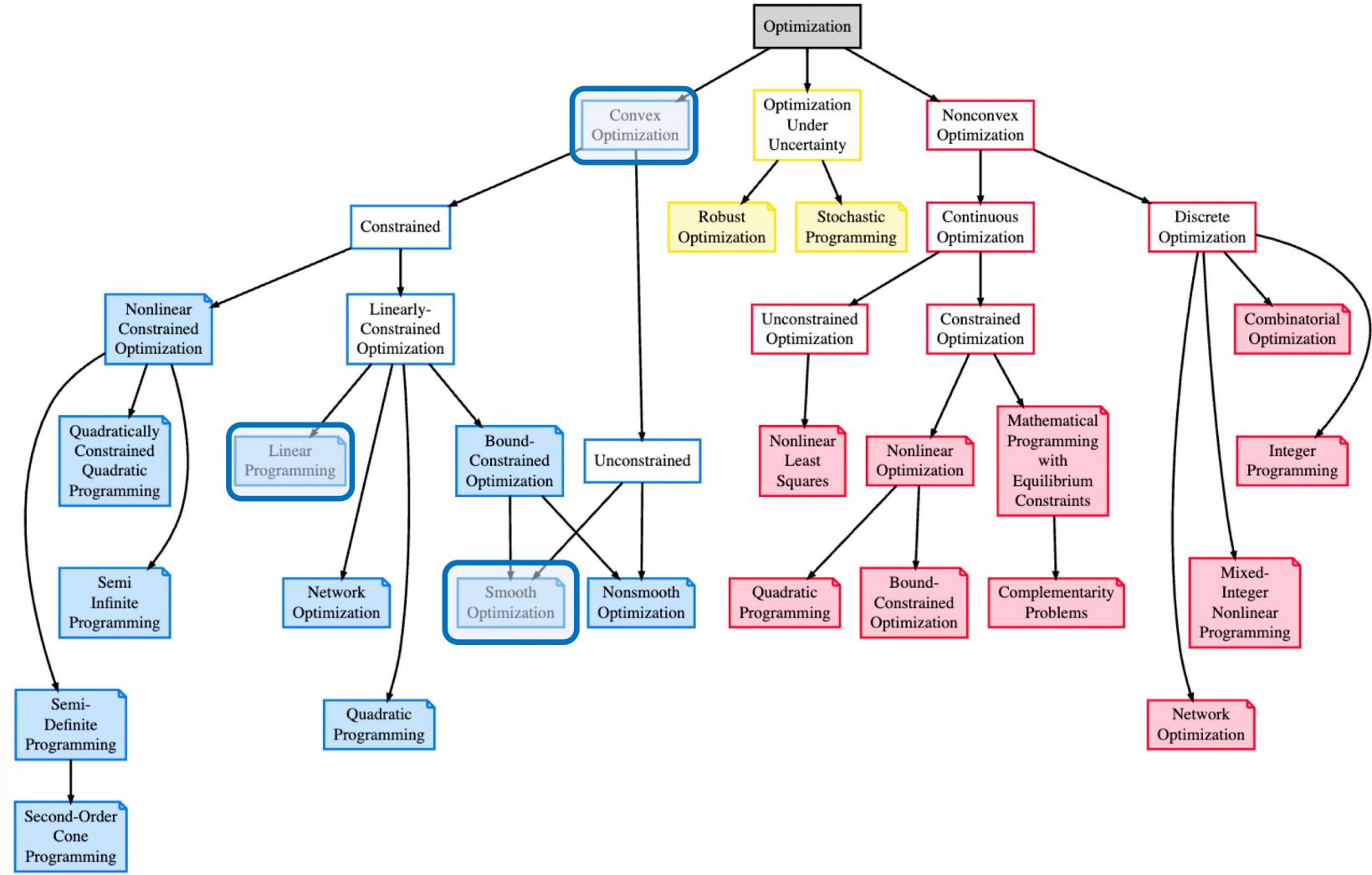
Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

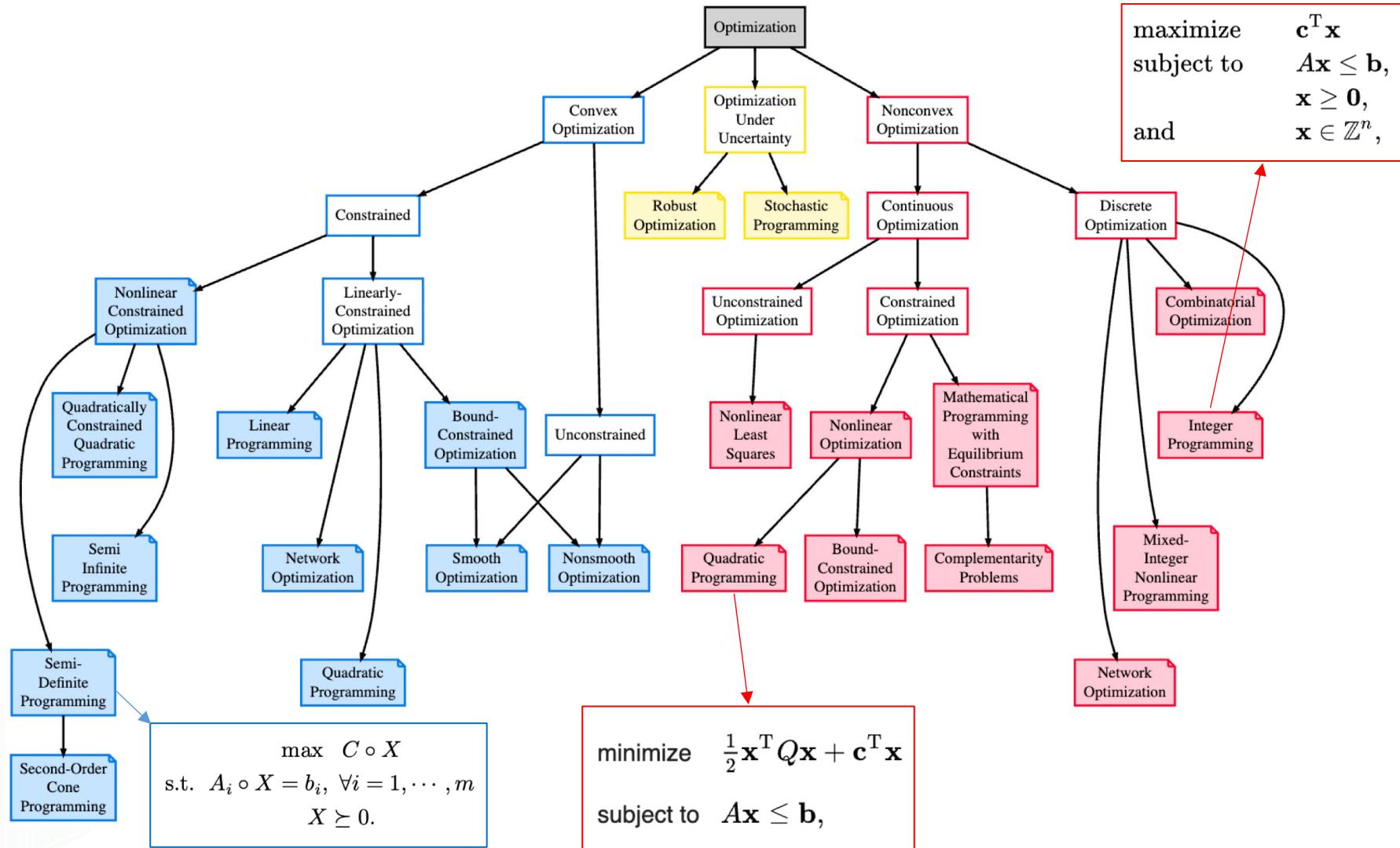
Part4 针对较复杂的多目标最优化问题，如何找到较好解？

多目标优化问题、NSGA-II算法

最优化问题的类别



最优化问题的类别



凸优化

Convex Optimization

Stephen Boyd

Department of Electrical Engineering
Stanford University

Lieven Vandenberghe

Electrical Engineering Department
University of California, Los Angeles



CAMBRIDGE
UNIVERSITY PRESS

介绍凸优化理论的经典书籍

完整电子版 及 Stanford课程幻灯（免费）

web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

web.stanford.edu/~boyd/cvxbook/bv_cvxslides.pdf

[book] Convex optimization

[S Boyd, SP Boyd, L Vandenberghe - 2004 - books.google.com](#)

... general **convex optimization** ... **convex** analysis, or the mathematics of **convex optimization**; several existing texts cover these topics well. Nor is the book a survey of algorithms for **convex** ...

☆ Save ⚡ Cite Cited by 67701 Related articles All 28 versions ☰

智能优化算法

Part1 如何从复杂度的角度衡量算法好坏？

算法复杂度、P问题与NP问题与NP难问题

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

无约束凸优化问题、梯度下降法、牛顿法；

线性规划问题、单纯形法、内点法；

有约束凸优化问题、KKT条件、内点法

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

多目标优化问题、NSGA-II算法

对于很难的问题（如NP难问题、目标函数不可微的问题），不存在已知的、理论证明可找到全局最优解的算法。此时，如何通过经验（如生物演化生存的经验）、不依赖梯度信息找到较优解？

智能优化算法

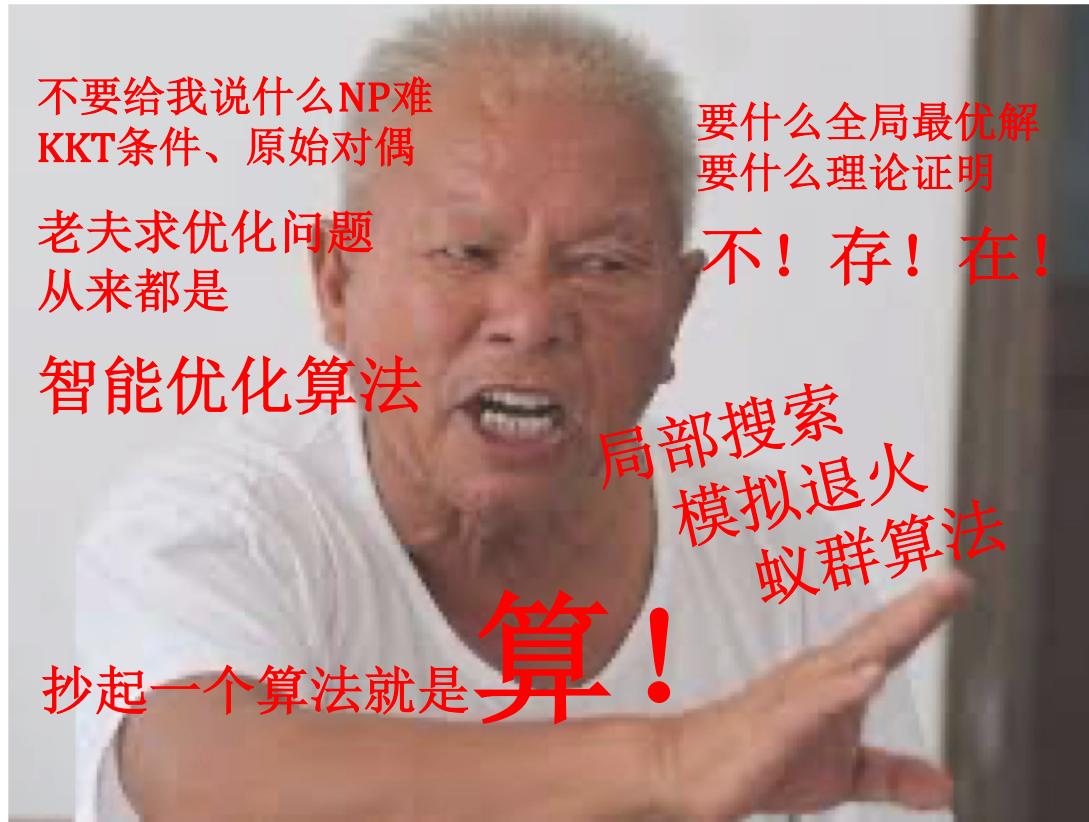
智能（intelligent）优化算法属于启发式（heuristic）算法

- “智能”、“启发”：无法从理论上严格证明性能

智能优化算法

智能（intelligent）优化算法属于启发式（heuristic）算法

- “智能”、“启发”：无法从理论上严格证明性能

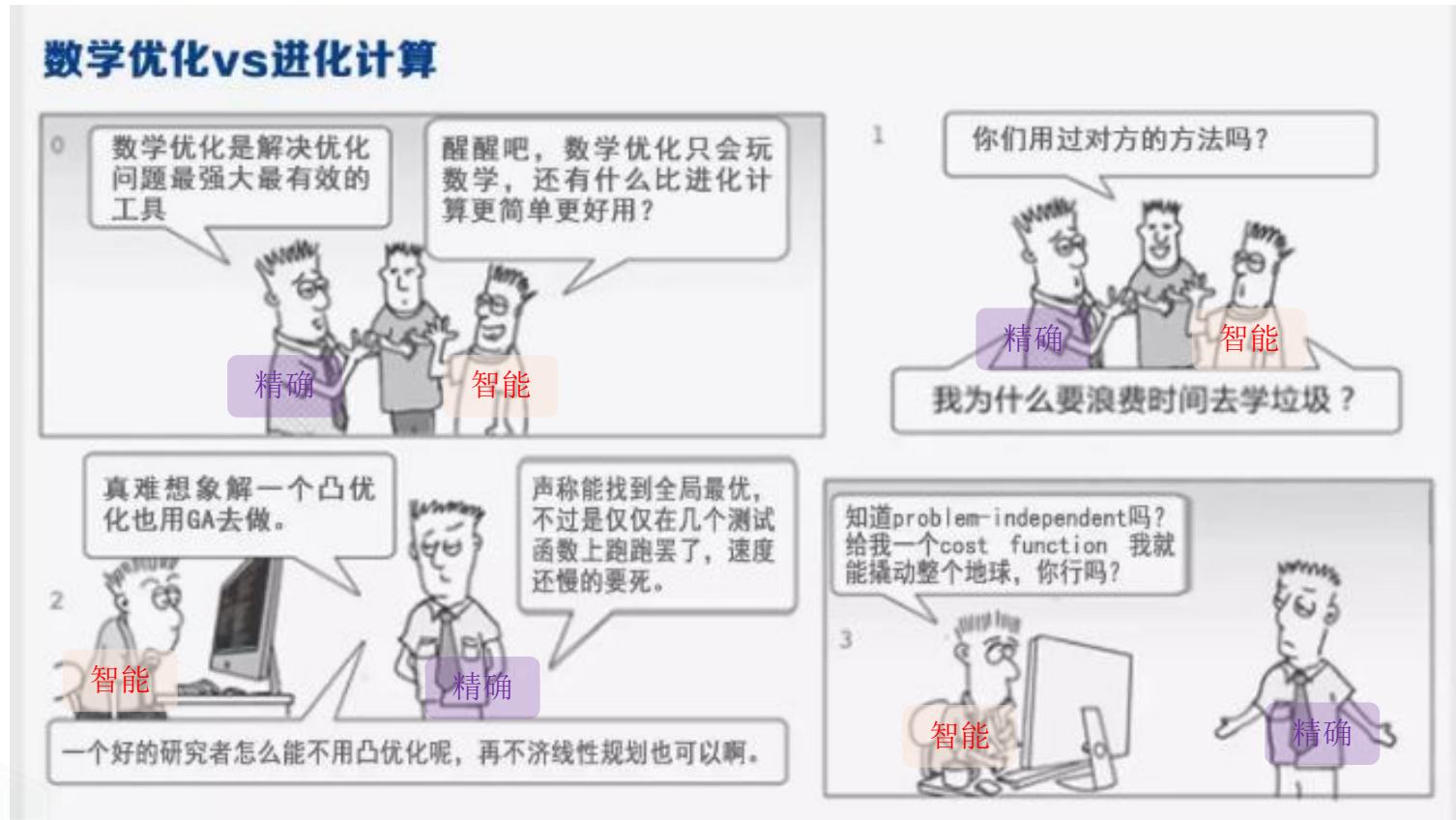


精确优化算法 vs 智能优化算法

精确优化算法（如单纯形法、内点法） VS 智能优化算法（如模拟退火、遗传算法）

精确优化算法 vs 智能优化算法

精确优化算法（如单纯形法、内点法） VS 智能优化算法（如模拟退火、遗传算法）



图片取自杨翠娥、王源 <进化计算PK数学优化>

精确优化算法 vs 智能优化算法

精确优化算法（如单纯形法、内点法） VS 智能优化算法（如模拟退火、遗传算法）

- 求解最优化问题的两种模式



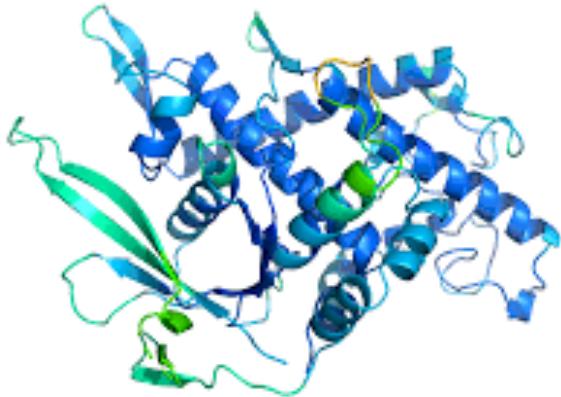
玄门正宗
利于长远、修习周期长



速成武功
见效快、缺少根基

精确优化算法 vs 智能优化算法

智能优化算法的思想有许多应用



Improved protein structure prediction using potentials from deep learning

[AW Senior, R Evans, J Jumper, J Kirkpatrick, L Sifre... - Nature, 2020 - nature.com](#)

Protein structure prediction can be used to determine the three-dimensional shape of a protein from its amino acid sequence 1. This problem is of fundamental importance as the structure of a protein largely determines its function 2; however, protein structures can be difficult to determine experimentally. Considerable progress has recently been made by leveraging genetic information. It is possible to infer which amino acid residues are in contact by analysing covariation in homologous sequences, which aids in the prediction of ...

[☆ Save](#) [⤒ Cite](#) [Cited by 1447](#) [Related articles](#) [All 11 versions](#)

AlphaFold: 针对输入的蛋白质氨基酸序列，利用深度学习，预测蛋白质的三维折叠结构

在最后一个步骤中，运用到了进化算法的思想

局部搜索

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

局部搜索

通过不断对现有解作局部改进而得到较优解

- 简单：算法复杂度低
- 有效：可应用于生产计划、生产调度、路径规划、资源配置等。许多（甚至大多数）运筹问题都可以用局部搜索方法求解（局部最优解）

局部搜索



基本步骤

第1步：选择初始解

第2步：对解做出局部调整取得改进

第3步：重复第2步直到找到目标状态（或者运行超时）

局部搜索



基本步骤

第1步：选择初始解 一般随机产生（有改进空间）

第2步：对解做出局部调整取得改进

如何评价当前解的质量?
如何定义局部？

第3步：重复第2步直到找到目标状态（或者运行超时）

设计具体操作时保证计算速度要快、一般不保存历史过程

登山搜索

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

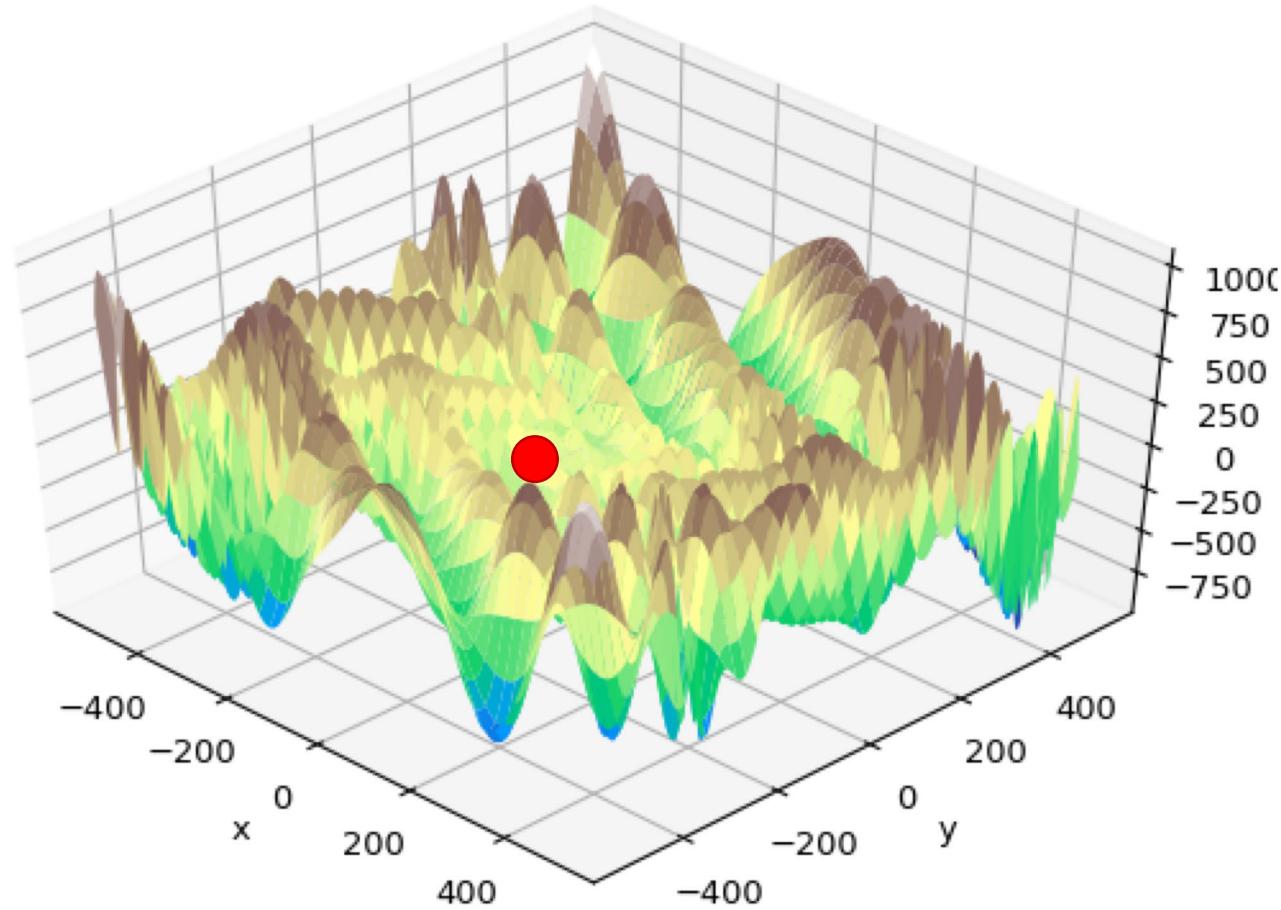
登山搜索

在陌生的山区，大雾弥漫，不能远视，如何才能到达山区中最高/较高的山峰？



登山搜索

(x,y)



登山搜索

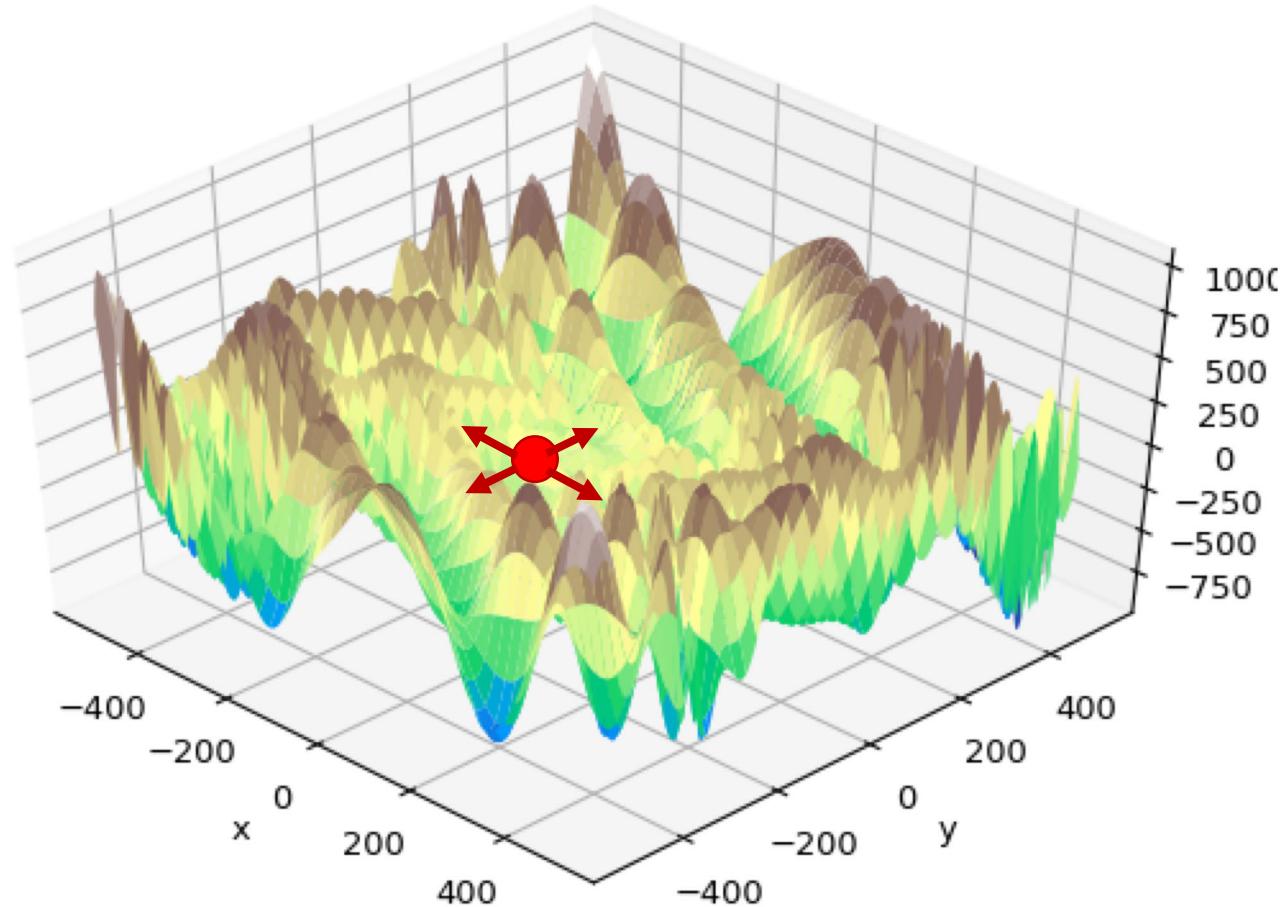
$h(x,y)$

$h(x+1,y)$

$h(x-1,y)$

$h(x,y+1)$

$h(x,y-1)$

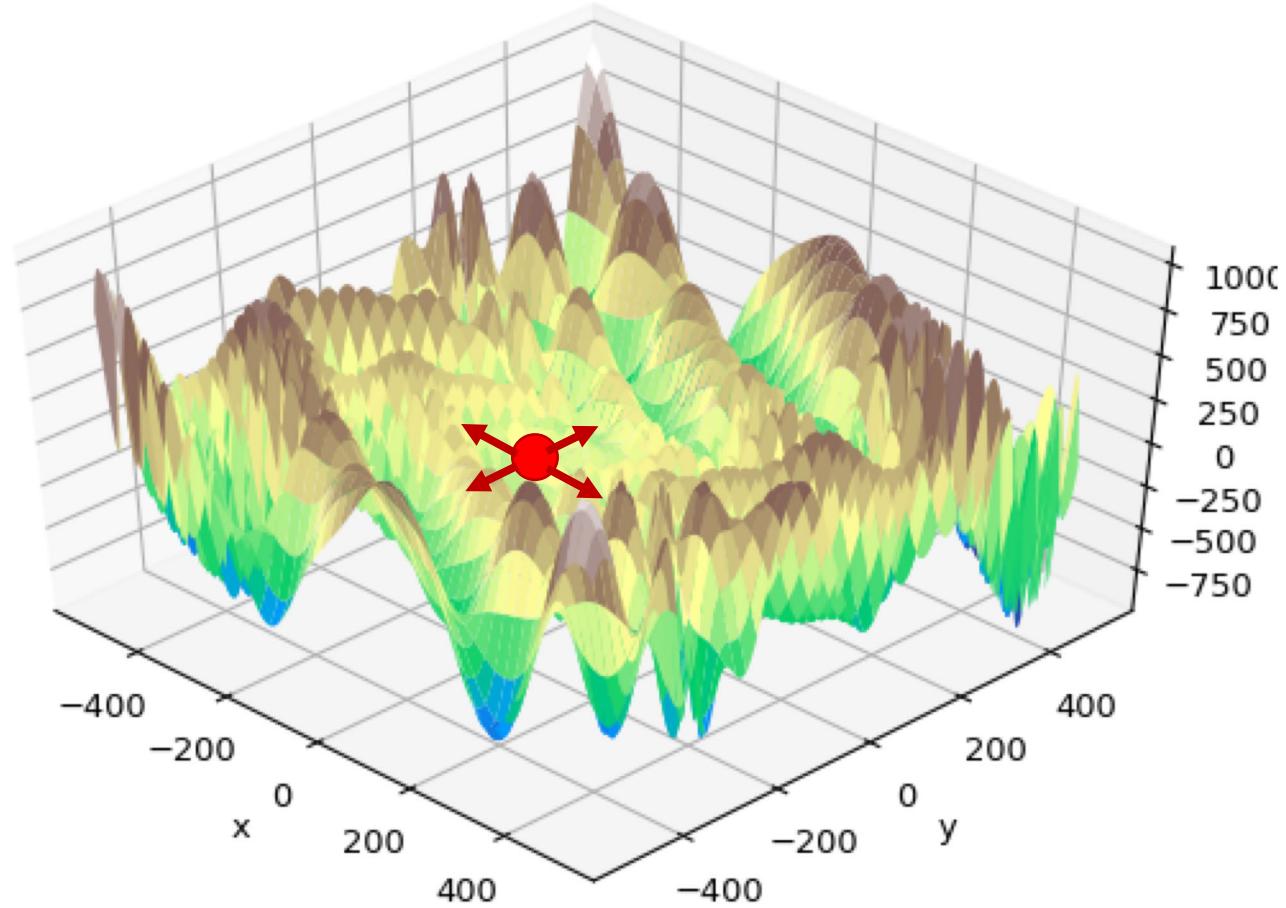


登山搜索

"Like climbing Everest in thick fog with amnesia"

浓雾——只在邻域搜索下一步

健忘症——不记录历史路径



登山搜索不能保证得到全局最优解

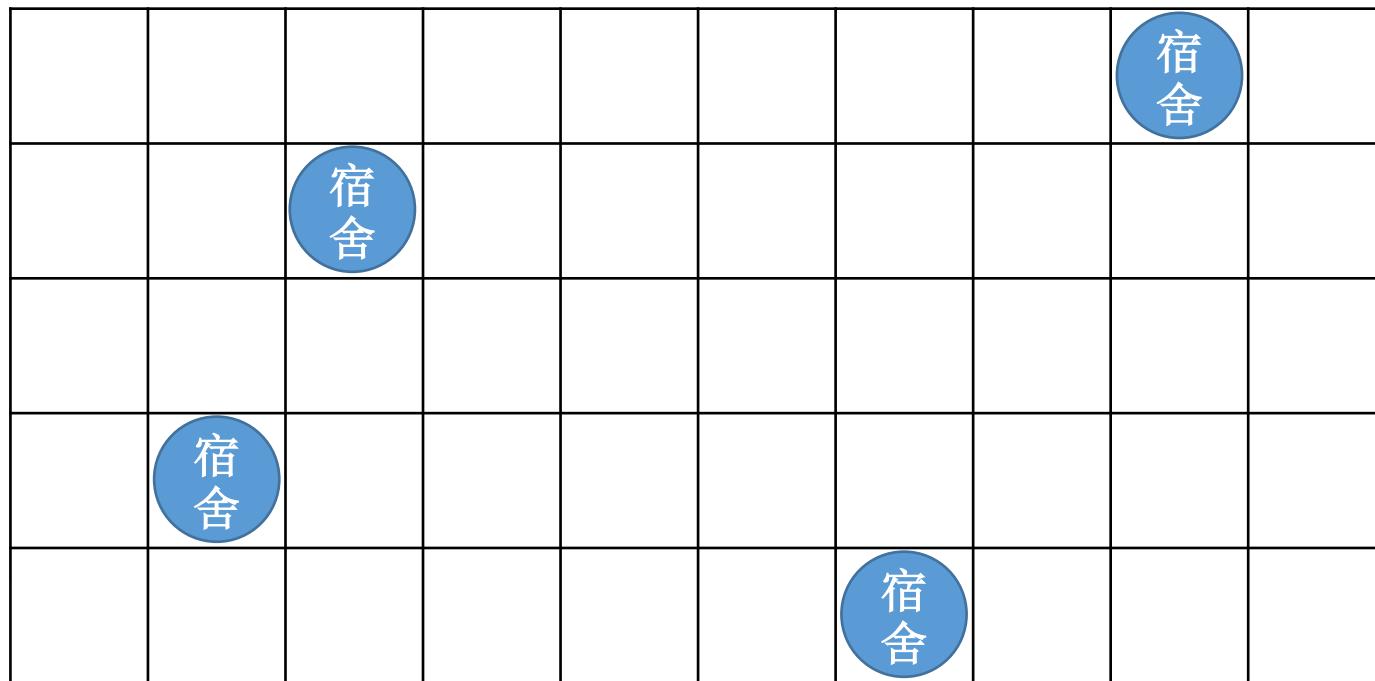
登山搜索

登山搜索（返回最大化问题的局部最优解）

- 0 初始化当前解 x_{current}
- 1 寻找当前解的所有邻域解 自行定义邻域
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 x_{neighbor}
- 3 若 $f(x_{\text{current}}) \geq f(x_{\text{neighbor}})$, 结束搜索; 否则, $x_{\text{current}} \leftarrow x_{\text{neighbor}}$, 返回1

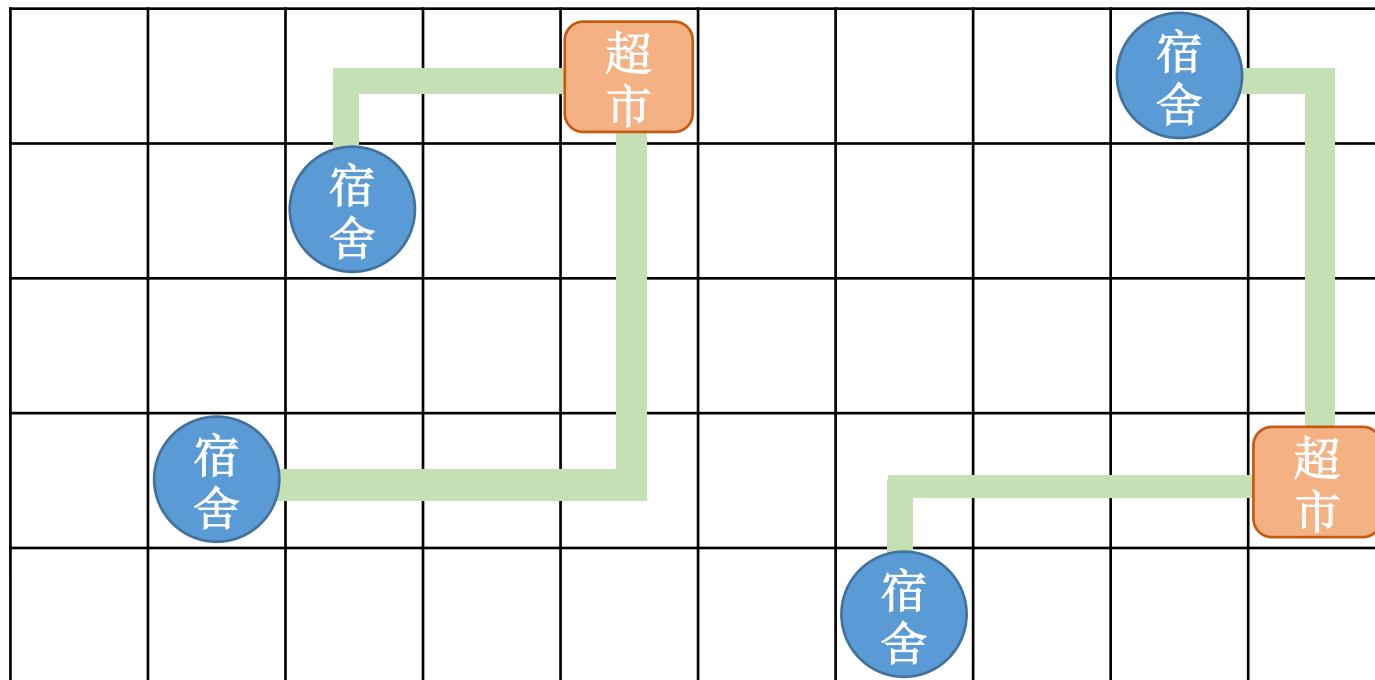
例子回顾

校区要新建两个超市，问如何选址可以最小化每个宿舍到离其最近超市的距离之和？



例子回顾

校区要新建两个超市，问如何选址可以最小化每个宿舍到离其最近超市的距离之和？



以上选址方案对应的目标函数值为17

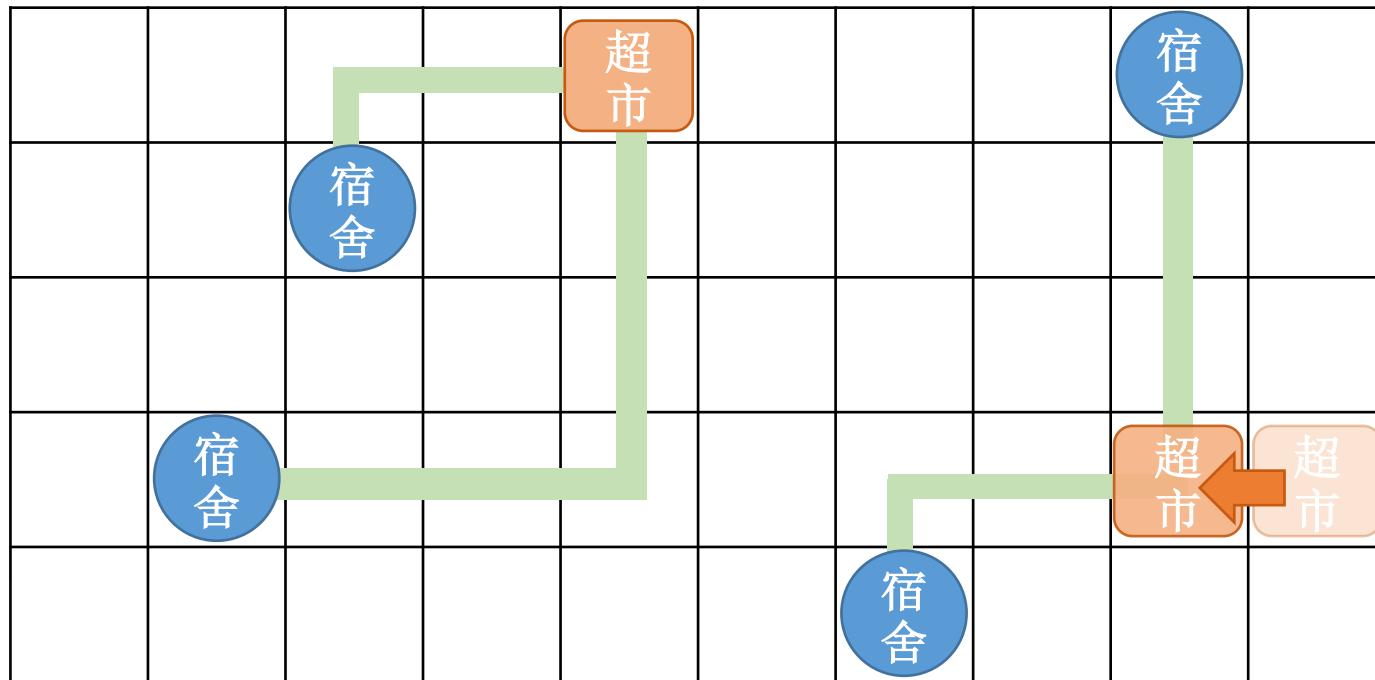
例子回顾

				超市	超市	超市			宿舍	
			宿舍		超市					
										超市
		宿舍						超市	超市	
							宿舍			超市

以上选址方案对应的目标函数值为17

例子回顾

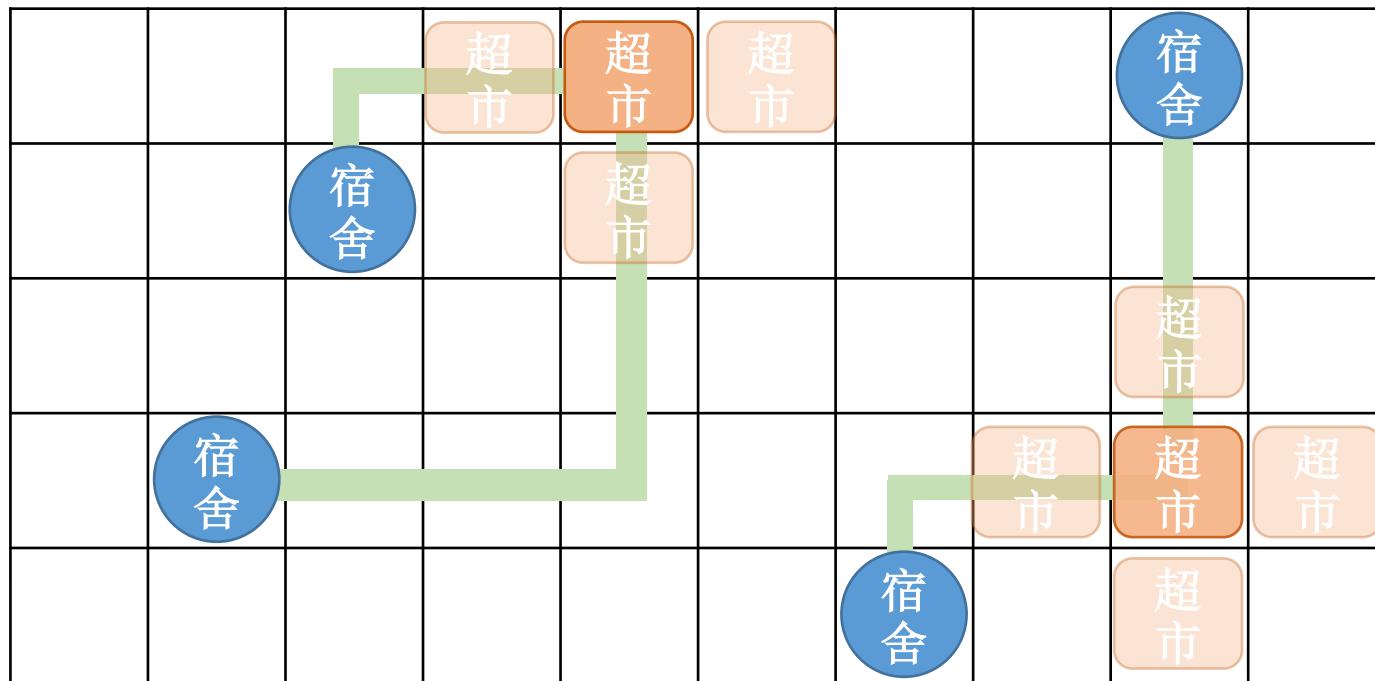
从6种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从17降低为15

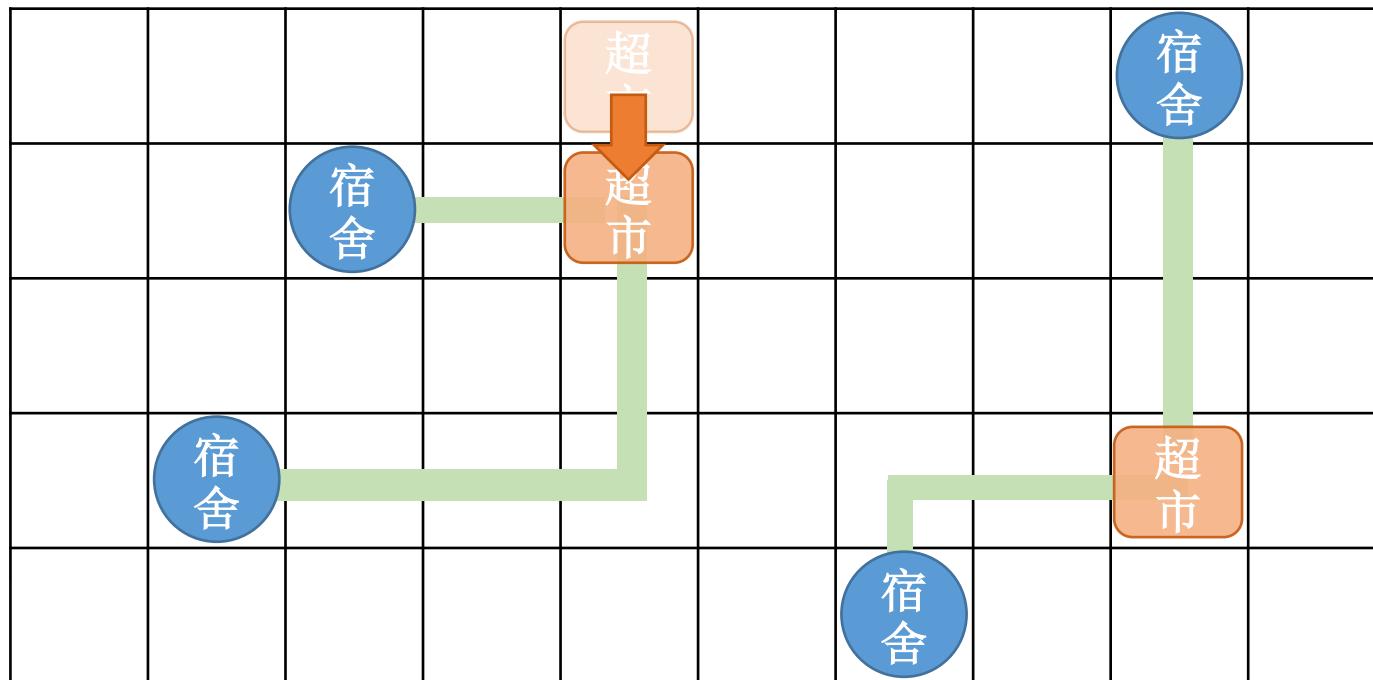
例子回顾

针对新方案又有7种更改方案
(仍仅考虑将一个超市移动一格的更改方案)



例子回顾

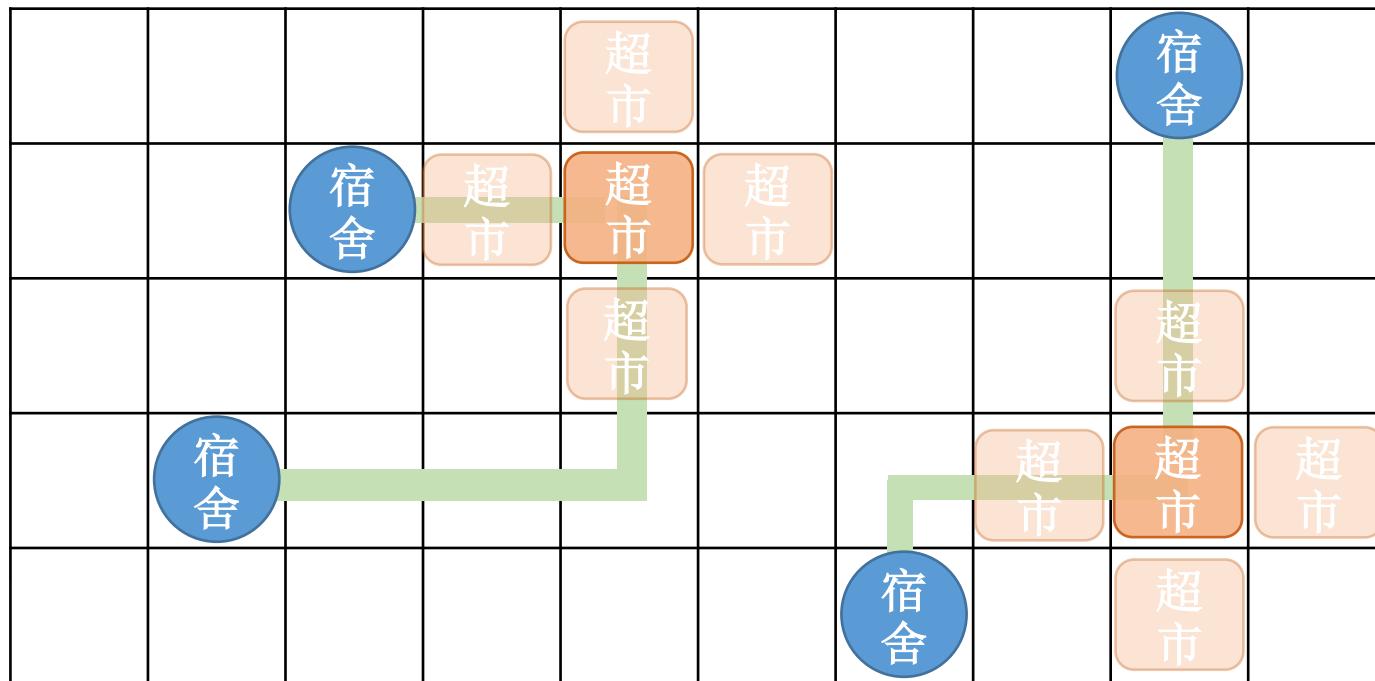
从7种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从15降低为13

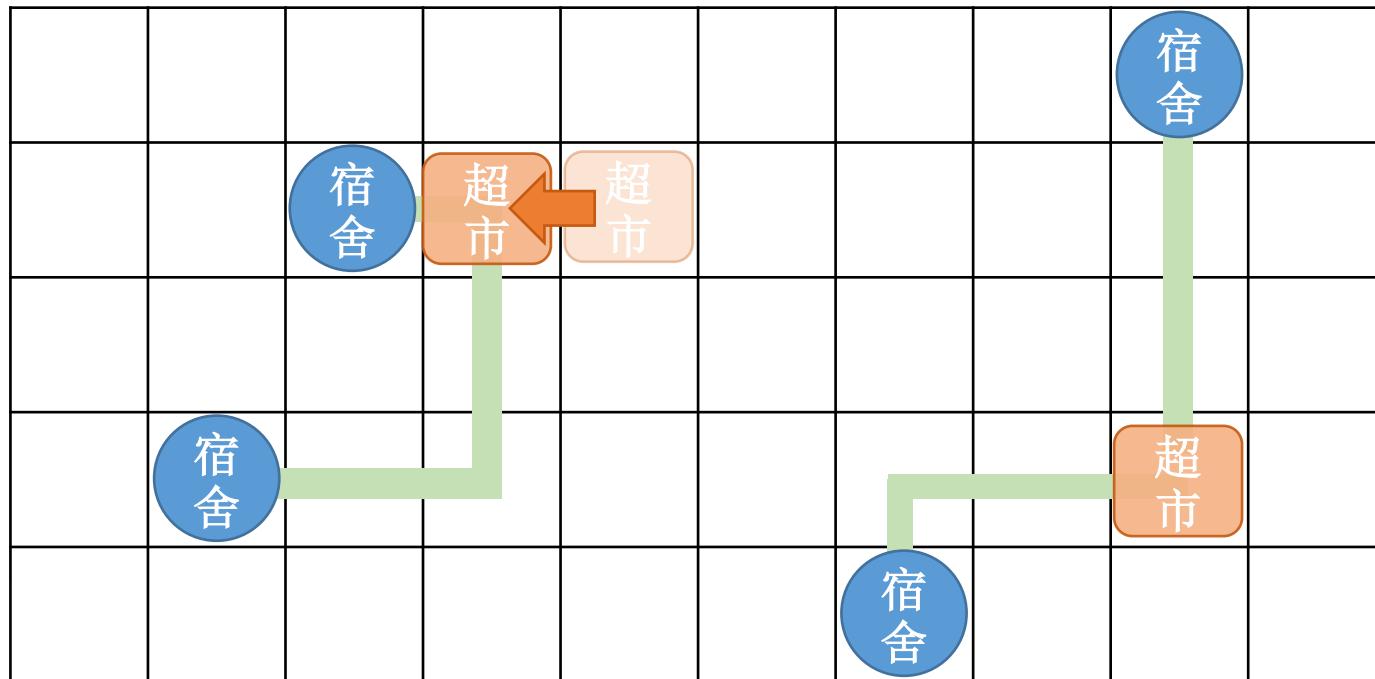
例子回顾

针对新方案又有8种更改方案
(仍仅考虑将一个超市移动一格的更改方案)



例子回顾

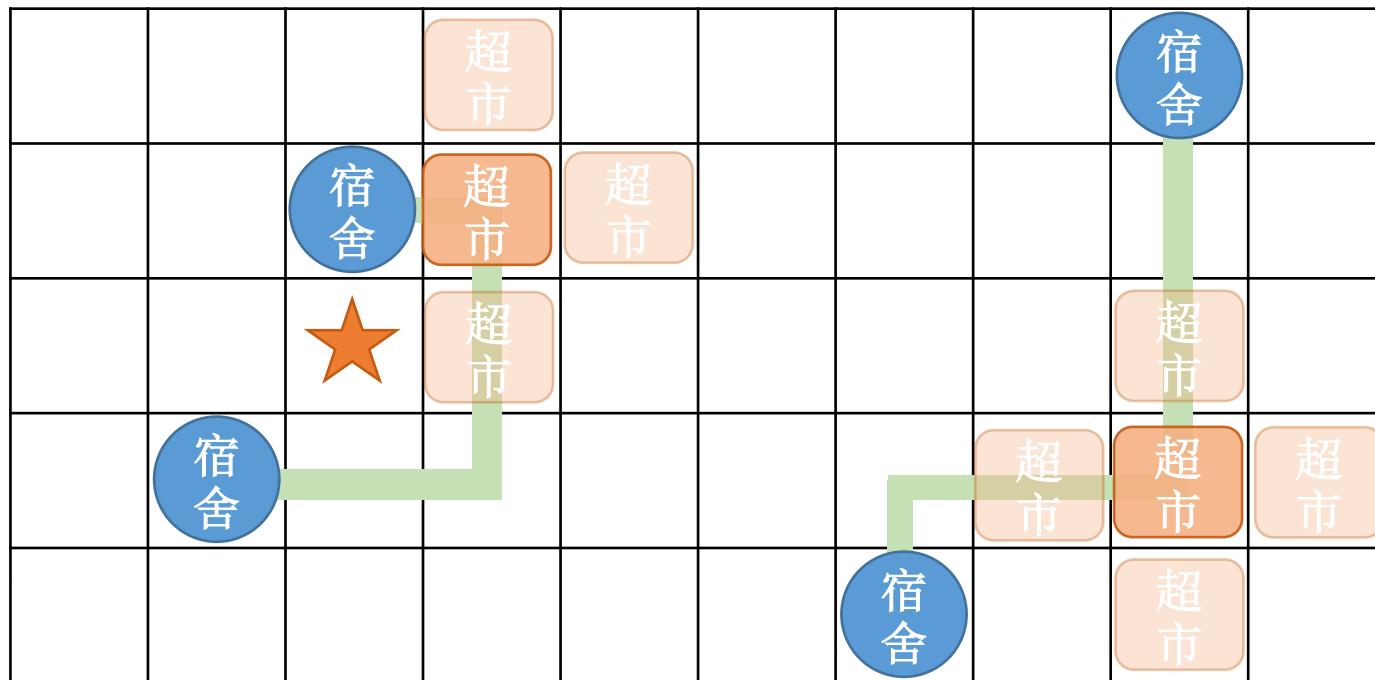
从8种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从13降低为11

例子回顾

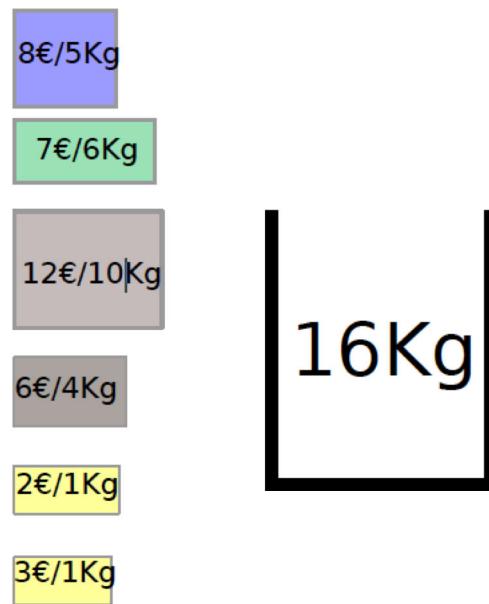
针对新方案有7种更改方案，但都不能进一步降低目标函数



该算法停止搜索，输出结果，对应目标函数值为11

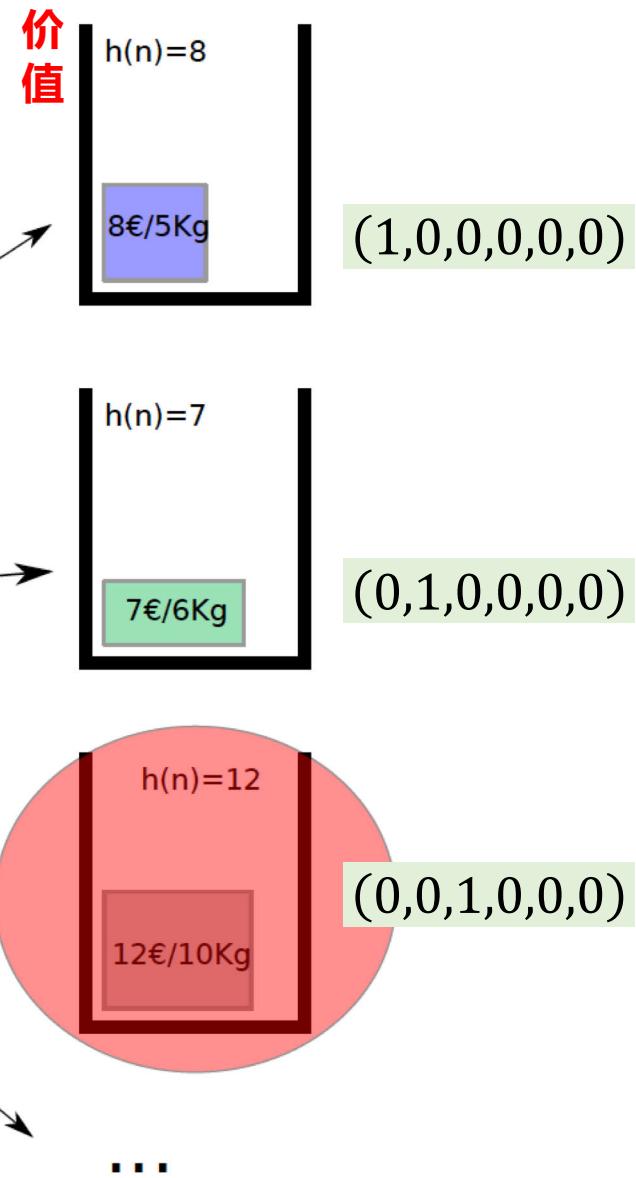
0-1背包问题（NP难）

给定一组物品，已知每个物品的重量和价值，问给定一个背包，找到方案使得装进背包的物品的重量不超过W、且价值最大



0-1背包问题（NP难）

物品列表：
8€/5Kg
7€/6Kg
12€/10Kg
6€/4Kg
2€/1Kg
3€/1Kg

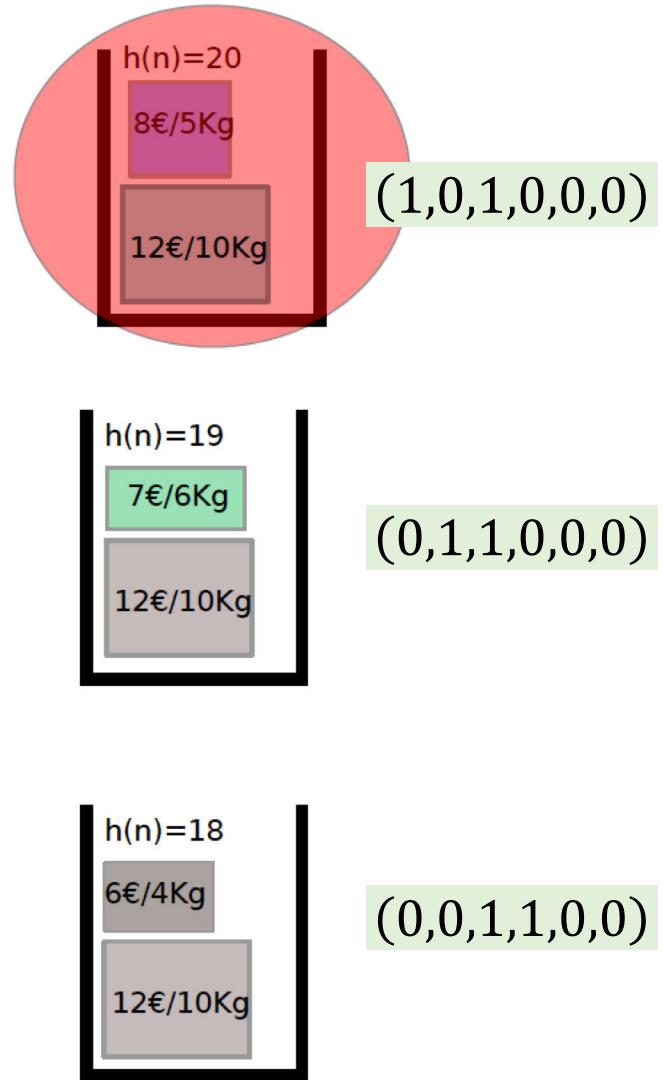
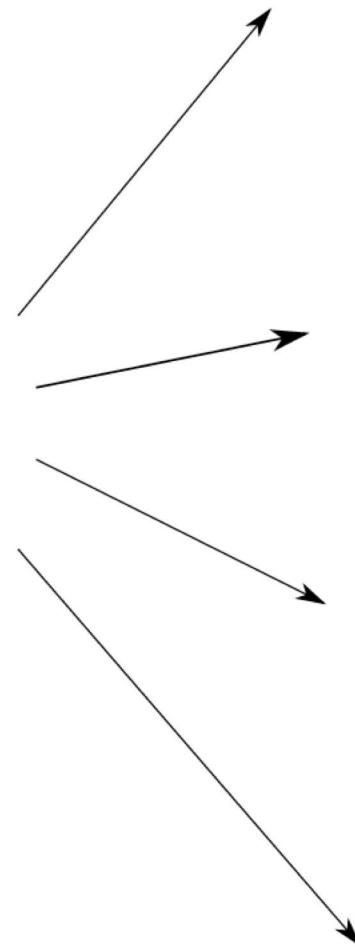
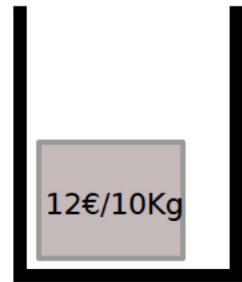


邻域自行定义

0-1背包问题（NP难）

8€/5Kg
7€/6Kg
~~12€/10Kg~~
6€/4Kg
2€/1Kg
3€/1Kg

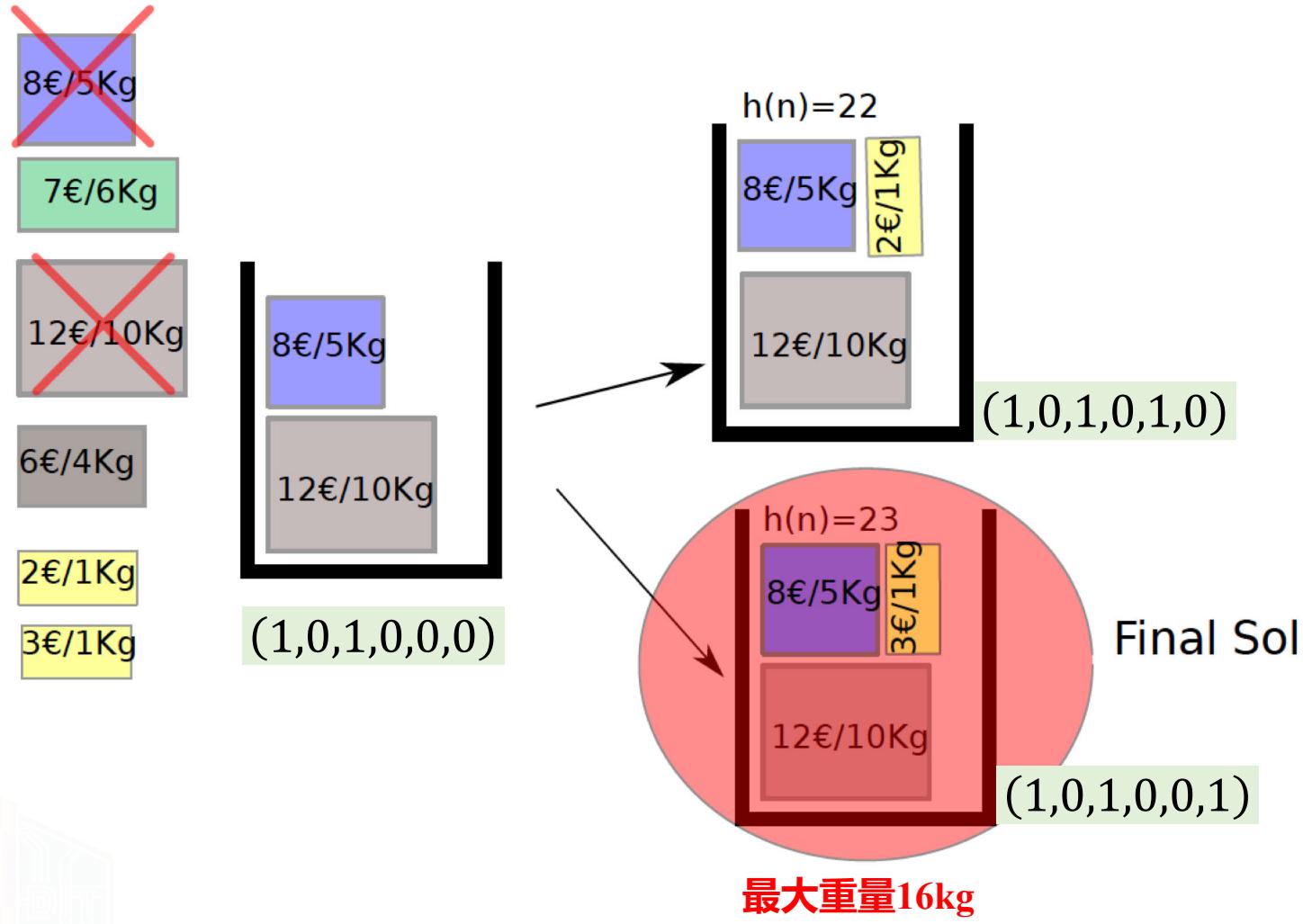
(0,0,1,0,0,0)



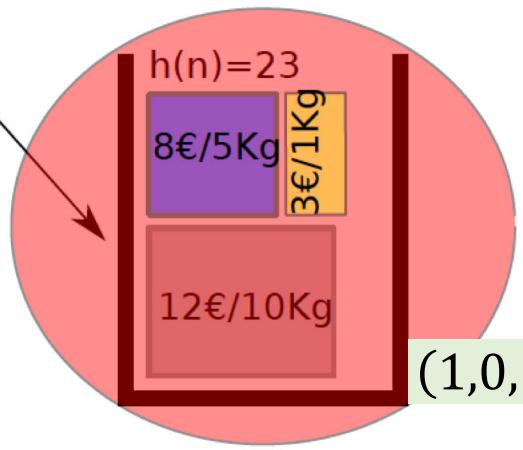
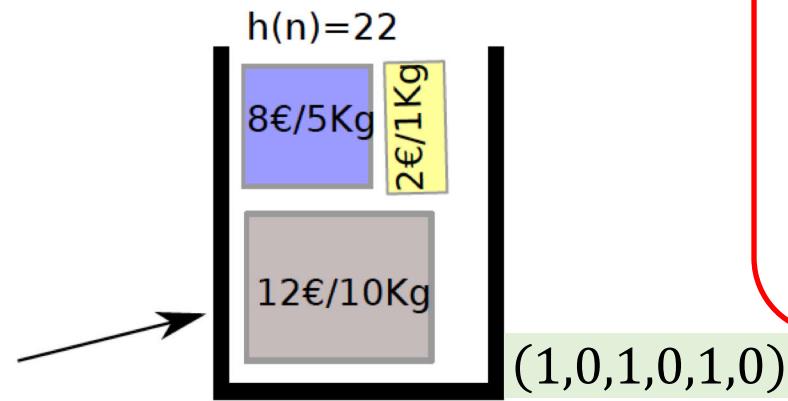
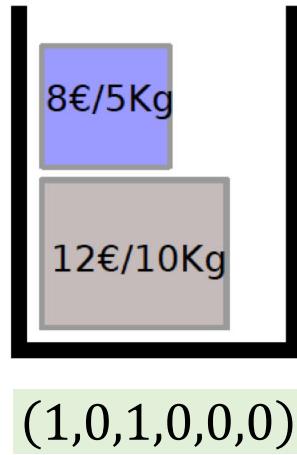
邻域自行定义

...

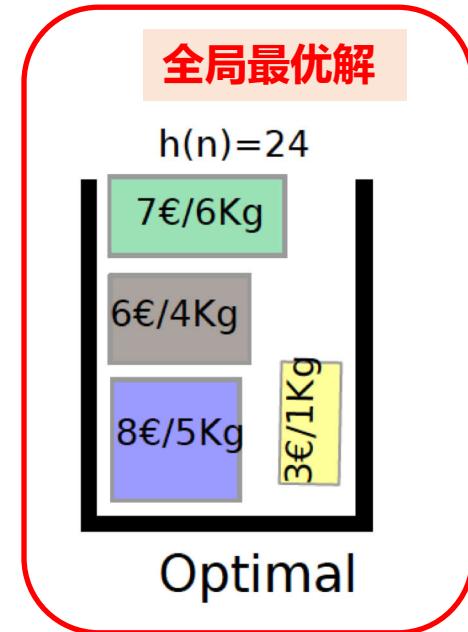
0-1背包问题（NP难）



0-1背包问题（NP难）



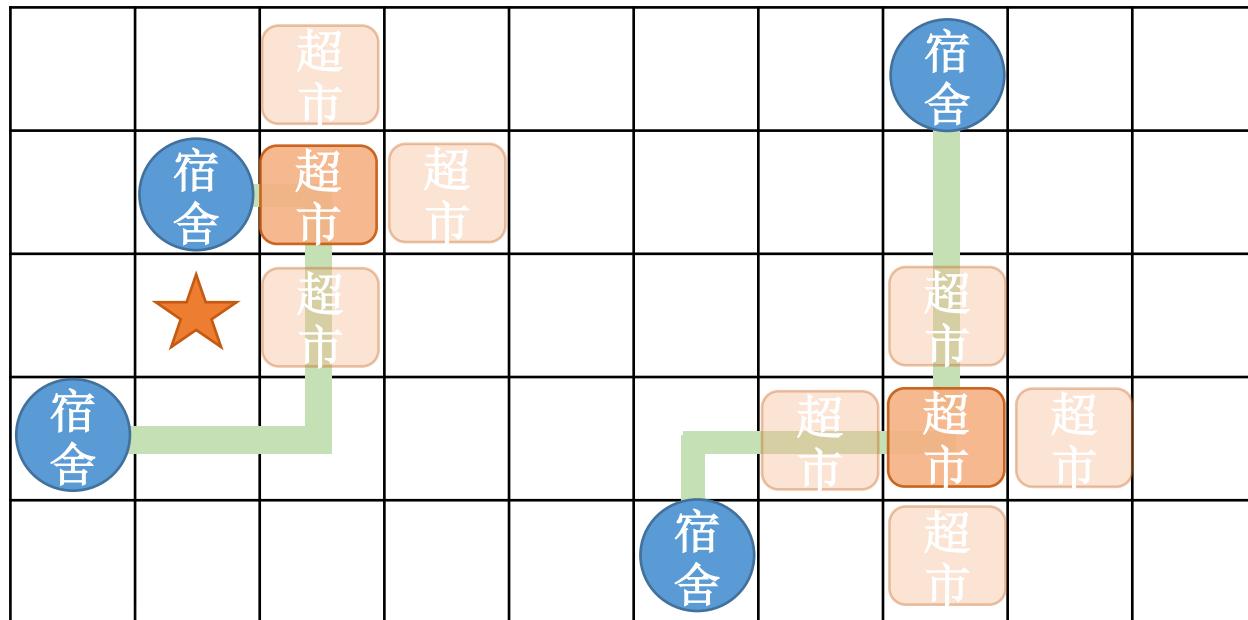
最大重量16kg



全局最优解

0-1背包问题（NP难）

超市选址、背包问题都有明确的待最大化的目标函数



8€/5Kg

7€/6Kg

12€/10Kg

6€/4Kg

2€/1Kg

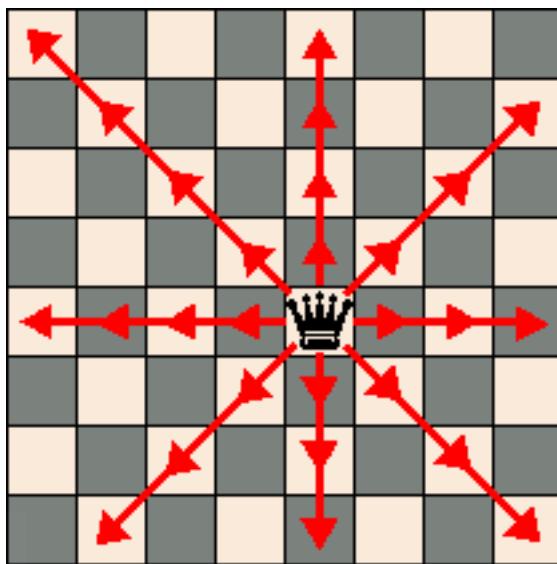
3€/1Kg

16Kg
Sol Inicial

约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

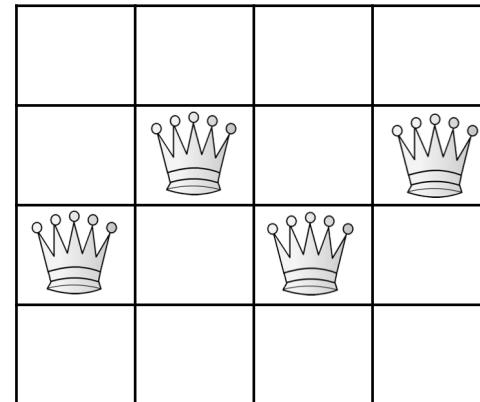
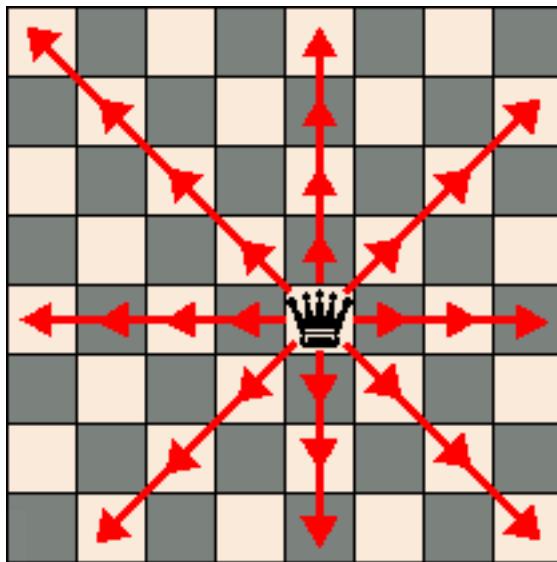
皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



比如以上摆放位置是当前解
如何用登山搜索找到满足要求的解？

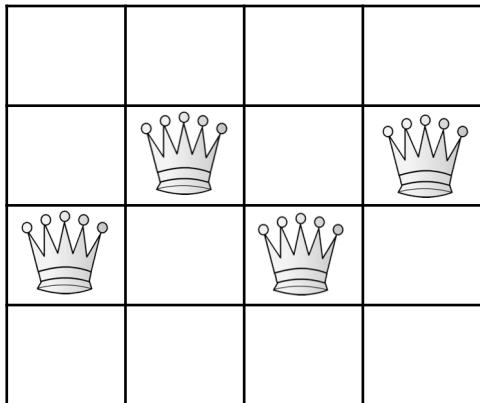


- 1 没有目标函数衡量不同解的质量？
- 2 如何定义邻域解？

约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



1 没有目标函数衡量不同解的质量？

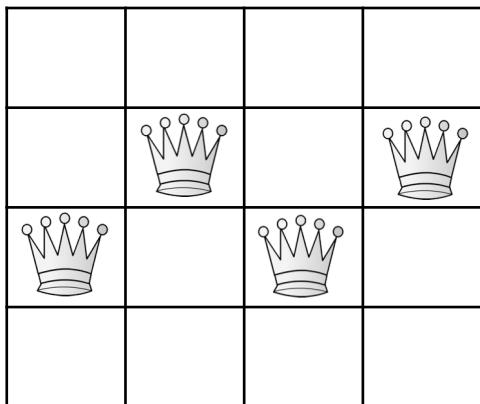
自行定义为当前冲突对的总数
如左图的目标函数值为5

2 如何定义邻域解？

约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



1 没有目标函数衡量不同解的质量？

自行定义为当前冲突对的总数
如左图的目标函数值为5

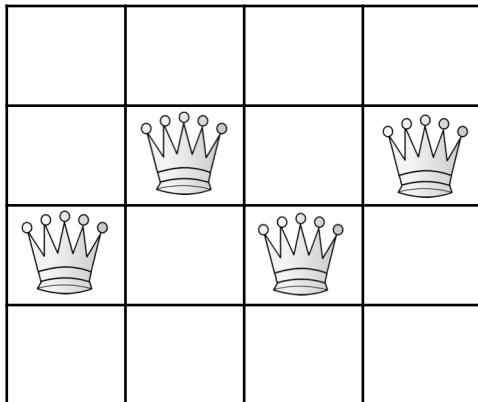
2 如何定义邻域解？

- 已知每一列最后都应该有一个皇后，所以限制每个皇后只能在当前列上下移动

约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



1 没有目标函数衡量不同解的质量？

自行定义为当前冲突对的总数
如左图的目标函数值为5

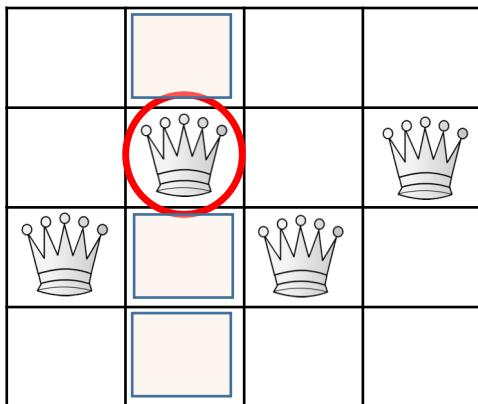
2 如何定义邻域解？

- 已知每一列最后都应该有一个皇后，所以限制每个皇后只能在当前列上下移动
- 一种有效的邻域定义方式：根据当前冲突最多皇后的可能位置的集合确定邻域

约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击



1 没有目标函数衡量不同解的质量？

自行定义为当前冲突对的总数
如左图的目标函数值为5

2 如何定义邻域解？

- 已知每一列最后都应该有一个皇后，所以限制每个皇后只能在当前列上下移动
- 一种有效的邻域定义方式：根据当前冲突最多皇后的可能位置的集合确定邻域

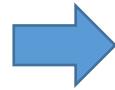
约束满足问题

还可以用于求解约束满足问题（Constrained Satisfaction Problem）

皇后问题（Queens）：在 $N \times N$ 的棋盘中放置 N 个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击

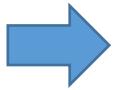
			
			

冲突对： 5



冲突对： 2



冲突对： 0

约束满足问题

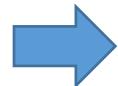
最小冲突启发式（Min-Conflicts Heuristic）算法

生成初始状态（随机）

重复以下过程

- 选取一个冲突变量
- 重新为该变量赋值，使得冲突次数最小
- 如果新状态没有冲突产生，那么返回该赋值





冲突对： 5

冲突对： 2

冲突对： 0

约束满足问题

最小冲突启发式（Min-Conflicts Heuristic）算法

生成初始状态（随机）

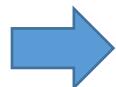
重复以下过程

- 选取一个冲突变量
- 重新为该变量赋值，使得冲突次数最小
- 如果新状态没有冲突产生，那么返回该赋值

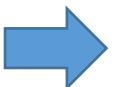
允许侧向移动（sideways moves）

只要还有冲突没有消除，
即便 $f(x_{\text{current}}) = f(x_{\text{neighbor}})$ ，
仍要继续执行算法

冲突对： 5



冲突对： 2



冲突对： 0

约束满足问题

最小冲突启发式（Min-Conflicts Heuristic）算法

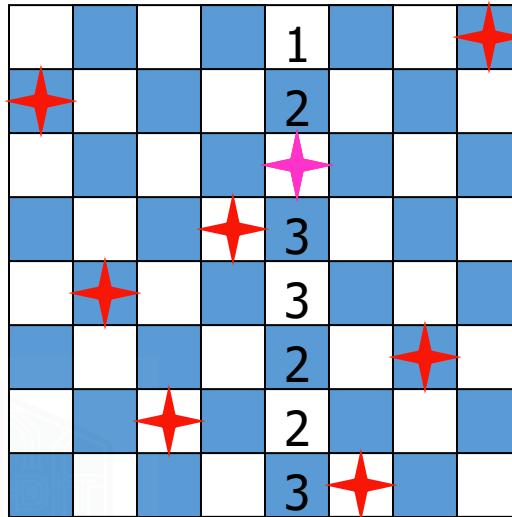
生成初始状态（随机）

重复以下过程

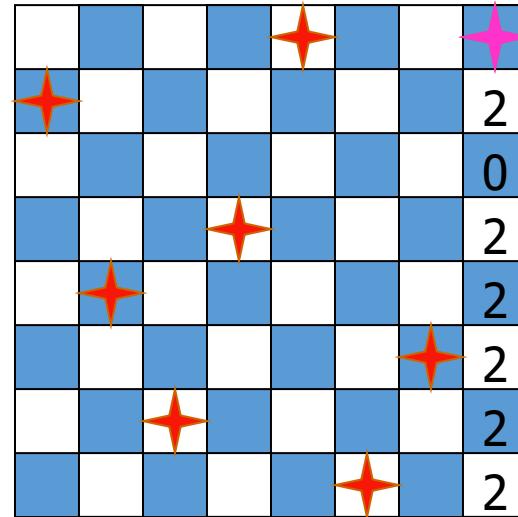
- 选取一个冲突变量
- 重新为该变量赋值，使得冲突次数最小
- 如果新状态没有冲突产生，那么返回该赋值

允许侧向移动（sideways moves）

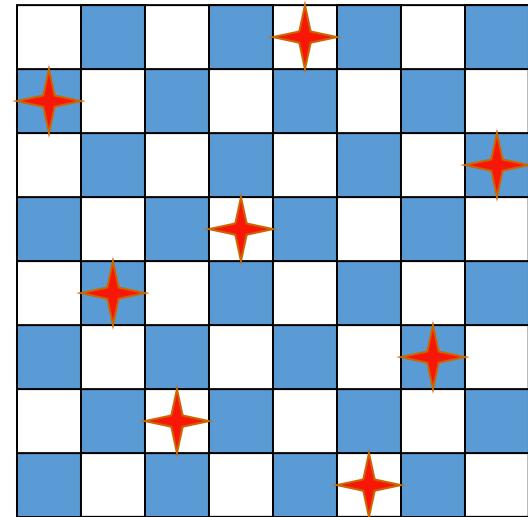
只要还有冲突没有消除，
即便 $f(x_{\text{current}}) = f(x_{\text{neighbor}})$ ，
仍要继续执行算法



冲突对数 1



冲突对数 1

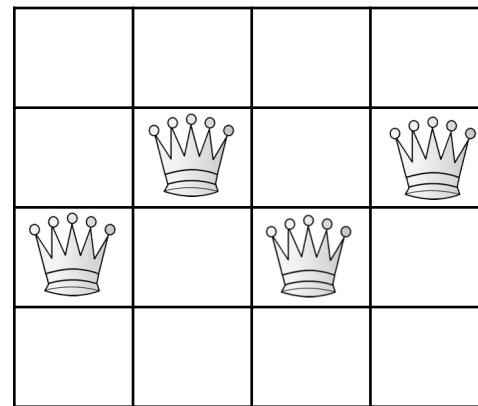


冲突对数 0

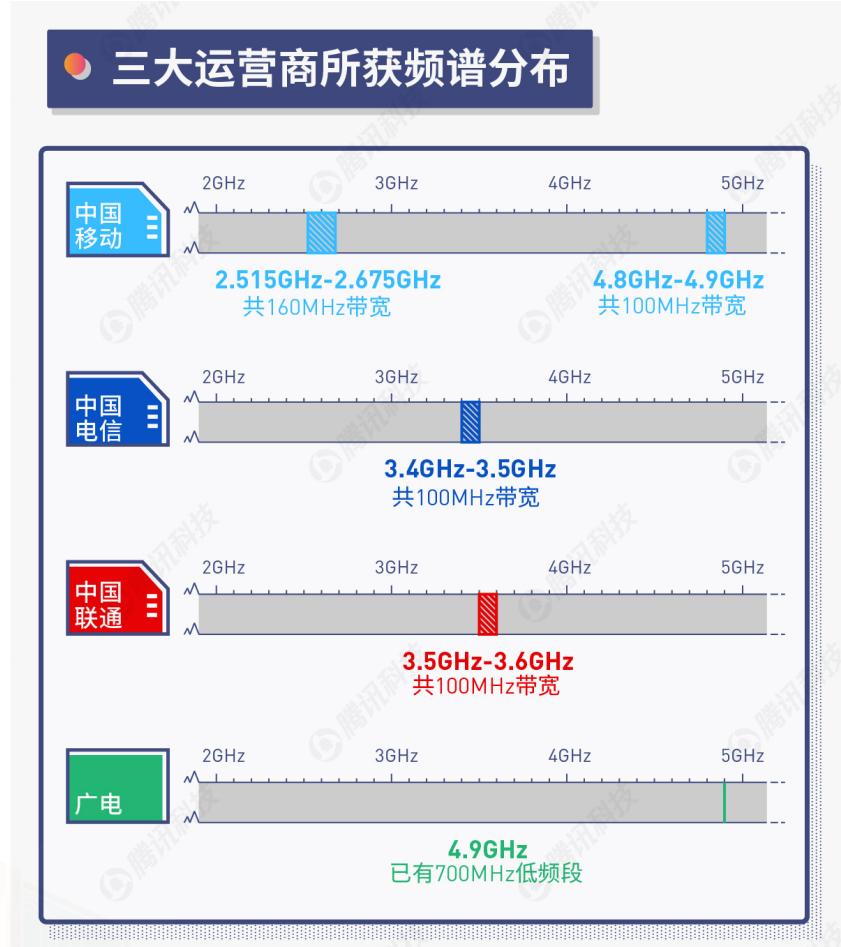
约束满足问题

皇后问题的解的空间： n^n

即便面对百万规模的皇后问题，运用最小冲突启发式算法（在改进初始化方式的情况下）能以平均50步左右的效率求到满足约束的状态



约束满足问题



基站

约束满足问题



频谱分配问题

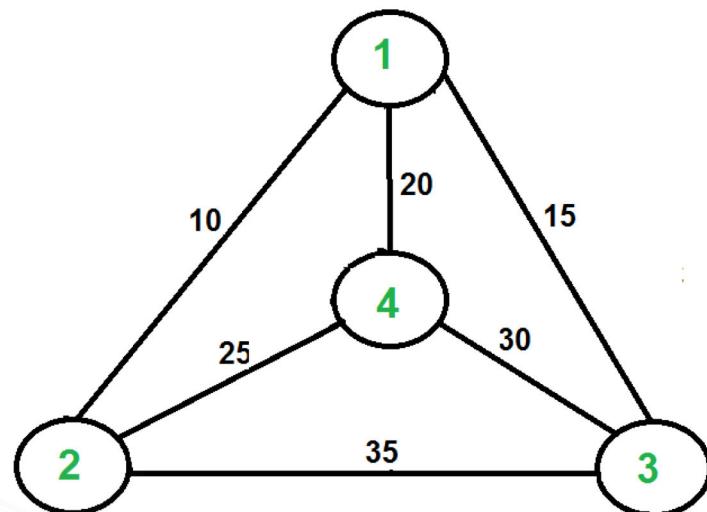
约束满足问题



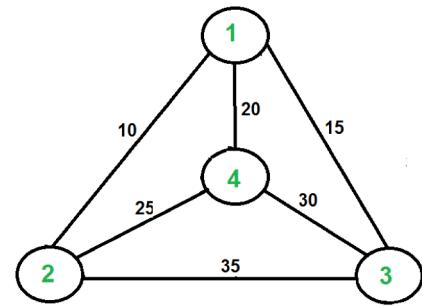
频谱分配问题

旅行商问题

旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）



旅行商问题

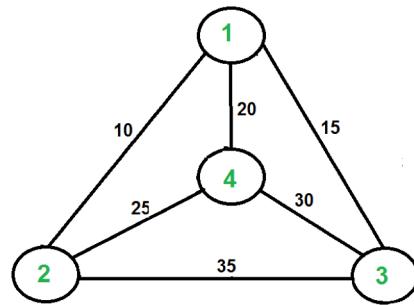


旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

登山搜索（返回最大化问题的局部最优解）

- 0 初始化当前解 $x_{current}$
- 1 寻找当前解的所有邻域解 **如何定义邻域？**
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 $x_{neighbor}$
- 3 若 $f(x_{current}) \geq f(x_{neighbor})$, 结束搜索; 否则, $x_{current} \leftarrow x_{neighbor}$, 返回1

旅行商问题



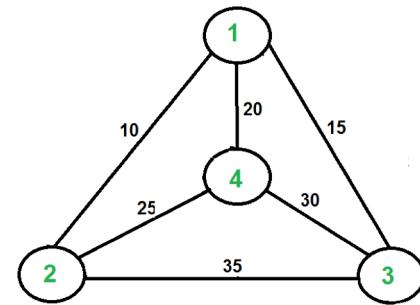
旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

登山搜索（返回最大化问题的局部最优解）

- 0 初始化当前解 $x_{current}$
- 1 寻找当前解的所有邻域解
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 $x_{neighbor}$
- 3 若 $f(x_{current}) \geq f(x_{neighbor})$, 结束搜索; 否则, $x_{current} \leftarrow x_{neighbor}$, 返回1

可以通过交换当前路径中任意两个**相邻**城市的位置生成邻域解

旅行商问题

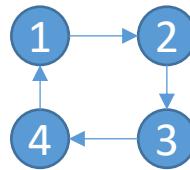


旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

登山搜索（返回最大化问题的局部最优解）

- 0 初始化当前解 $x_{current}$
- 1 寻找当前解的所有邻域解
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 $x_{neighbor}$
- 3 若 $f(x_{current}) \geq f(x_{neighbor})$, 结束搜索; 否则, $x_{current} \leftarrow x_{neighbor}$, 返回1

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$



$2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2$
 $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$
 $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$
 $4 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4$

旅行商问题

四个实验场景



a) 15 cities



a) 25 cities



b) 100 cities



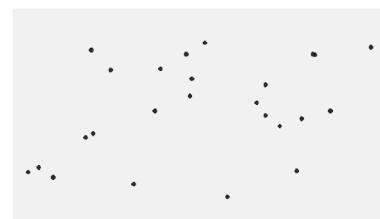
d) 25 cities alternative

旅行商问题

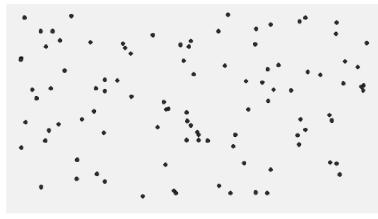
四个实验场景



a) 15 cities



a) 25 cities



b) 100 cities



d) 25 cities alternative

四个实验场景下三个算法的结果

Table 1: Experimental Results. Smallest costs are in bold.

Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	6200
	25 Cities	0.188	9221
	25 Cities A	0.190	13362
	100 Cities	0.671	47231
Evolutionary Algorithm 进化（遗传）算法	15 Cities	0.116	6567
	25 Cities	0.335	10138
	25 Cities A	0.328	12044
	100 Cities	1.268	52499

旅行商问题

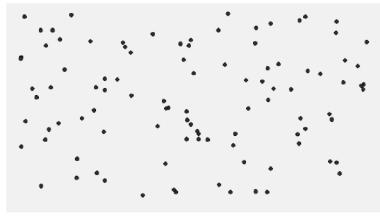
四个实验场景



a) 15 cities



a) 25 cities



b) 100 cities



d) 25 cities alternative

四个实验场景下三个算法的结果

Table 1: Experimental Results. Smallest costs are in bold.

Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	6200
	25 Cities	0.188	9221
	25 Cities A	0.190	13362
	100 Cities	0.671	47231
Evolutionary Algorithm 进化（遗传）算法	15 Cities	0.116	6567
	25 Cities	0.335	10138
	25 Cities A	0.328	12044
	100 Cities	1.268	52499

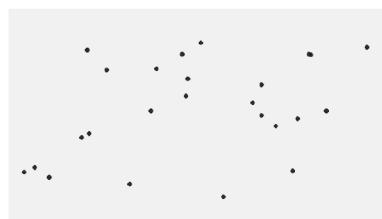
登山搜索最快、但性能不如后两者

旅行商问题

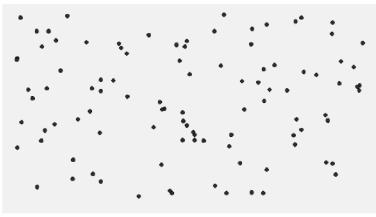
四个实验场景



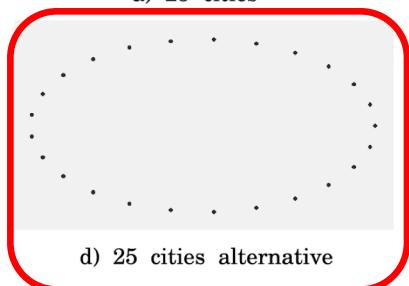
a) 15 cities



a) 25 cities



b) 100 cities



d) 25 cities alternative

四个实验场景下三个算法的结果

Table 1: Experimental Results. Smallest costs are in bold.

Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	6200
	25 Cities	0.188	9221
	25 Cities A	0.190	13362
	100 Cities	0.671	47231
Evolutionary Algorithm 进化（遗传）算法	15 Cities	0.116	6567
	25 Cities	0.335	10138
	25 Cities A	0.328	12044
	100 Cities	1.268	52499

旅行商问题

四个实验场景



四个实验场景下三个算法的结果

Table 1: Experimental Results. Smallest costs are in bold.

Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	6200
	25 Cities	0.188	9221
	25 Cities A	0.190	13362
	100 Cities	0.671	47231
Evolutionary Algorithm 进化（遗传）算法	15 Cities	0.116	6567
	25 Cities	0.335	10138
	25 Cities A	0.328	12044
	100 Cities	1.268	52499

Tracing the ellipse will obviously yield the best result. However, the local search algorithms discussed in this paper does not have access to this intuition. Another characteristic of this layout is the fact that finding near optimal solutions are more difficult compared to random layouts, as there exists many near optimal solutions in other layouts, whereas in this layout the optimal solution seems to be isolated from solutions.

旅行商问题

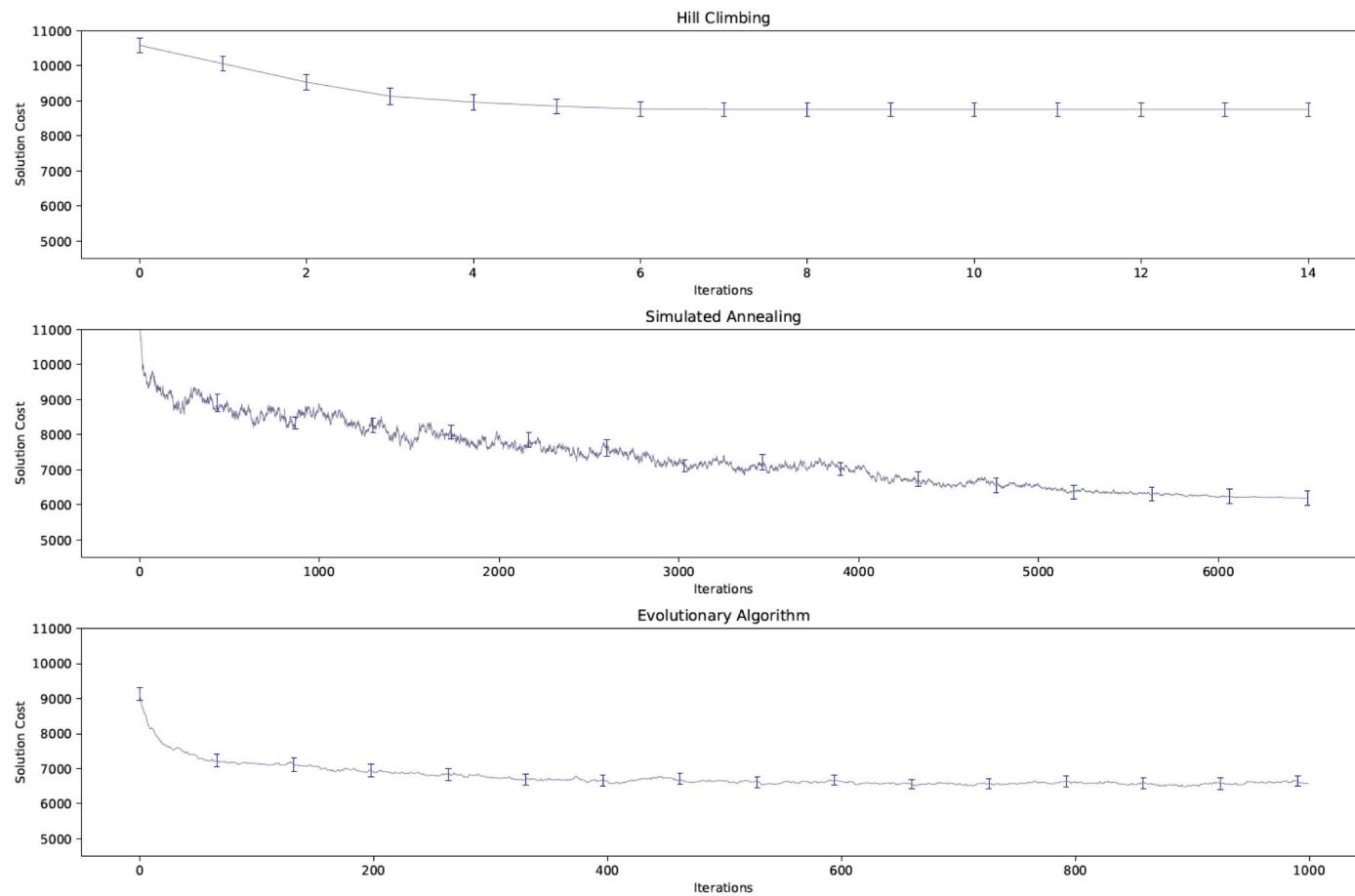
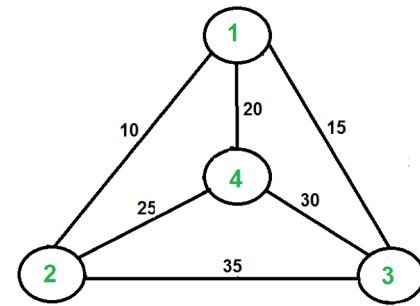


Figure 2: Performance curves for the three algorithms for a TSP with 15 cities.

旅行商问题



旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

登山搜索（返回最大化问题的局部最优解）

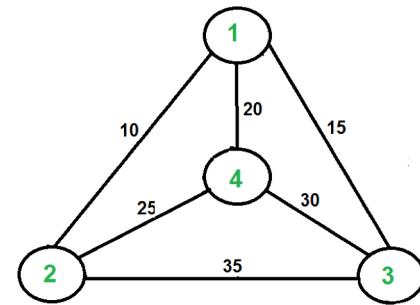
- 0 初始化当前解 $x_{current}$
- 1 寻找当前解的所有邻域解
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 $x_{neighbor}$
- 3 若 $f(x_{current}) \geq f(x_{neighbor})$, 结束搜索; 否则, $x_{current} \leftarrow x_{neighbor}$, 返回1

可以通过交换当前路径中任意两个**相邻城市**的位置生成邻域解

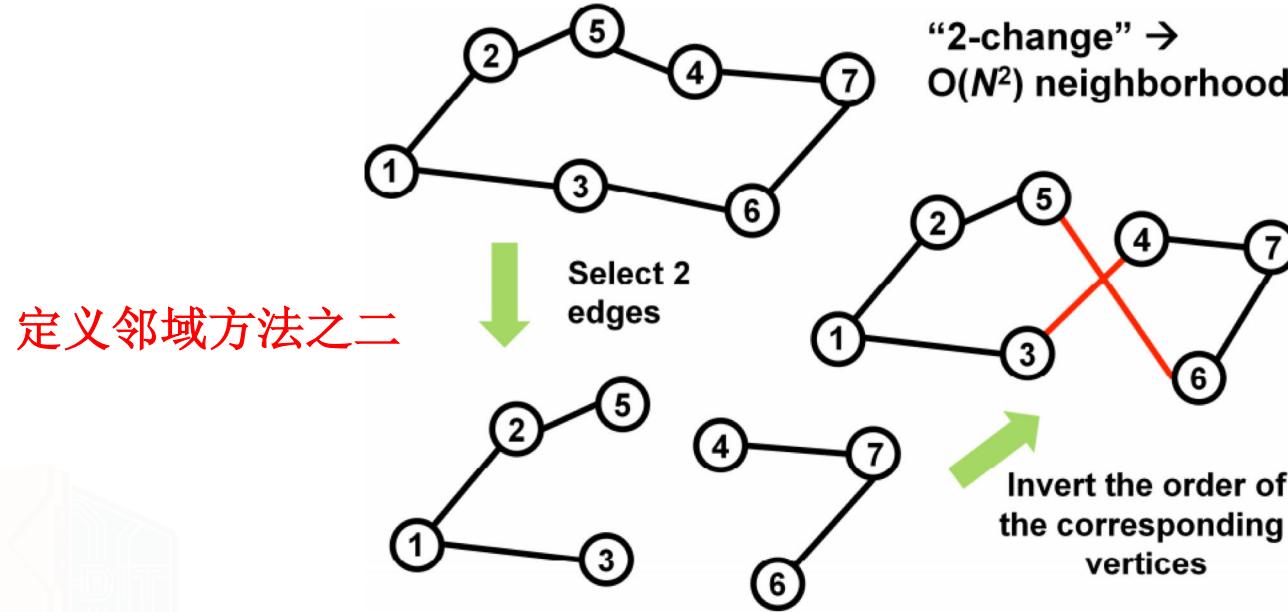


有没有其它定义邻域的方法？

旅行商问题

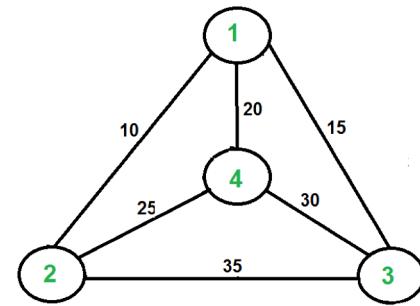


旅行商问题：给定 n 个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）



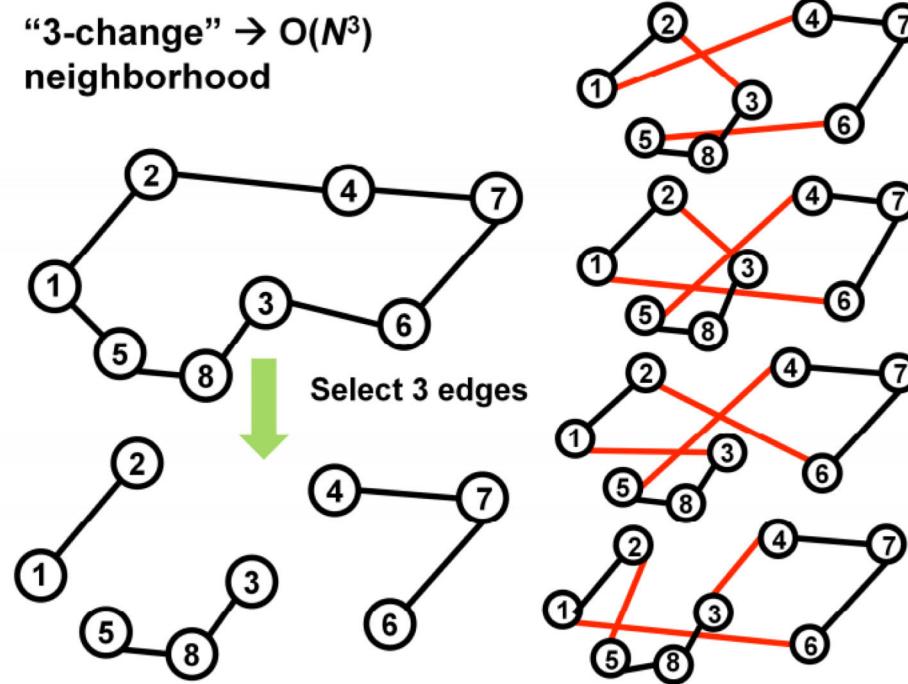
图片取自Doina Precup COMP-424<Artificial Intelligence>

旅行商问题

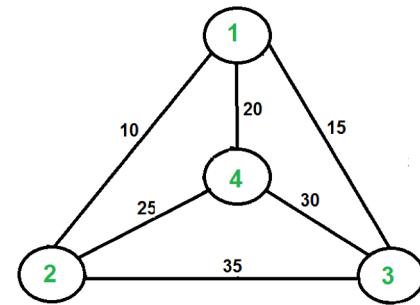


旅行商问题：给定 n 个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

定义邻域方法之三



旅行商问题



旅行商问题：给定 n 个城市，任意两城市间有路相连且路程距离已知。**找到总长度最小的路径**（从一个城市出发、不重复的遍历所有城市并回到起点）

- A smaller neighborhood means fewer neighbors to evaluate (so cheaper computation, but possibly worse solutions)
- A bigger neighborhood means more computation, but maybe fewer local optima, so better final result
- Defining the set of neighbors is a *design choice* (like choosing the heuristic for A^*) and has a crucial impact on performance

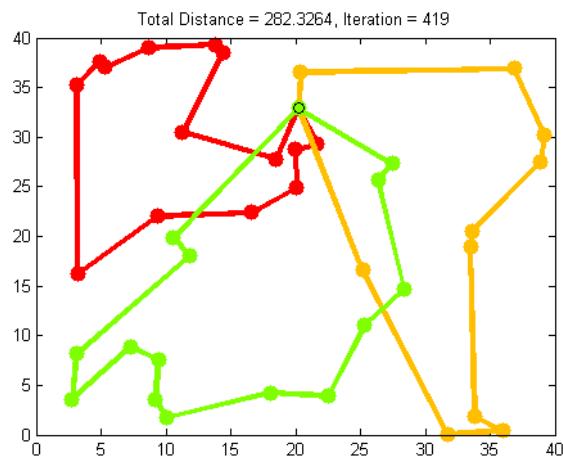
旅行商问题

➤ 求解旅行商问题在现实生活中有什么应用？



旅行商问题

- 求解旅行商问题在现实生活中有什么应用？
- 旅行商问题和外卖员取餐问题有什么不同？
 - 取餐送餐：对先后顺序有要求
 - 多个外卖员



旅行商问题



“从算法数据角度剖析，德国数学博士揭秘外卖骑手困局的本质”

<https://www.bilibili.com/video/BV1SA411E7Rx?from=search&seid=9335264868448425252>

登山搜索的改进

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

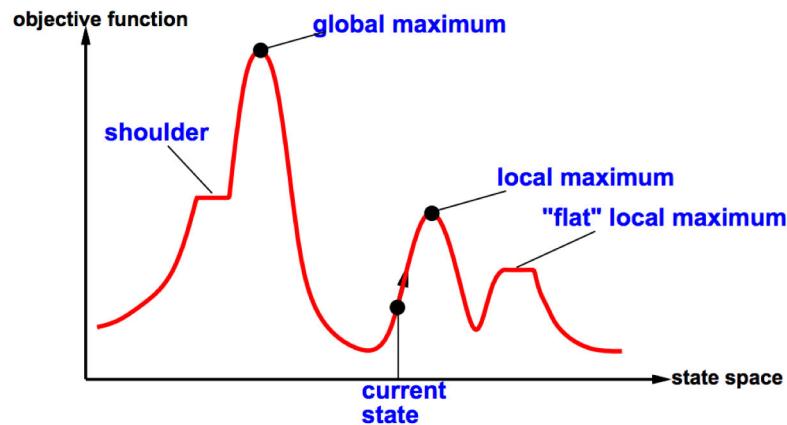
登山搜索的局限

算法最后输出的解不一定是全局最优解，那么具体有哪些可能？

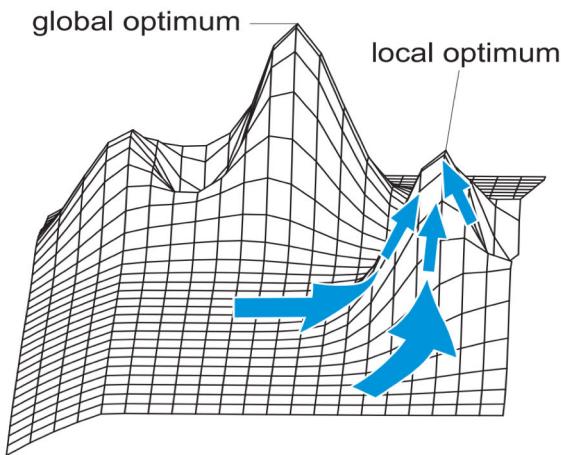


登山搜索的局限

算法最后输出的解不一定是全局最优解，那么具体有哪些可能？



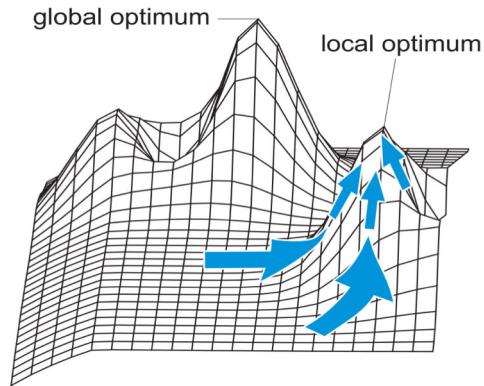
单个变量的情况



两个变量的情况

改进登山搜索

如何改进登山搜索？



登山搜索（返回最大化问题的局部最优解）

- 0 初始化当前解 x_{current}
- 1 寻找当前解的所有邻域解
- 2 得到对应目标函数值 $f(\cdot)$ 最高的邻域解 x_{neighbor}
- 3 若 $f(x_{\text{current}}) \geq f(x_{\text{neighbor}})$, 结束搜索; 否则, $x_{\text{current}} \leftarrow x_{\text{neighbor}}$, 返回1

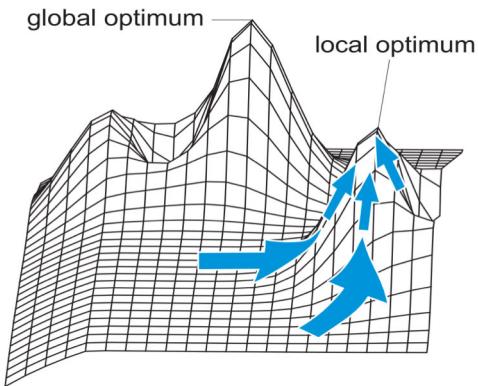
改进登山搜索

如何改进登山搜索？

- 首选登山搜索
- 随机登山搜索
- 侧向移动登山搜索
- 随机重启登山搜索

首选登山搜索

思路：加入随机性，跳出局部最优



首选登山搜索 (First-choice hill climbing)

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 若 $f(x_{\text{current}}) \geq f(x_{\text{neighbor}})$, 结束搜索; 否则, $x_{\text{current}} \leftarrow x_{\text{neighbor}}$, 返回1

下一步不一定会跳到邻域中质量最高的解

当邻域的解数目很多时, 效果比较好

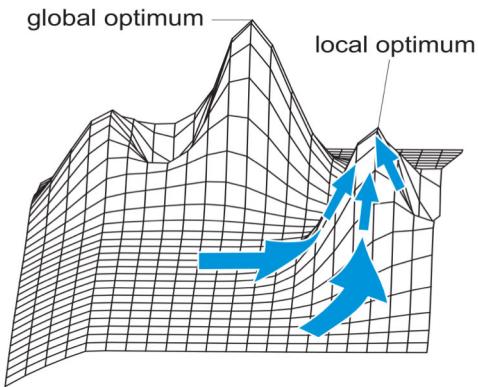
改进登山搜索

如何改进登山搜索？

- 首选登山搜索
- 随机登山搜索
- 侧向移动登山搜索
- 随机重启登山搜索

随机登山搜索

思路：加入随机性，跳出局部最优

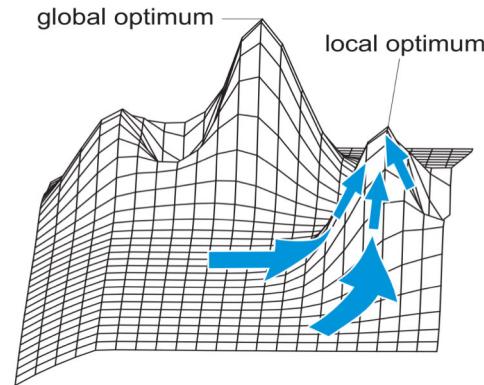


随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

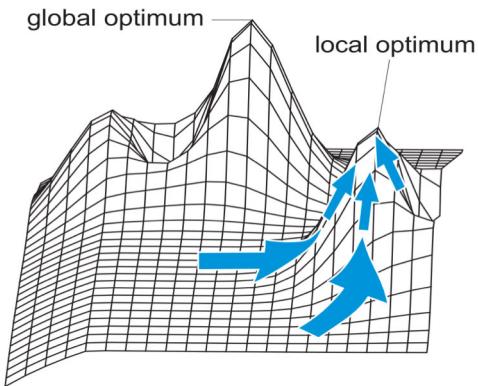


怎么推导而来的？

如果每次随机选的邻居点数目不止1个，如K个，该如何计算概率？

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1 + e^{\frac{f(x_{\text{current}}) - f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

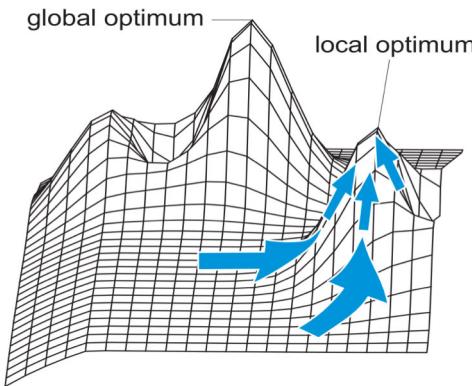


怎么推导而来的？

$$\frac{1}{1 + e^{\frac{f(x_{\text{current}}) - f(x_{\text{neighbor}})}{T}}} = \frac{e^{\frac{f(x_{\text{neighbor}})}{T}}}{e^{\frac{f(x_{\text{neighbor}})}{T}} + e^{\frac{f(x_{\text{current}})}{T}}}$$

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

替换的概率

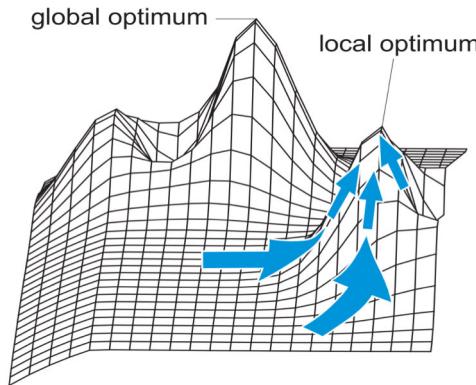
$$\frac{e^{\frac{f(x_{\text{neighbor}})}{T}}}{e^{\frac{f(x_{\text{neighbor}})}{T}} + e^{\frac{f(x_{\text{current}})}{T}}}$$

不替换的概率

$$\frac{e^{\frac{f(x_{\text{current}})}{T}}}{e^{\frac{f(x_{\text{neighbor}})}{T}} + e^{\frac{f(x_{\text{current}})}{T}}}$$

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

替换的概率

$$\frac{e^{\frac{f(x_{\text{neighbor}})}{T}}}{e^{\frac{f(x_{\text{neighbor}})}{T}} + e^{\frac{f(x_{\text{current}})}{T}}}$$

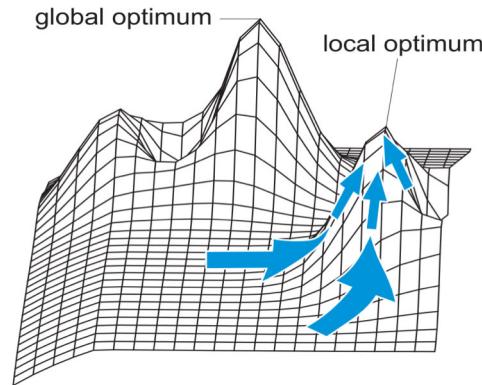
不替换的概率

$$\frac{e^{\frac{f(x_{\text{current}})}{T}}}{e^{\frac{f(x_{\text{neighbor}})}{T}} + e^{\frac{f(x_{\text{current}})}{T}}}$$

用指数形式而非 $\frac{f(x_{\text{neighbor}})}{f(x_{\text{neighbor}})+f(x_{\text{current}})}$ 的原因？参数 T 的作用？

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1



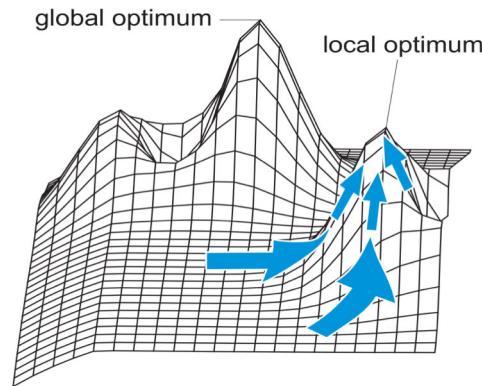
如果每次随机选的邻居点数目不止1个，如K个，该如何计算概率？

$x_{\text{current}}, x_{\text{neighbor}1}, x_{\text{neighbor}2}$

$f(x_{\text{current}}), f(x_{\text{neighbor}1}), f(x_{\text{neighbor}2})$

随机登山搜索

思路：加入随机性，跳出局部最优



随机登山搜索

- 0 初始化当前解 x_{current}
- 1 随机选择一个邻域解 x_{neighbor}
- 2 以概率 $\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$ 令 $x_{\text{current}} \leftarrow x_{\text{neighbor}}$
- 3 若算法终止条件不满足，返回1

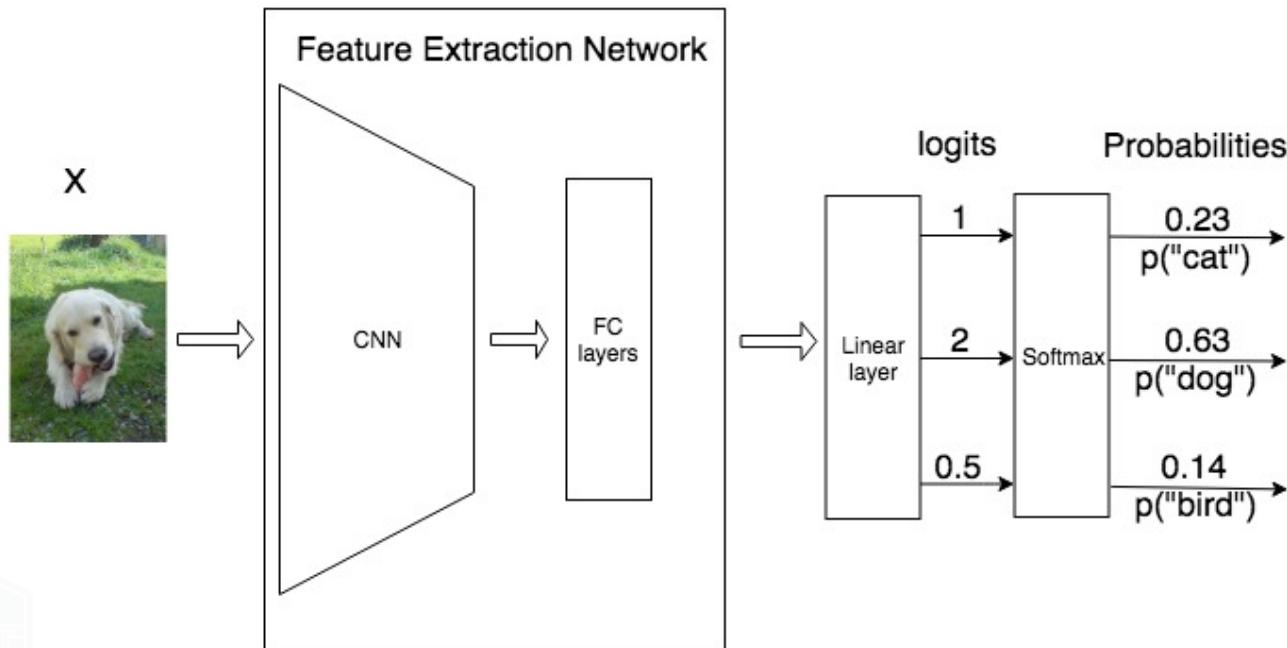
The standard (unit) softmax function $\sigma : \mathbb{R}^K \rightarrow [0, 1]^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$

随机登山搜索

The standard (unit) softmax function $\sigma : \mathbb{R}^K \rightarrow [0, 1]^K$ is defined by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \text{ for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K$$



Softmax函数在机器学习中常用，如分类预测

改进登山搜索

如何改进登山搜索？

- 首选登山搜索
- 随机登山搜索
- 侧向移动登山搜索
- 随机重启登山搜索

侧向移动登山搜索

侧向移动（**Sideways moves**）

如果所有邻域内的解都不比当前解更好，
用一个和当前解同样质量的邻域解代替当前解

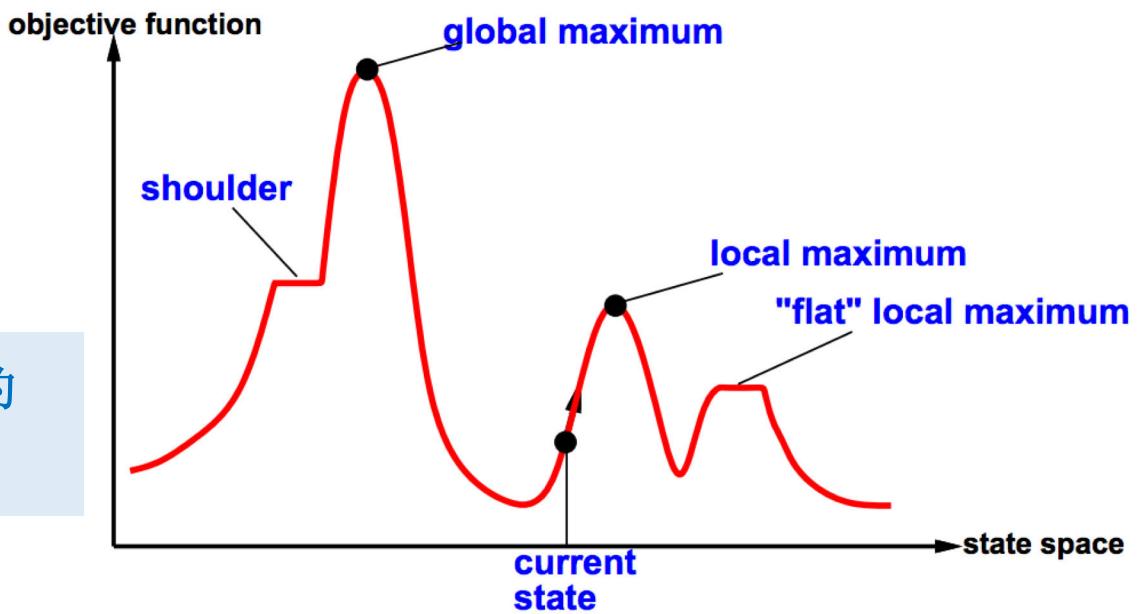
侧向移动登山搜索

侧向移动（Sideways moves）

如果所有邻域内的解都不比当前解更好，
用一个和当前解同样质量的邻域解代替当前解

用于应对哪种情况？

需要设置侧向移动次数的
上限M，避免循环



改进登山搜索

如何改进登山搜索？

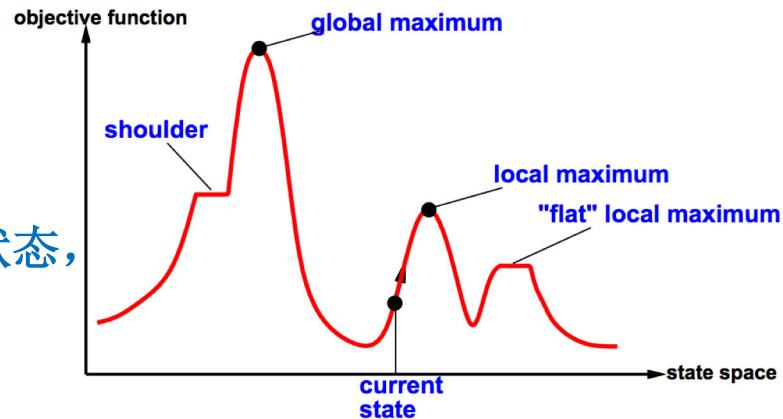
- 首选登山搜索
- 随机登山搜索
- 侧向移动登山搜索
- 随机重启登山搜索

随机重启登山搜索

随机重启 (Random restart)

前后运行多次登山搜索，每次随机选择初始状态，在每次搜索中，如遇到如下情况则中止搜索：

- 停滞在某个解；
- 持续没有改进。



后两种改进的效果对比

- Hill-climbing can solve large instances of n -queens ($n = 10^6$) in a few (ms)seconds
- 8 queens statistics:
 - State space of size ≈ 17 million
 - Starting from random state, steepest-ascent hill climbing solves 14% of problem instances
 - It takes 4 steps on average when it succeeds, 3 when it gets stuck

原始登山搜索（无侧向移动）



Amazing

sideways moves (M):
 $M=100 \rightarrow 94\%$ solved instances
for the 8-queens!
21 steps avg. on success
64 steps avg. on “failure”

random restarts:
100% solved instances
28 steps avg.

随机重启登山搜索

侧向移动登山搜索

局部束搜索

Part1 如何从复杂度的角度衡量算法好坏？

Part2 针对一些有特定形式的最优化问题，如何找到最优解？

Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？

局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法

Part4 针对较复杂的多目标最优化问题，如何找到较好解？

局部束搜索

登山搜索的局限性：每个时刻只存储一个解

局部束搜索（Local beam Search）：同时存储 K 个解

局部束搜索

局部束搜索

- 0 初始化 K 个当前解： $x_{\text{current},1}, x_{\text{current},2}, \dots, x_{\text{current},K}$
- 1 寻找 K 个当前解的所有邻域解
- 2 计算邻域解中最高目标函数值
- 3 若结束条件满足，结束算法；否则，选择 K 个最好的解作为新当前解，返回1

当 $K = 1$ 时，局部束搜索等价于原始登山搜索



局部束搜索是不是等价于 K 次随机重启的登山算法？

局部束搜索

局部束搜索是不是等价于 K 次随机重启的登山算法？不等价

假设 $K = 3$
初始解位置



局部束搜索

局部束搜索迭代一次后解位置



随机重启的登山算法（分别）迭代一次后解位置

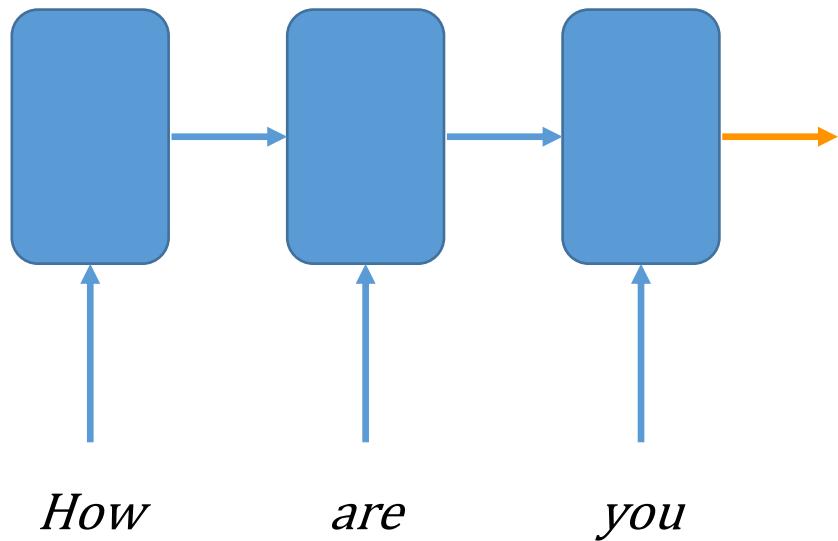


各个点之间共享信息（代价是存储空间）

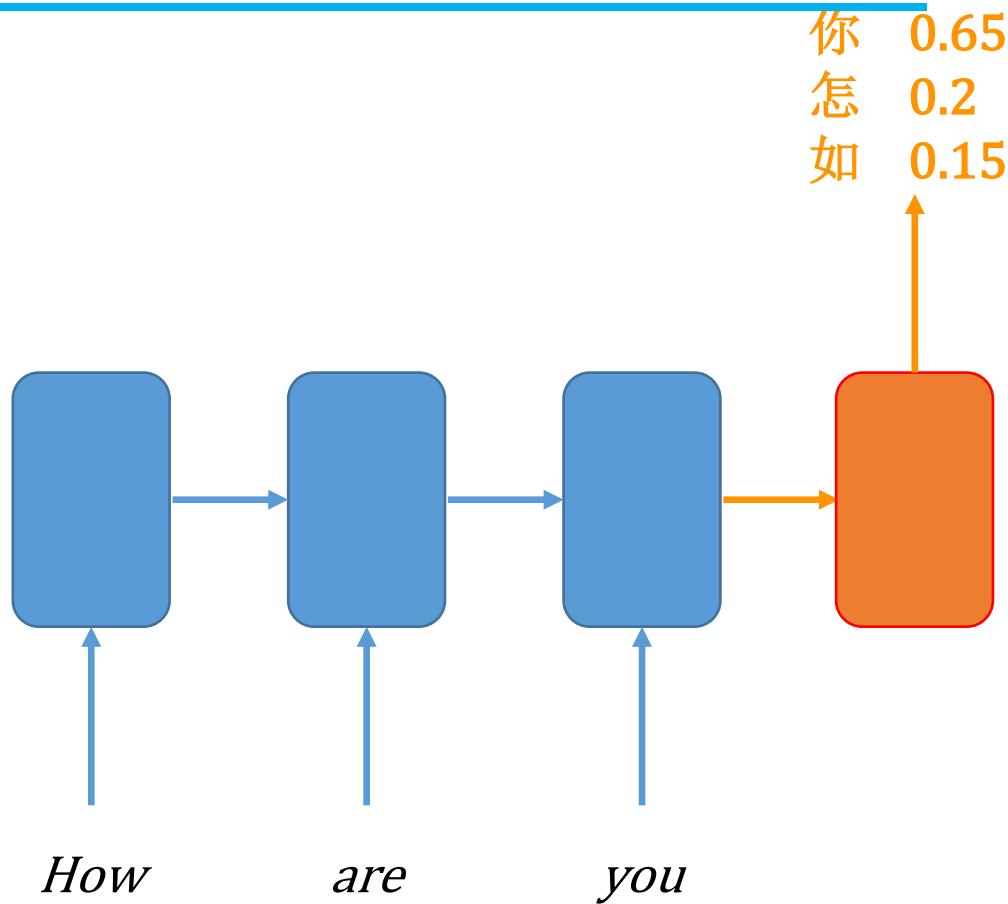
可以先后执行，不占额外的存储空间

束搜索在机器翻译中的应用

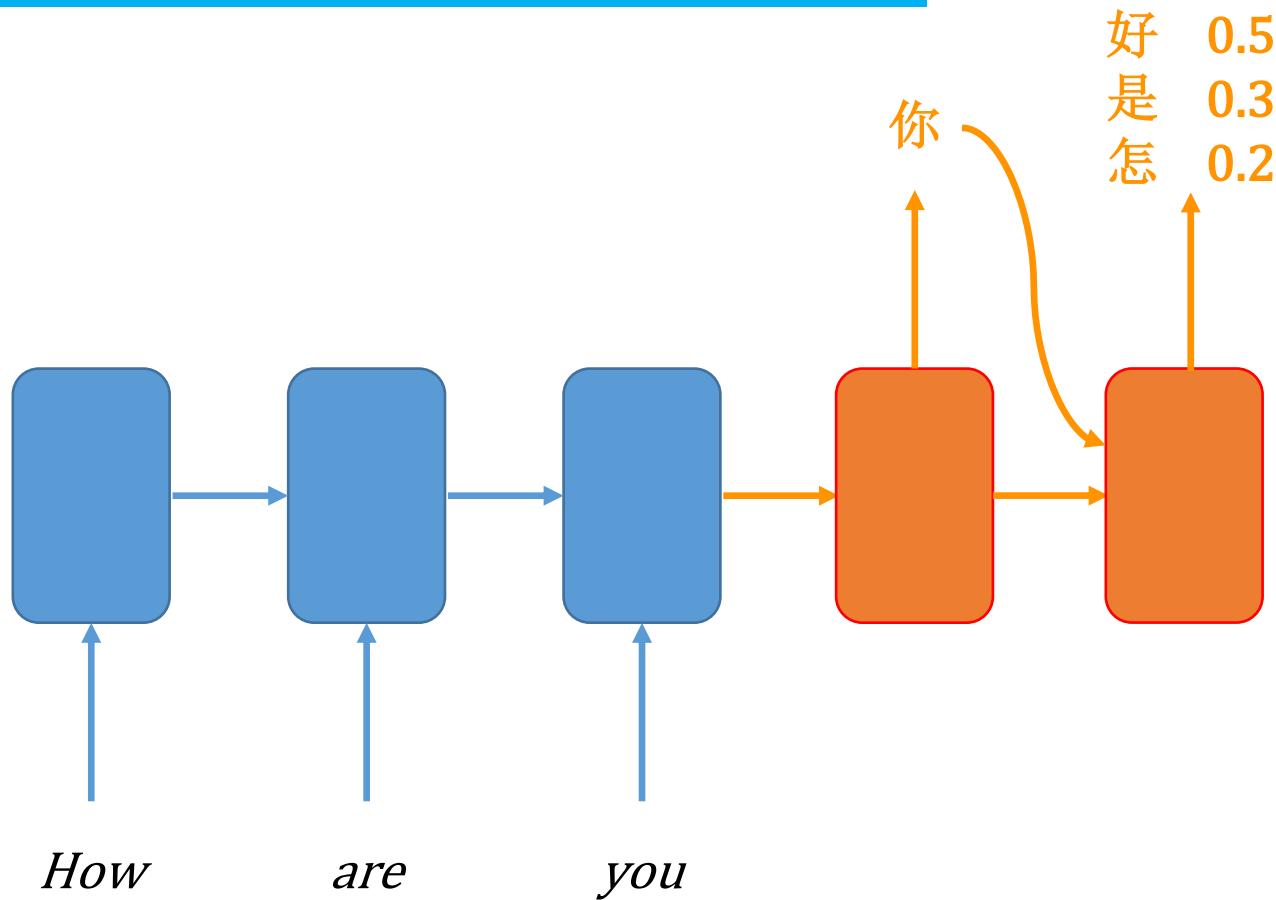
基于循环神经网络（RNN）的原始翻译方式：



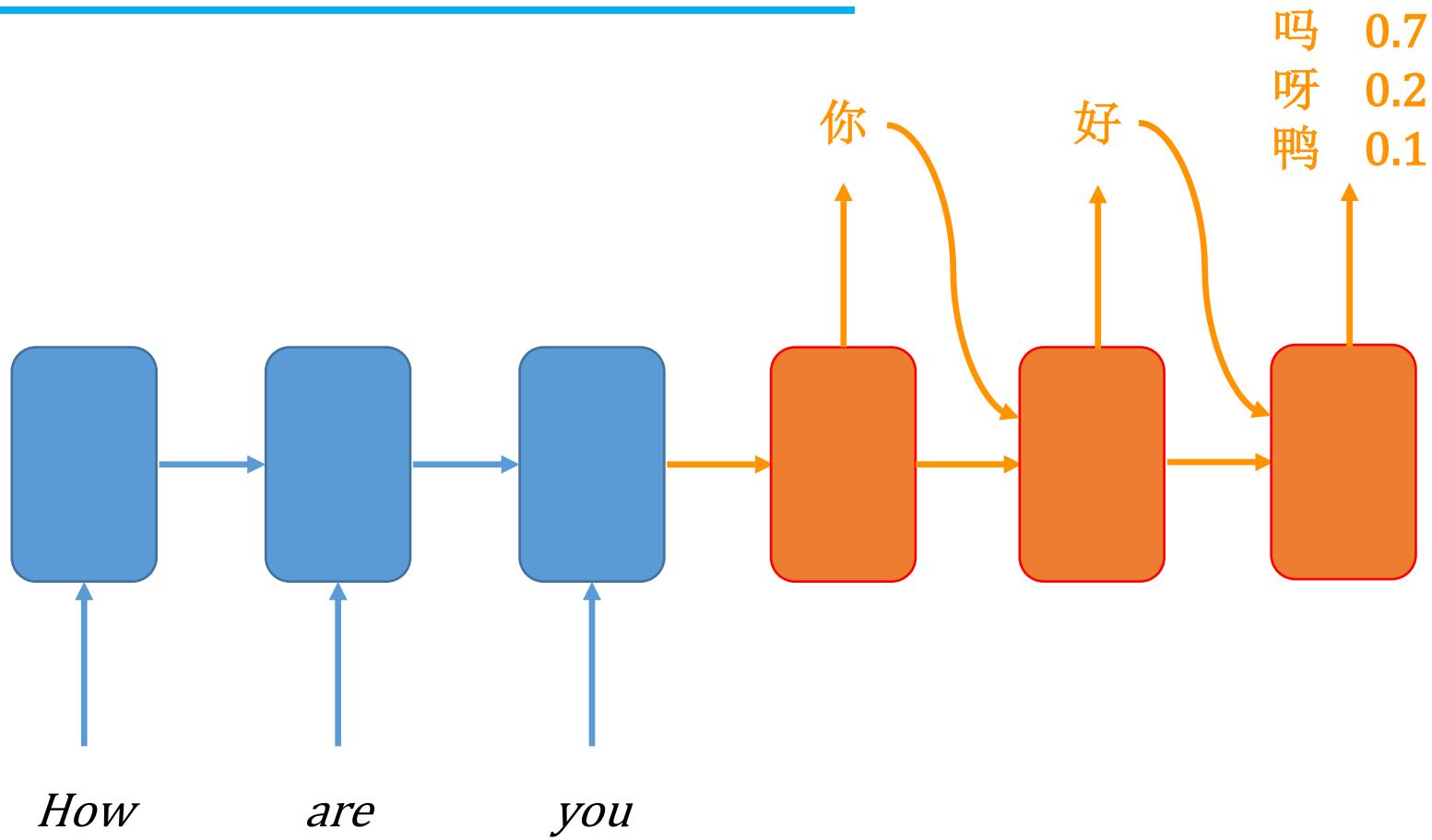
束搜索在机器翻译中的应用



束搜索在机器翻译中的应用



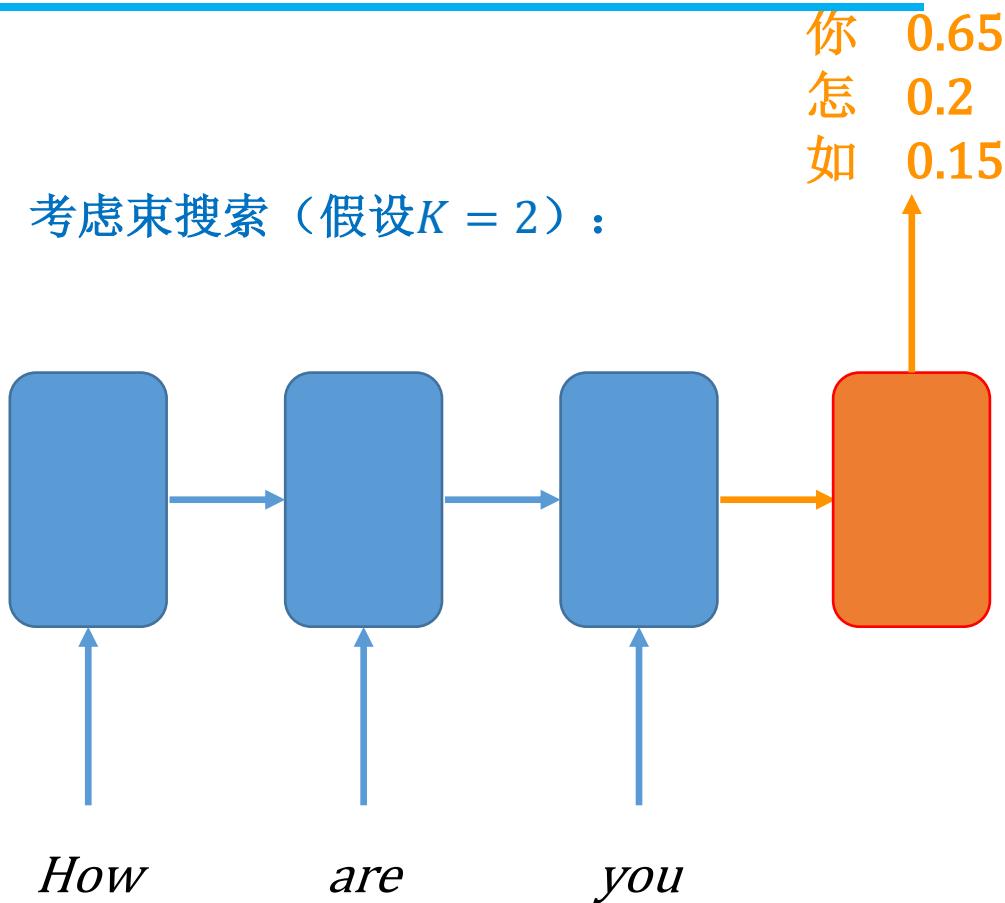
束搜索在机器翻译中的应用



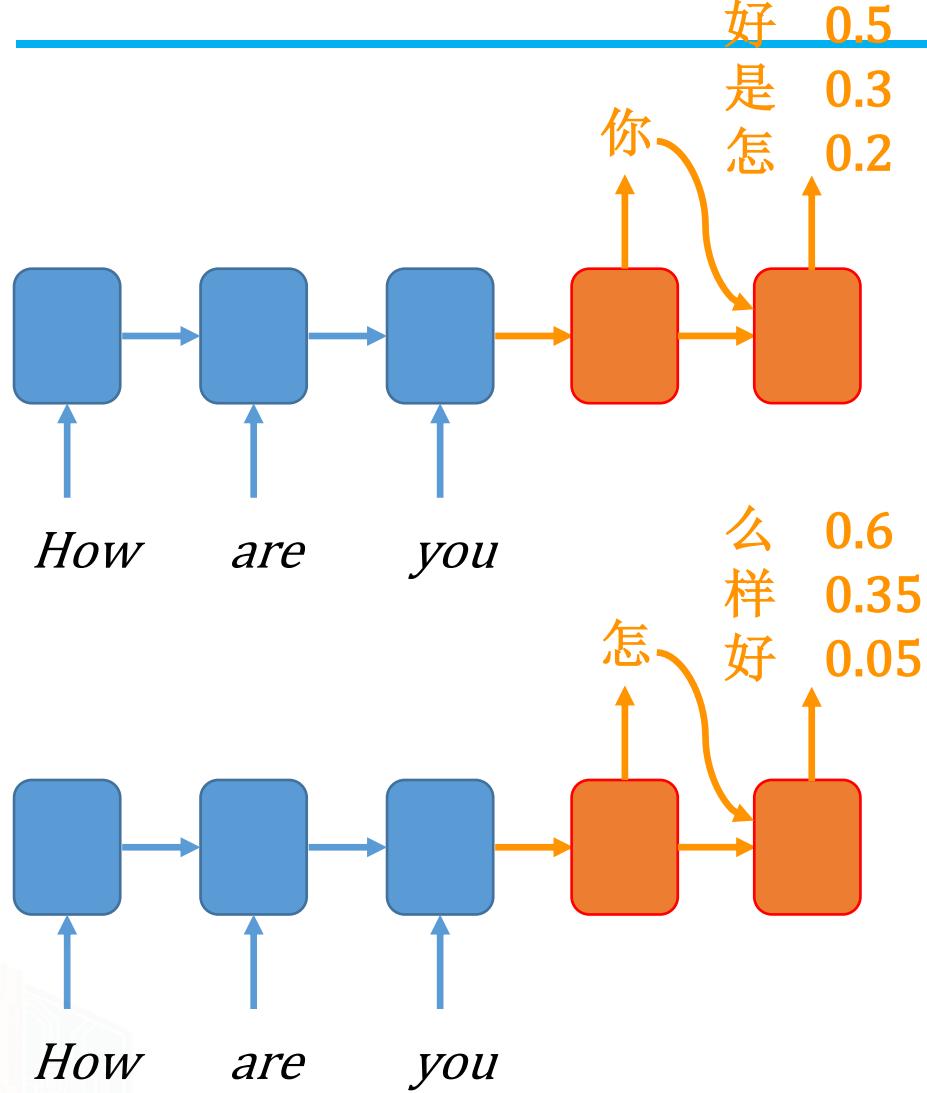
非常依赖对第一个字翻译的准确度

束搜索在机器翻译中的应用

考虑束搜索（假设 $K = 2$ ）：



束搜索在机器翻译中的应用



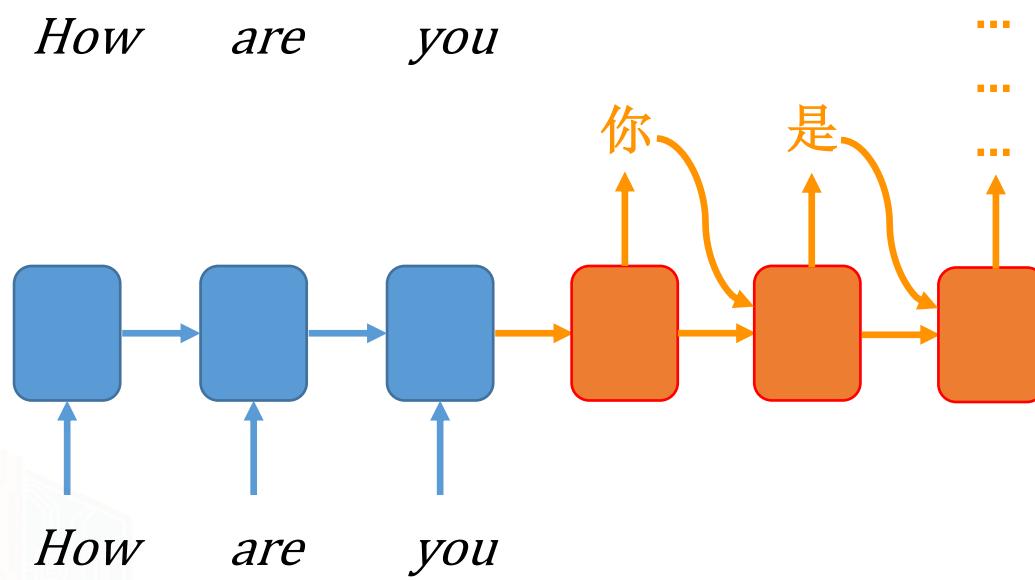
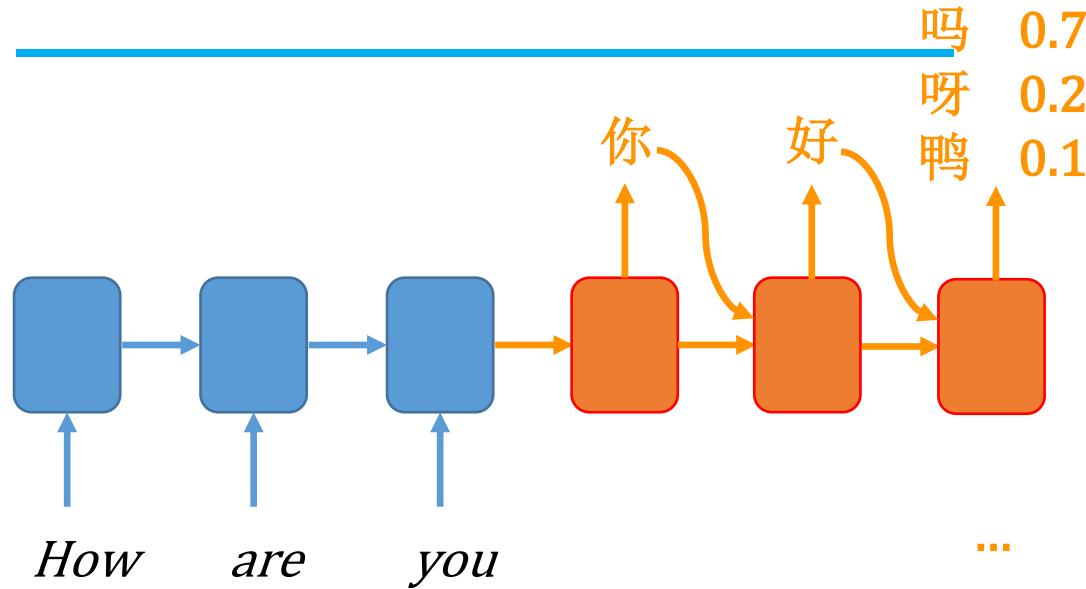
计算：

$P(\text{你好} | \text{How are you})$
 $P(\text{你是} | \text{How are you})$
 $P(\text{怎样} | \text{How are you})$
 $P(\text{怎么} | \text{How are you})$
 $P(\text{怎样} | \text{How are you})$
 $P(\text{您好} | \text{How are you})$

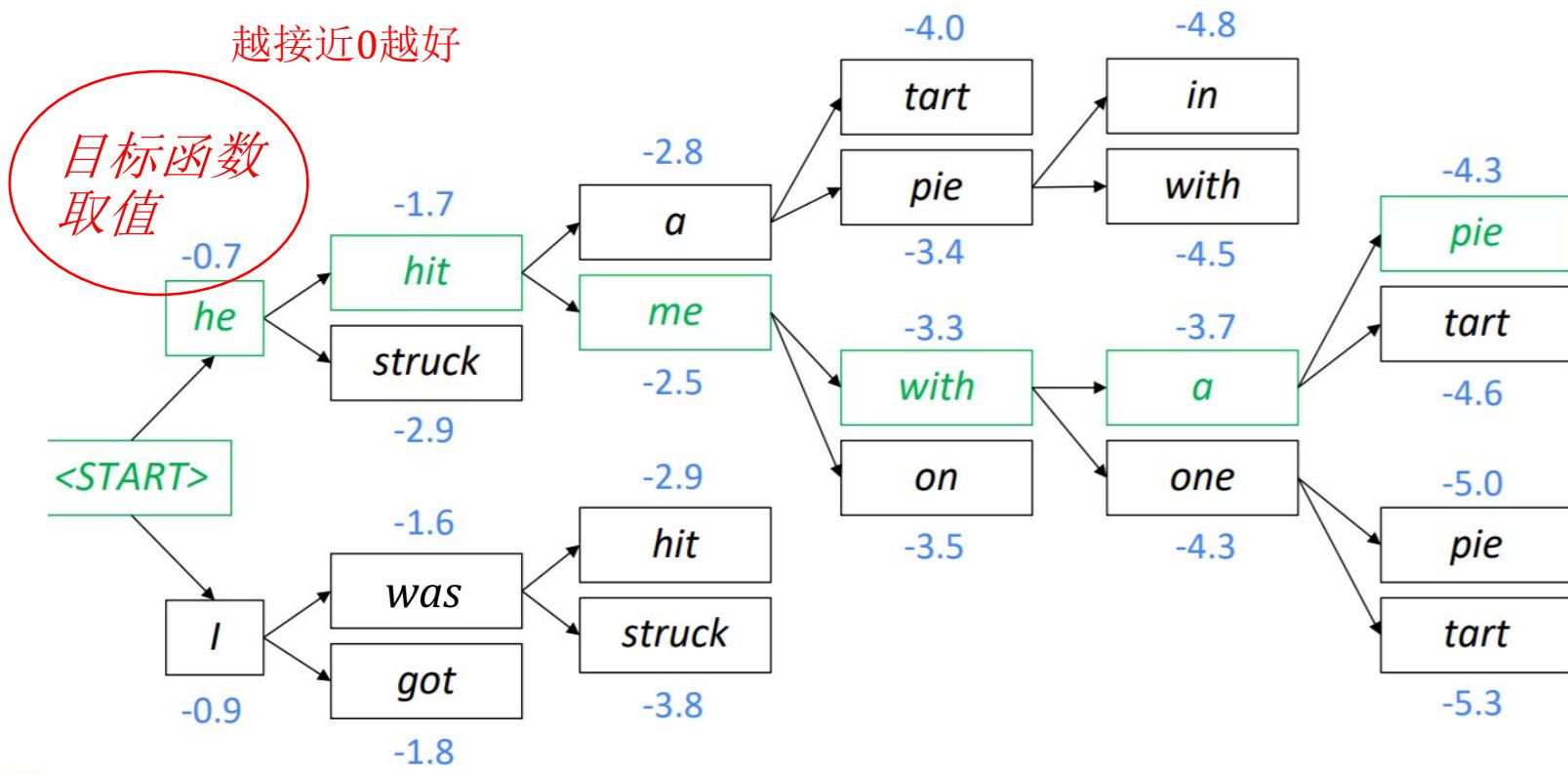
选出其中概率最高的两项

假设前两项概率最高

束搜索在机器翻译中的应用



束搜索在机器翻译中的应用



束搜索 ($K = 2$)

取自网络例子

本讲小结



有约束凸优化问题、KKT条件、内点法



局部搜索之登山搜索、局部束搜索

主要参考资料

Daniel P. Palomar <ELEC5470/IEDA6100A - Convex Optimization> Slides

Stephen Boyd, Lieven Vandenberghe <Convex Optimization> Book

neos-guide.org <Guide to Optimization>

杨翠娥、王源 <进化计算PK数学优化> 文章

LSI-FIB-UPC <Artificial Intelligence> Slides

Kagan Tumer <ROB537-Learning Based Control> Lecture Notes

Doina Precup <COMP-424 Artificial Intelligence> Slides

留德华叫兽“从算法数据角度剖析，德国数学博士揭秘外卖骑手困局的本质”视频

谢谢！

