第六章

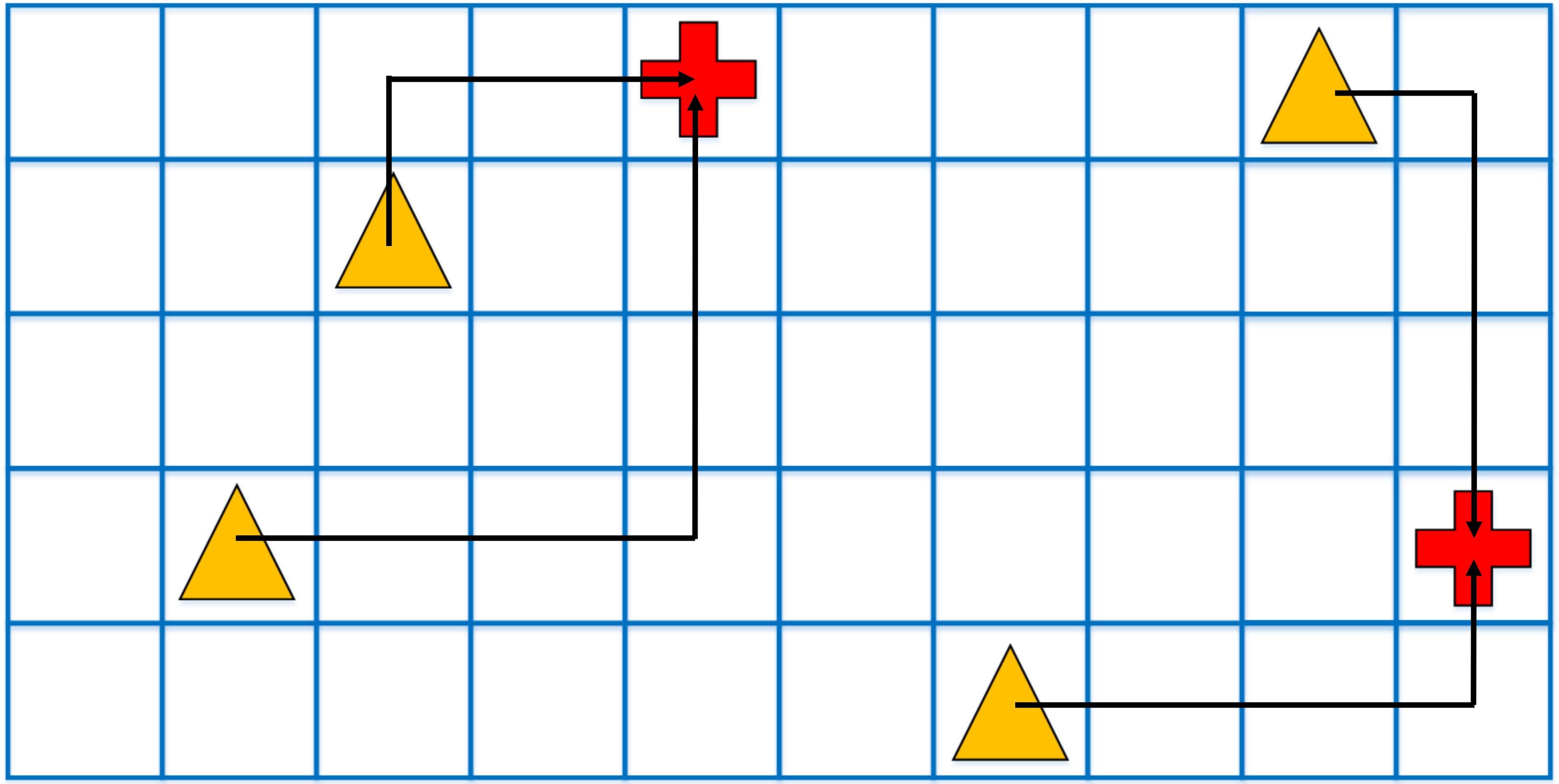
优化算法

优化的概念

- **优化 (optimization)**
  - ✓ 从一组选项（解）中选出最佳选项（最优解）

爬山算法 (hill climbing)

- **医院建造问题**
  - ✓ 在下图中建造2所医院
  - ✓ 如何选取建造的位置，使得4间房子到医院的距离之和最短

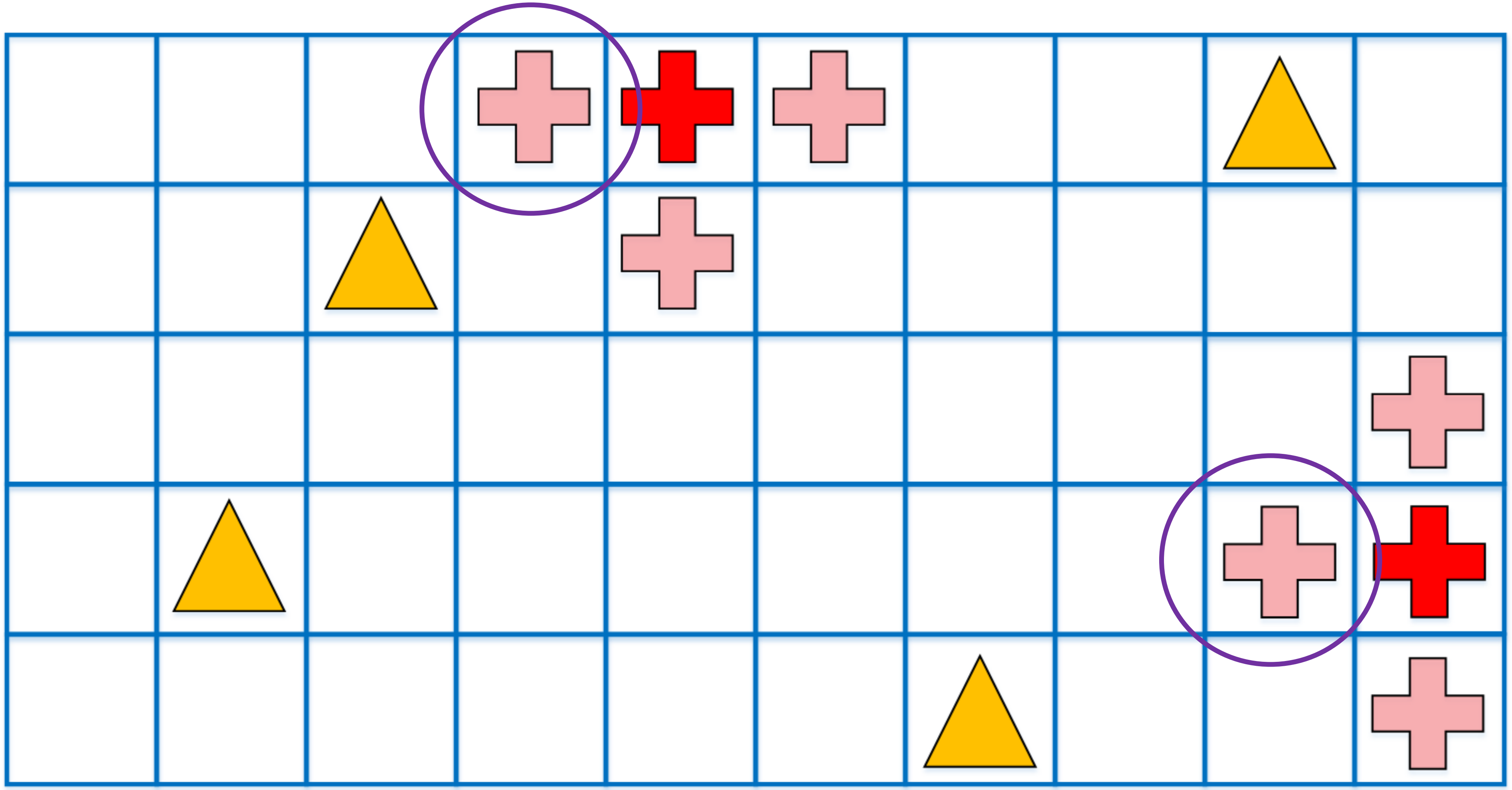


曼哈顿距离之和=

$3+6+4+4=17$

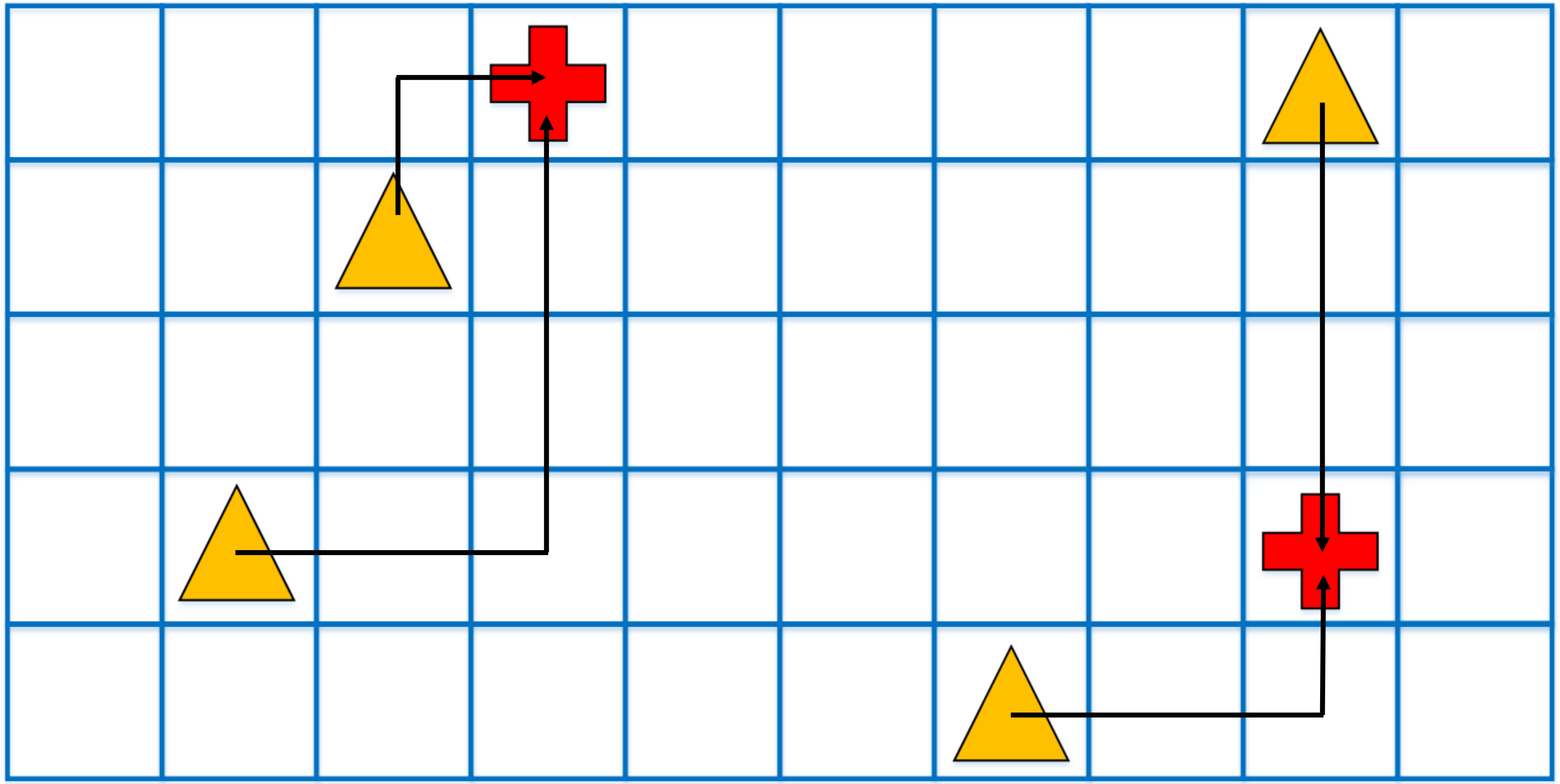
爬山算法 (hill climbing)

- **医院建造问题**
  - ✓ 在医院当前位置的所有相邻位置中，选取一个最优的位置



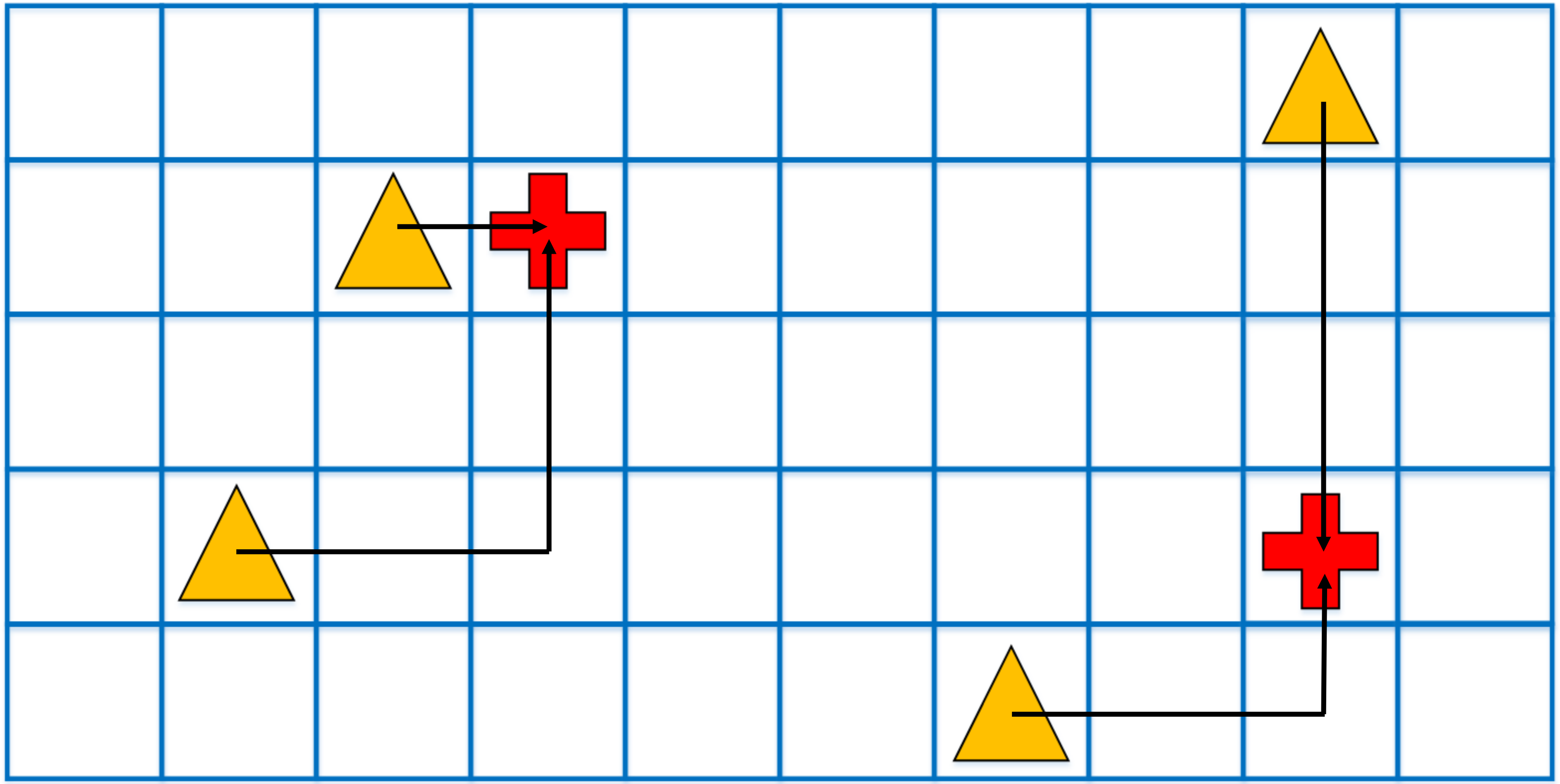
爬山算法 (hill climbing)

- **医院建造问题**
  - ✓ 曼哈顿距离之和减小为了$2+5+3+3=13$



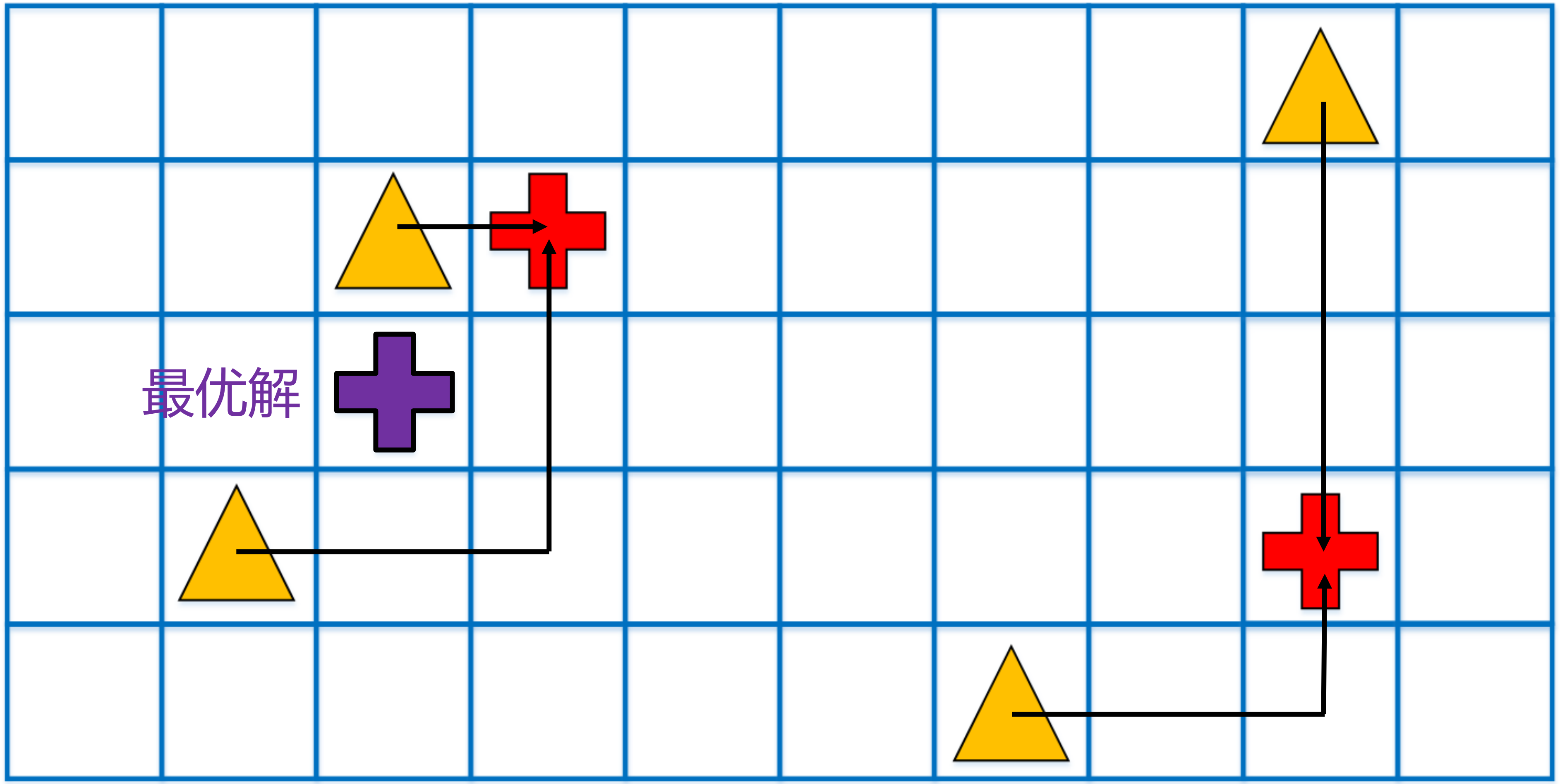
爬山算法 (hill climbing)

- **医院建造问题**
  - ✓ 继续选择邻居，曼哈顿距离之和减小为11



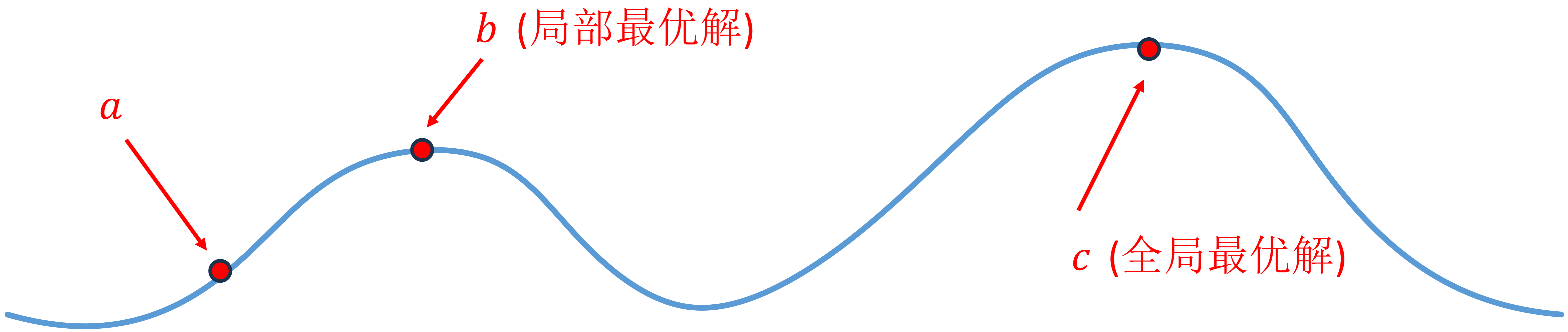
爬山算法 (hill climbing)

- **医院建造问题**
  - ✓ 医院的位置无法移动了，但当前解并不是最优解



爬山算法 (hill climbing)

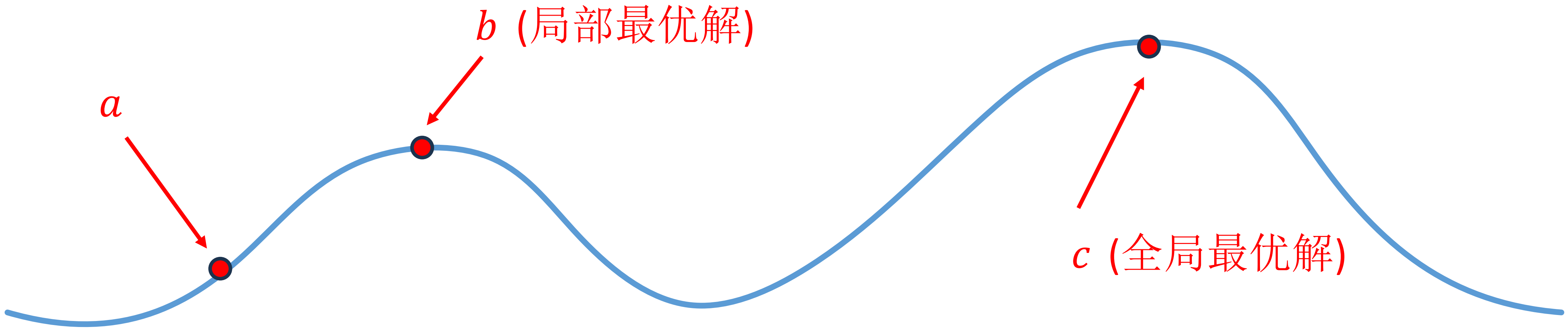
- **为什么叫爬山算法?**
  - ✓ 每次都从当前解的临近解空间中，选择一个最优解作为当前解，直到达到一个局部最优解
  - ✓ 假设有函数 $f(x)$，从$x = a$出发，寻找函数的最大值
  - ✓ 每次都把 $f(x)$ 和 $f(x-1)$，$f(x+1)$比较大小，并更新 x 的值



爬山算法 (hill climbing)

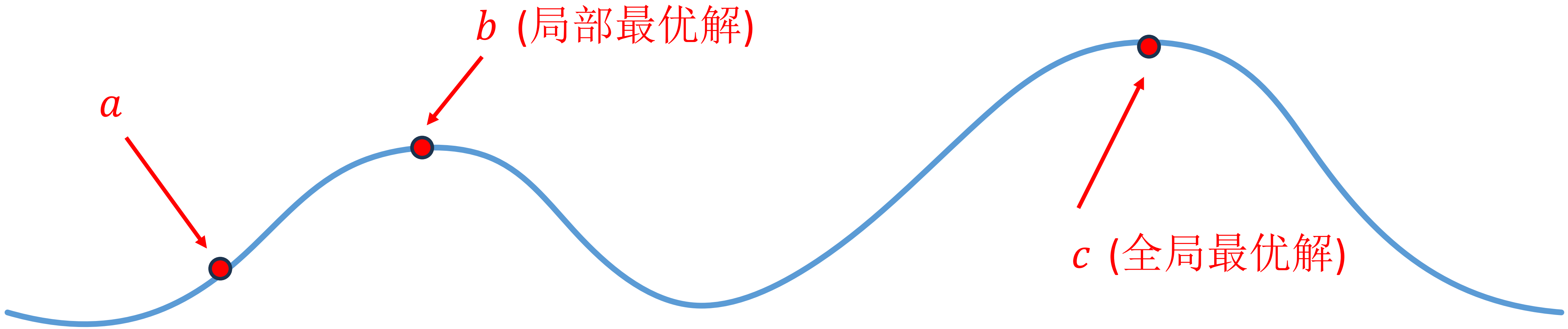
- **爬山算法的变体**

变体	描述
Steepest-ascent	选择邻居中的最优者
Stochastic	从更优的邻居中随机选择一个
First-choice	选择第一个找到的更优邻居
Random-restart	多次执行爬山算法，每次都选取不同的初始状态



模拟退火 (Simulated Annealing)

- 开始时：大概率允许邻居比当前状态更差
- 一段时间后：小概率允许邻居比当前状态更差



模拟退火 (Simulated Annealing)

伪代码：

def simulated_annealing(问题, *max*):

  current = 问题的初始状态

  for t = 1 to *max*:

    T = TEMPERATURE(*t*) # 当前时间的温度，影响改变当前状态的概率

    neighbor = 随机选择 *current* 的一个邻居

    $\Delta E = neighbor$ 比 *current* 好多少

    if $\Delta E > 0$:

      $current = neighbor$

    else: # 邻居比当前状态差的情况下，概率性地接收改变当前状态

      以$e^{\Delta E/T}$ 的概率设置 $current = neighbor$

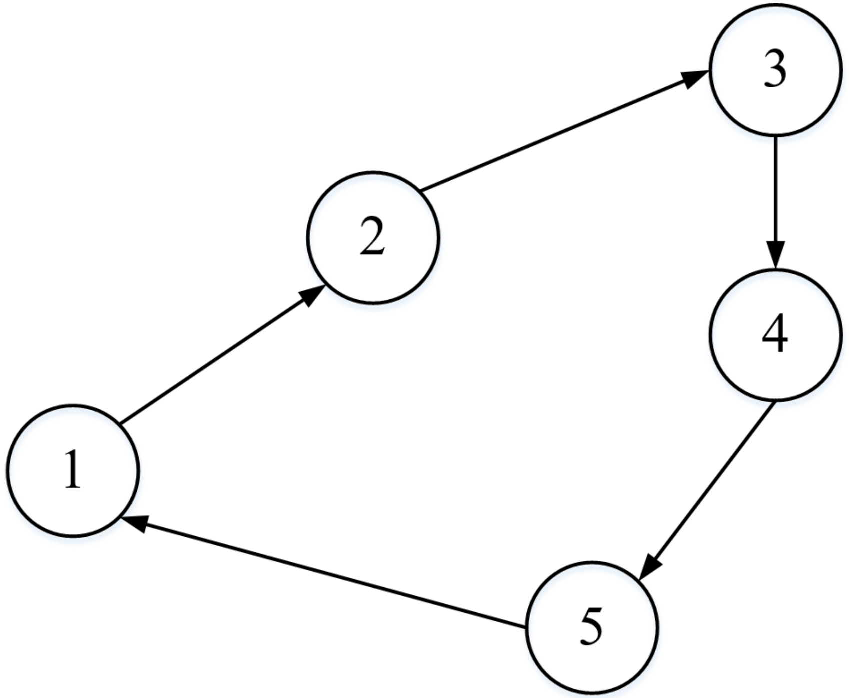
模拟退火 (Simulated Annealing)

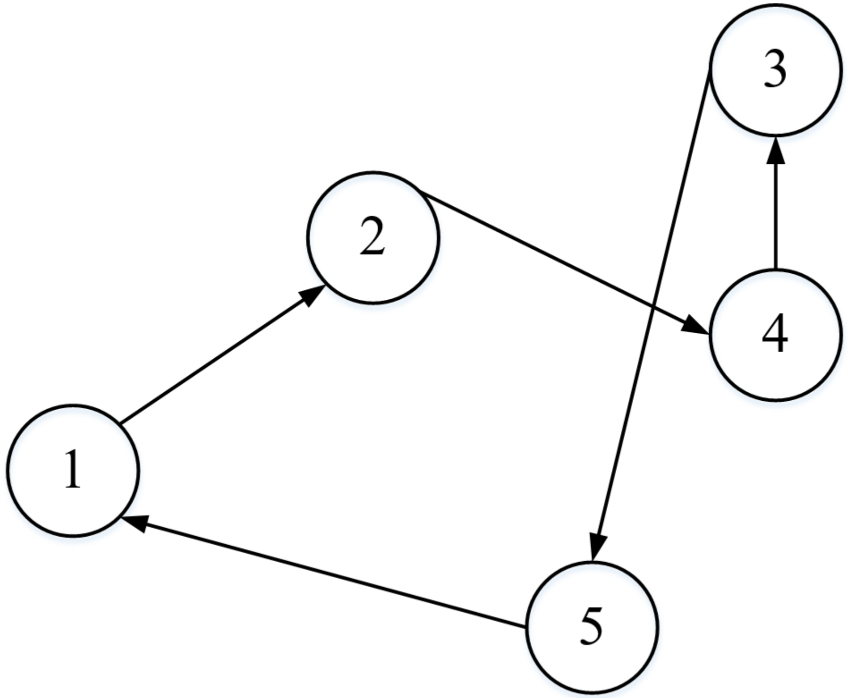
- **为什么叫模拟退火?**
  - ✓ “高温”阶段，愿意尝试非优解
  - ✓ “冷却”阶段，越来越抵触非优解

- **观看视频：** https://www.bilibili.com/video/BV1j64y1Y7FB/

遗传算法 (Genetic Algorithm, GA)

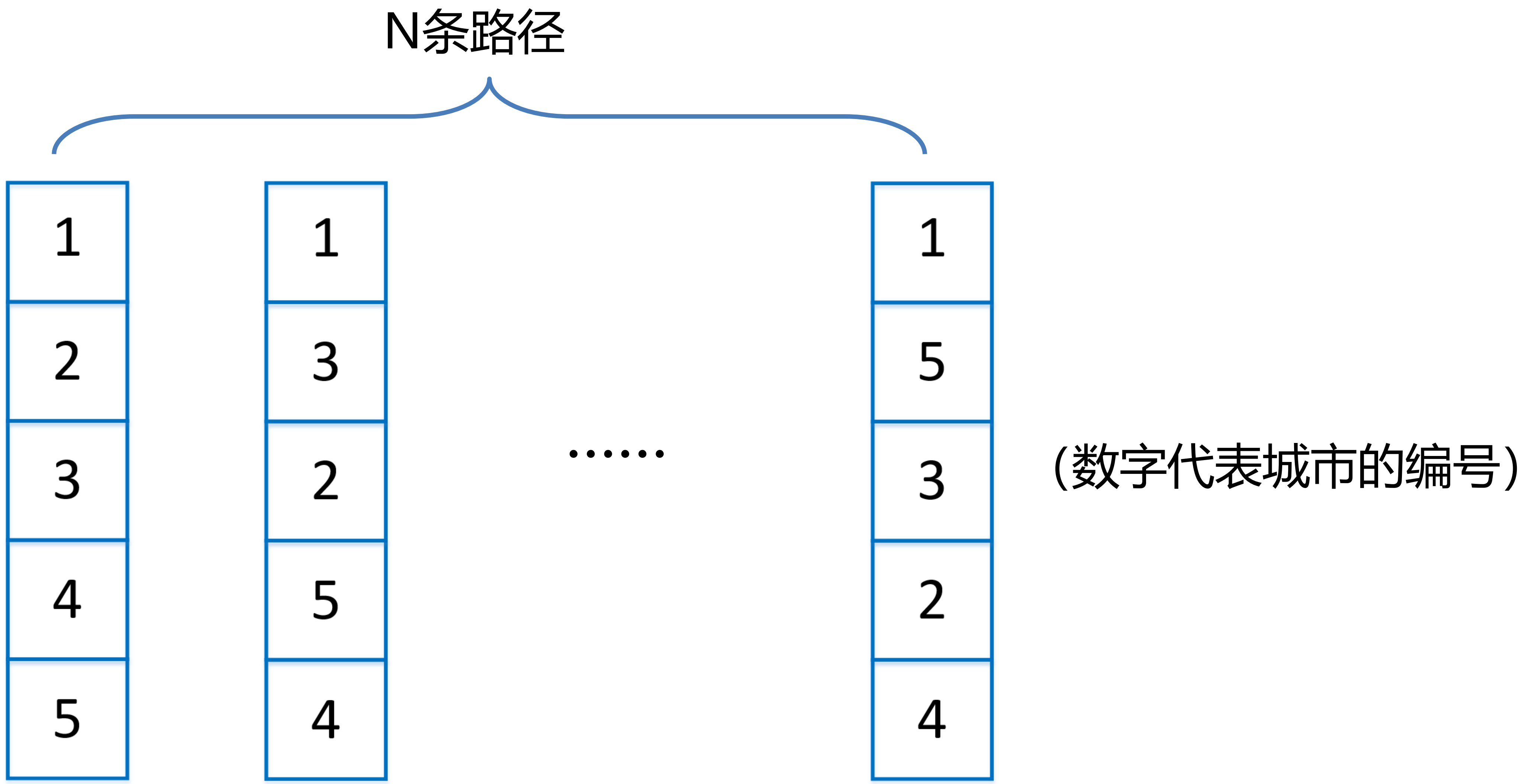
- **旅行商问题（Traveling Salesman Problem, TSP）**
  - 假设有一个旅行商人要拜访N个城市；
  - 需要规划一条路径，满足：每个城市只能拜访一次，且最后回到出发的城市；
  - 目标：求得路径的长度在所有路径中最小。





遗传算法 (Genetic Algorithm, GA)

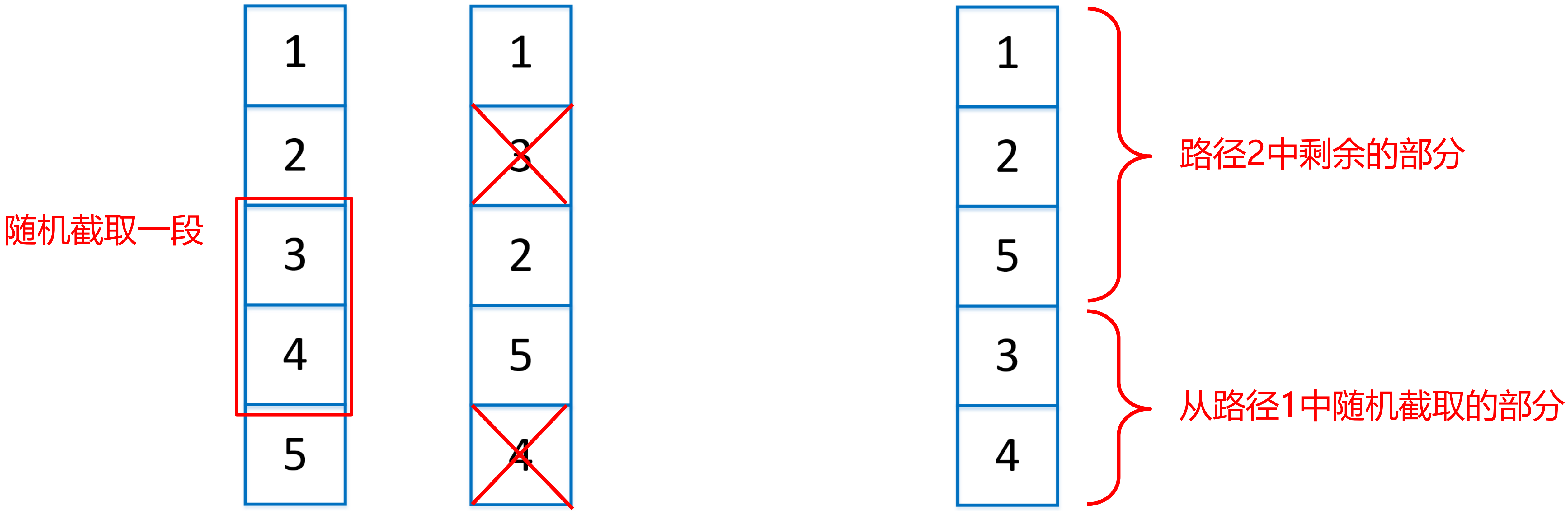
- **第一步：**假设有5个城市，随机生成N条路径（群体，population）



遗传算法 (Genetic Algorithm, GA)

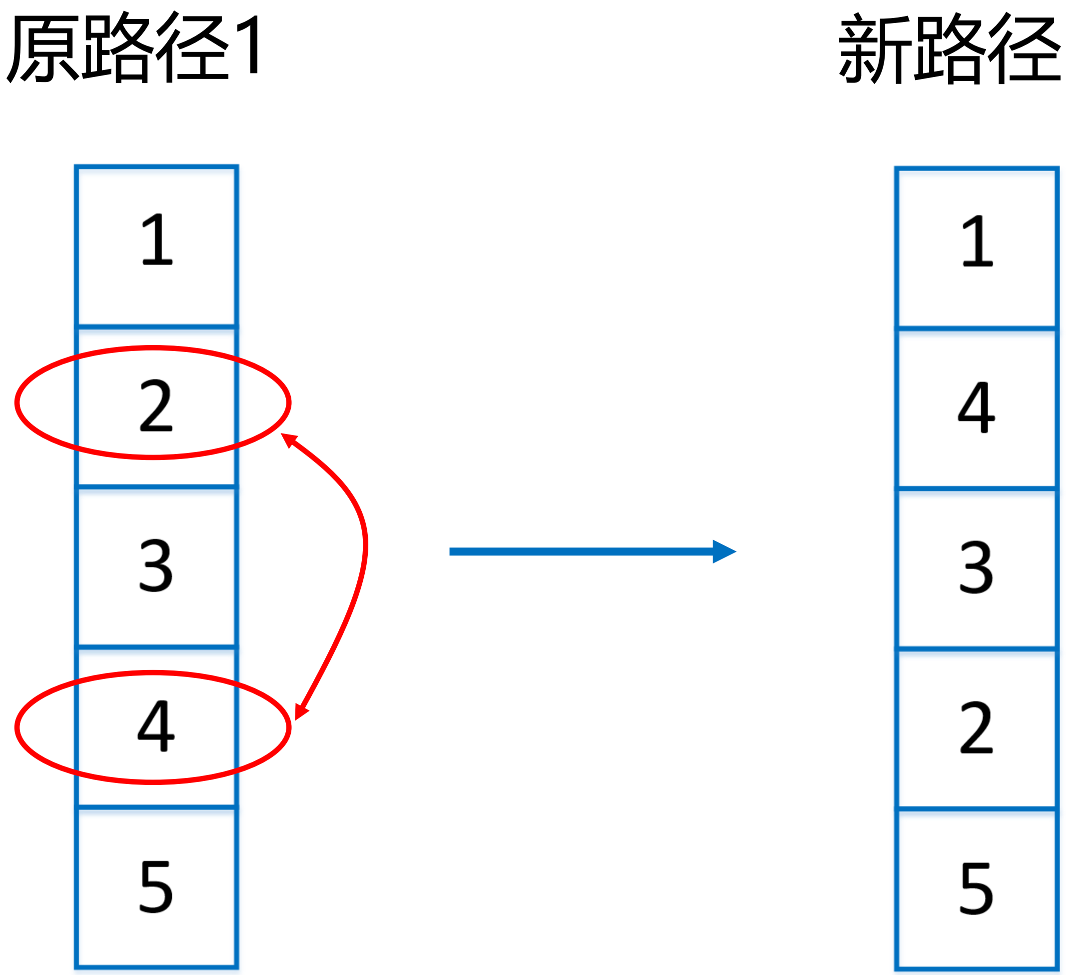
- **第二步：**对每条路径概率性地使用交叉（crossover）生成新的路径

原路径1　随机选取的路径　　　重组得到一条新路径



遗传算法 (Genetic Algorithm, GA)

- **第三步：**对每条路径概率性地使用变异（mutation）生成新的路径



遗传算法 (Genetic Algorithm, GA)

- **第四步：** 计算N条新路径的长度，并使用赌轮选择（roulette wheel selection）选出N条路径

- **赌轮选择：** 有放回抽取，长度越短的路径越容易被选中

  ✓ 假设共有3条新路径：路径1长度为15km，路径2长度为30km，路径3长度为60km

  ✓ 适应度（fitness）：路径1为 $\frac{1}{15}$，路径2长度为 $\frac{1}{30}$，路径3长度为 $\frac{1}{60}$

  ✓ 选择N次，每次路径1被选中的概率均为：$\frac{\frac{1}{15}}{\frac{1}{15}+\frac{1}{30}+\frac{1}{60}} = \frac{4}{7}$

  ✓ 同理，每次路径2被选中的概率为 $\frac{2}{7}$，路径3被选中的概率为 $\frac{1}{7}$

遗传算法 (Genetic Algorithm, GA)

湖北大学 HUBEI UNIVERSITY

- **第五步：重复EPOCH次上述操作，然后返回当前最优解**

遗传算法 (Genetic Algorithm, GA)

- **为什么叫遗传算法?**
- **观看视频：** https://www.bilibili.com/video/BV1s44y1z7xt/

粒子群优化算法 (Particle Swarm Optimization, PSO)

- **鸟群觅食问题：**假设鸟可以知道离食物多远，但不知道食物在哪个方向
- 为了找到食物，下一时刻应该朝哪个方向，以多大的速率飞行?

原方向

鸟群达到过的离食物最近的位置

自身达到过的离食物最近的位置

粒子群优化算法 (Particle Swarm Optimization, PSO)

- 假设共有有m只鸟，第 i 只鸟的当前位置为 $L_i = (x_i, y_i)$
- 第 i 只鸟达到过的离食物最近的位置为 $P_i = (a_i, b_i)$，群体最近(a_{best}, b_{best})
- 第 i 只鸟的速度为 $V_i = (v_i, u_i)$ （x轴和y轴方向上的速率）

原方向

群体达到过的离食物最近

的位置(a_{best}, b_{best})

自身达到过的离食物最近

的位置 $P_i = (a_i, b_i)$

粒子群优化算法 (Particle Swarm Optimization, PSO)

- 第 i 只鸟下一时刻（$(k+1)$时刻）的
  - ✓ 速率 $v_i^{k+1} = w \cdot \boxed{v_i^k} + c_1 \cdot rand() \cdot \boxed{(a_i - x_i^k)} + c_2 \cdot rand() \cdot \boxed{(a_{best} - x_i^k)}$

惯性因子
一般取0.8

个体学习因子
一般取2.0

0~1之间的
随机数

社会学习因子
一般取2.0

  - ✓ 速率 $u_i^{k+1} = w \cdot u_i^k + c_1 \cdot rand() \cdot (b_i - y_i^k) + c_2 \cdot rand() \cdot (b_{best} - y_i^k)$

粒子群优化算法 (Particle Swarm Optimization, PSO)

湖北大学 HUBEI UNIVERSITY

- 第 i 只鸟下一时刻（$(k+1)$时刻）的
  - ✓ 位置 $x_i^{k+1} = x_i^k + \alpha \cdot v_i^{k+1}$

    约束因子
    或收敛因子
  - ✓ 位置 $y_i^{k+1} = y_i^k + \alpha \cdot u_i^{k+1}$

粒子群优化算法 (Particle Swarm Optimization, PSO)

- 迭代EPOCH次，最后返回离食物最近的鸟

蚁群算法（Ant Colony Optimization, ACO）

- **蚁群觅食：**

  - ✓ 每只蚂蚁都会在走过的路径上留下信息素
  - ✓ 每只蚂蚁都会选择信息素浓度较高的路径
  - ✓ 设每只蚂蚁携带的信息素总量为 Q，走过的路径为 L，留在路径上的信息素浓度为 $\frac{Q}{L}$
  - ✓ 信息素会随着时间不断挥发

小概率选择信息素浓度较低的路径

大概率选择信息素浓度较高的路径

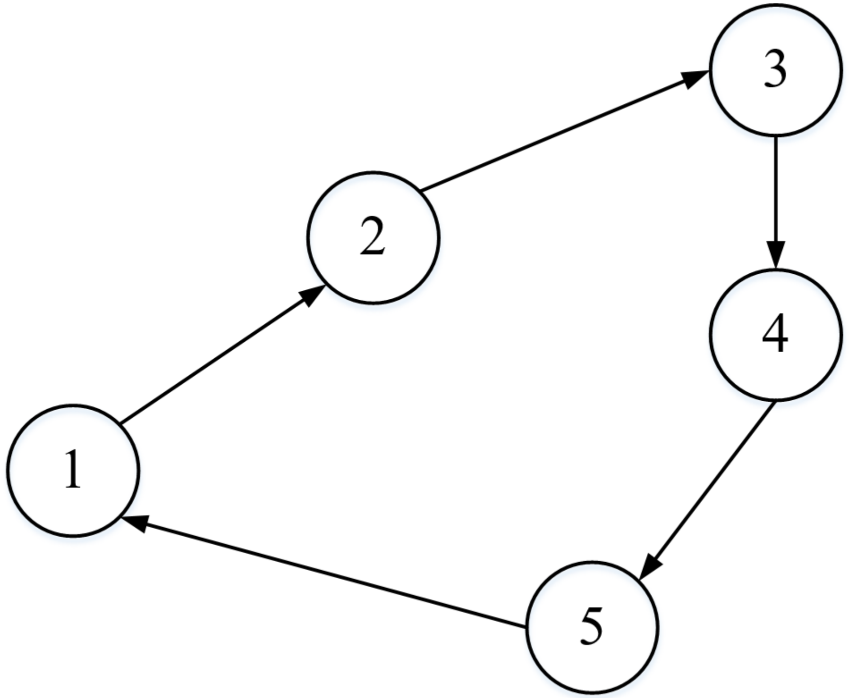
蚁群算法

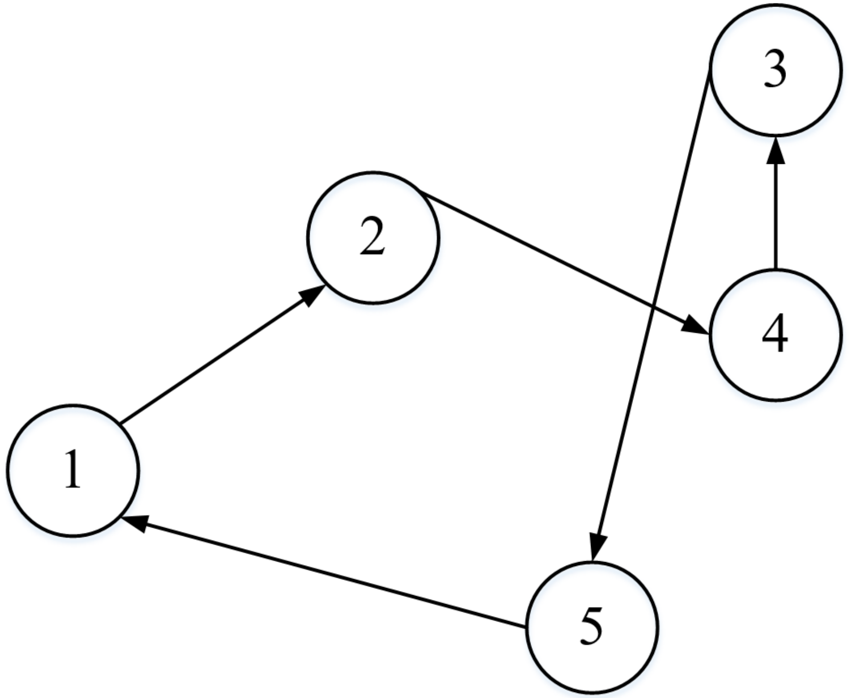
- **旅行商问题（Traveling Salesman Problem, TSP）**

  ✓ 假设有一个旅行商人要拜访n个城市；

  ✓ 需要规划一条路径，满足：每个城市只能拜访一次，且最后回到出发的城市；

  ✓ 目标：求得路径的长度在所有路径中最小。





蚁群算法

- **步骤一：参数初始化**
  - ✓ 设蚂蚁数量 m，城市数量 n，城市 i 和 j 之间的距离为 d_{ij}
  - ✓ 在 t 时刻（第 t 轮迭代中），城市 i 和城市 j 之间的路径上信息素浓度为 $\tau_{ij}(t)$
  - ✓ 初始时，各条路径的信息素浓度相同，设 $\tau_{ij}(0)$

- **步骤二：计算选择路径的概率**

✓ 设第 k 只蚂蚁当前在城市 i

尚未访问过的城市的集合

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{\tau_{ij}(t)}{\sum_{s\in allowed_k}\tau_{is}(t)} & if \quad j\in allowed_k\\ 0 & if \quad j\notin allowed_k\end{cases}$$

t 时刻第 k 只蚂蚁从城市 i 选择去城市 j 的概率

湖北大学 HUBEI UNIVERSITY

- **步骤二：改进概率的计算方法，加快收敛速度**

城市 i 到城市 j 的距离的倒数

选择较短路径的概率更大

$$P_{ij}^{k}(t)\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\frac{1}{d_{ij}(t)}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\times[\frac{1}{d_{is}(t)}]^{\beta}} & if \quad j\in allowed_k\\ 0 & if \quad j\notin allowed_k\end{cases}$$

湖北大学
HUBEI UNIVERSITY

- **步骤三：**蚂蚁走完所有城市后，更新信息素浓度

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

信息素挥发

新增的信息素

其中，$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & if \text{ 第 } k \text{ 只蚂蚁曾经走过路径 } i \text{ 到 } j \\ 0 & if \text{ 其他} \end{cases}$

新增信息素的浓度，
其中L_k为第k只蚂蚁走过的总路径长度

- **步骤四：**记录当前的最优路径，并继续迭代。如果迭代次数达到上限则返回当前的最优路径

湖北大学 HUBEI UNIVERSITY

- **实例：**2只蚂蚁，4个城市，每只蚂蚁携带的信息素总量$Q=10$，挥发率$\rho=0.5$，城市之间的初始信息素$\tau_{ij}(0)=0.4$

- 路径长度矩阵 $A=\begin{bmatrix}0&2&6&3\\2&0&4&1\\6&4&0&8\\3&1&8&0\end{bmatrix}$，假设第一次迭代后：

  ✓ 蚂蚁1选择的路径：3-2-1-4-3，总长度为17，信息素浓度 $\frac{10}{17}=0.59$

  ✓ 蚂蚁2选择的路径：2-4-1-3-2，总长度为14，信息素浓度 $\frac{10}{14}=0.71$

更新信息素浓度：

- 蚂蚁1选择的路径：3-2-1-4-3，总长度为17，信息素浓度 $\frac{10}{17} = 0.59$
- 蚂蚁2选择的路径：2-4-1-3-2，总长度为14，信息素浓度 $\frac{10}{14} = 0.71$
- $\tau_{24}(1) = (1-\rho)\tau_{ij}(0) + \sum_{k=1}^{2} \Delta\tau_{ij}^{k} = 0.5 \times 0.4 + 0 + 0.71 = 0.91$

$t = 1$时刻，城市2→城市4
之间的信息素浓度

蚂蚁1没有在城市2→城市4
之间留下信息素

蚂蚁2留下了
浓度为0.71的信息素

理论课 第5次平时作业（4月16日上课前交给班长）

湖北大学 HUBEI UNIVERSITY

1. 简述爬山算法和模拟退火算法的异同点
2. 简述模拟退火算法是如何避免算法陷入局部最优解的
3. 在ppt的例子中，计算 $t=1$ 时刻，城市3 → 城市2之间的信息素浓度$\tau_{32}(1)$

结束语

谢 谢!