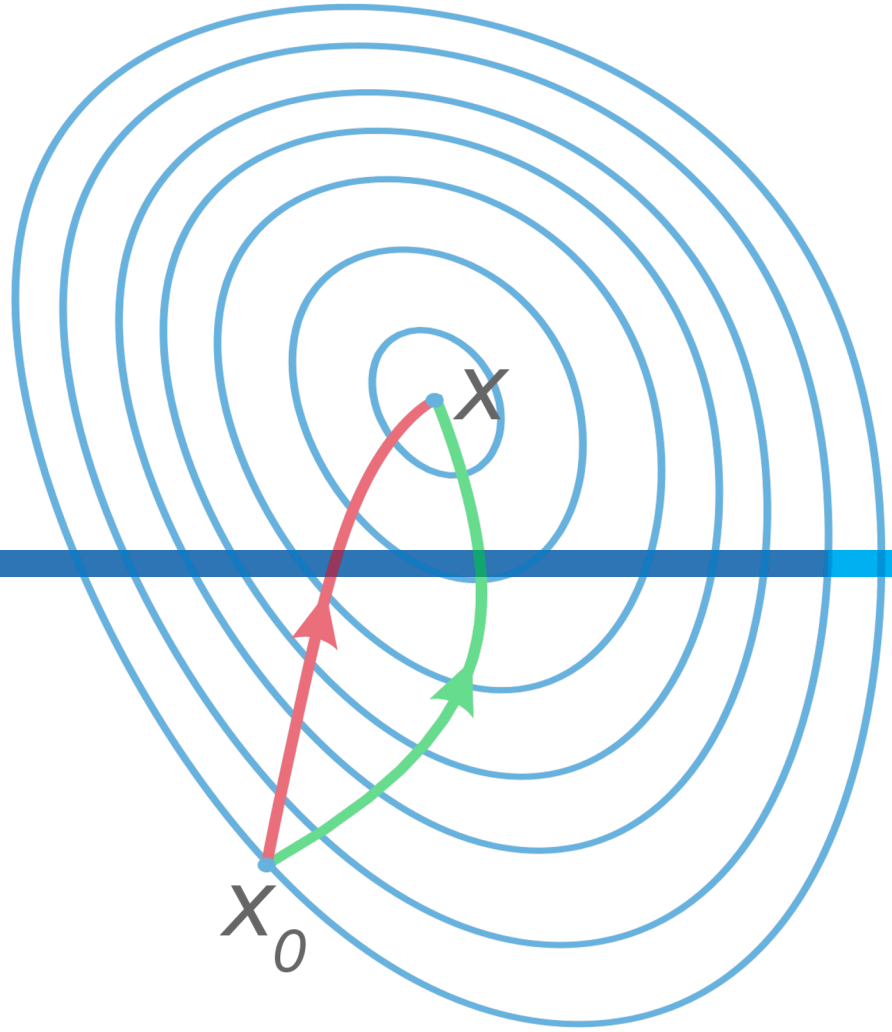
最优化方法



第五周

计算机学院
余皓然

2024/3/25

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）

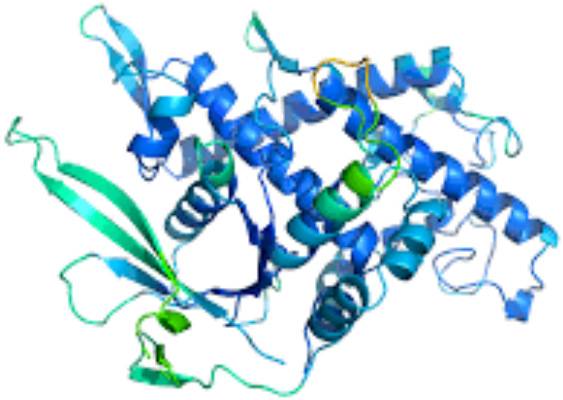
- 求解最优化问题的两种模式

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 99 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

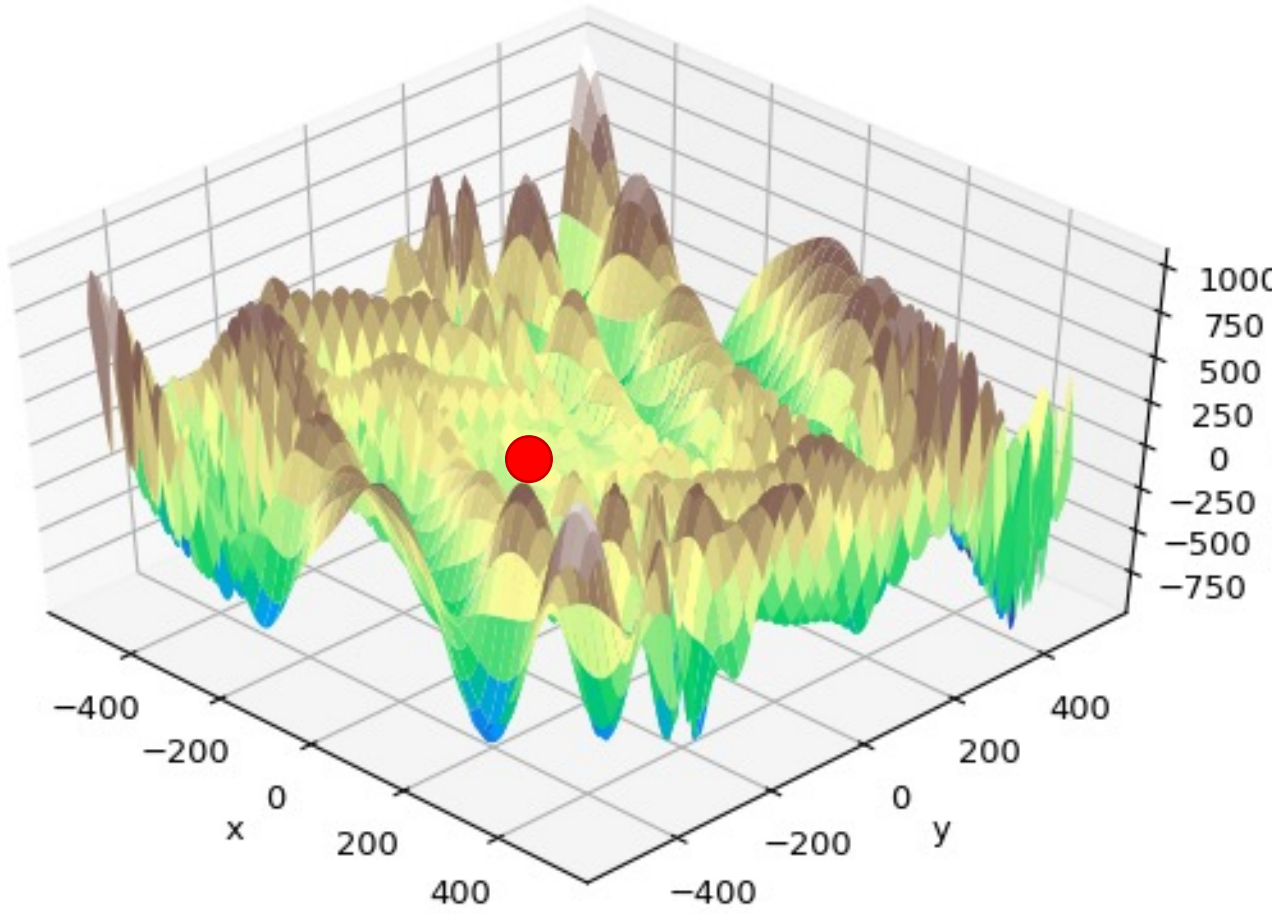
Part1 如何从复杂度的角度衡量算法好坏？
Part2 针对一些有特定形式的最优化问题，如何找到最优解？
Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？
局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法
Part4 针对较复杂的多目标最优化问题，如何找到较好解？

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

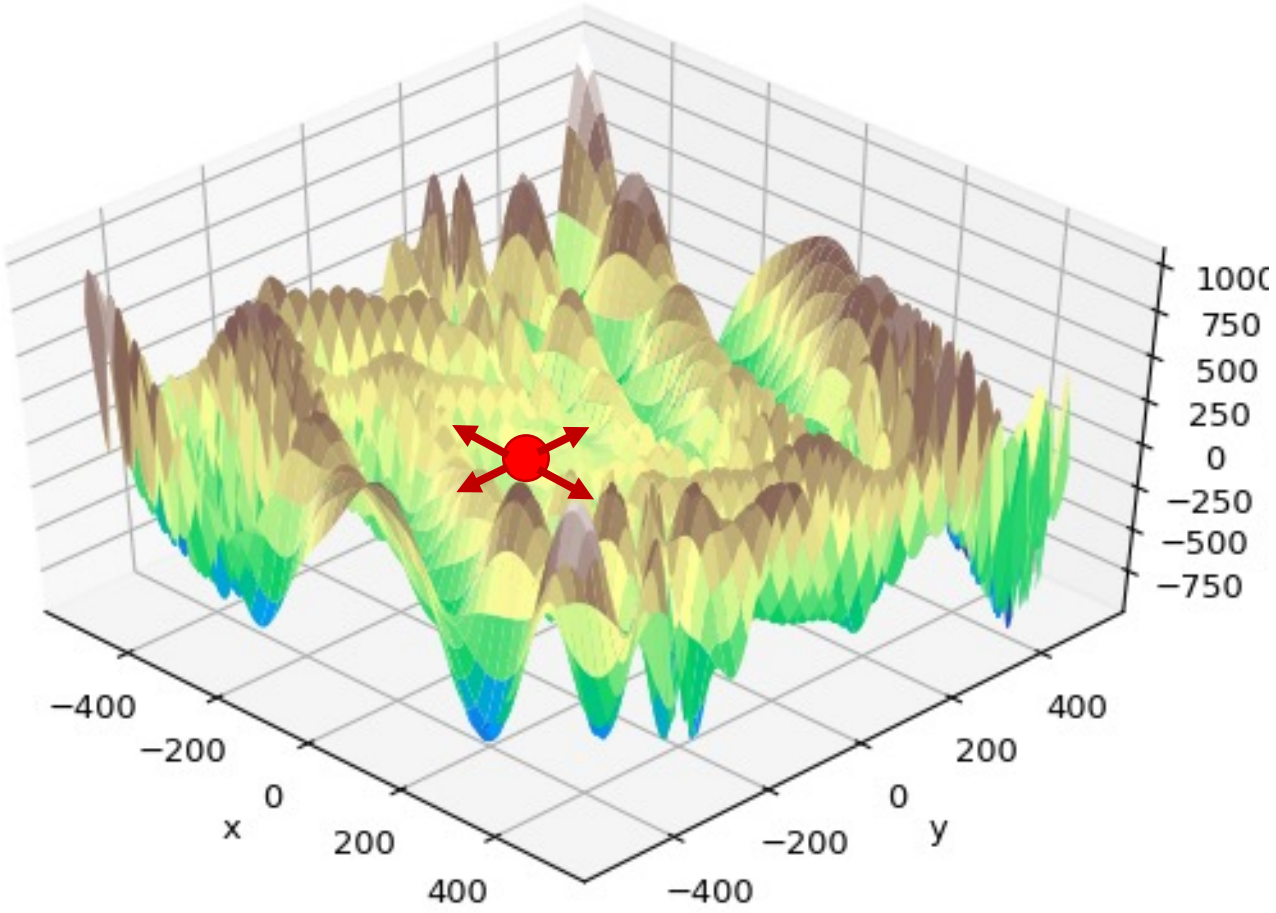
(x,y)


1000
750
500
250
0
−250
−500
−750
−400
−200
0
200
400
x
400
200
0
−200
−400
y

登山搜索

h(x,y)

h(x+1,y)
h(x-1,y)
h(x,y+1)
h(x,y-1)

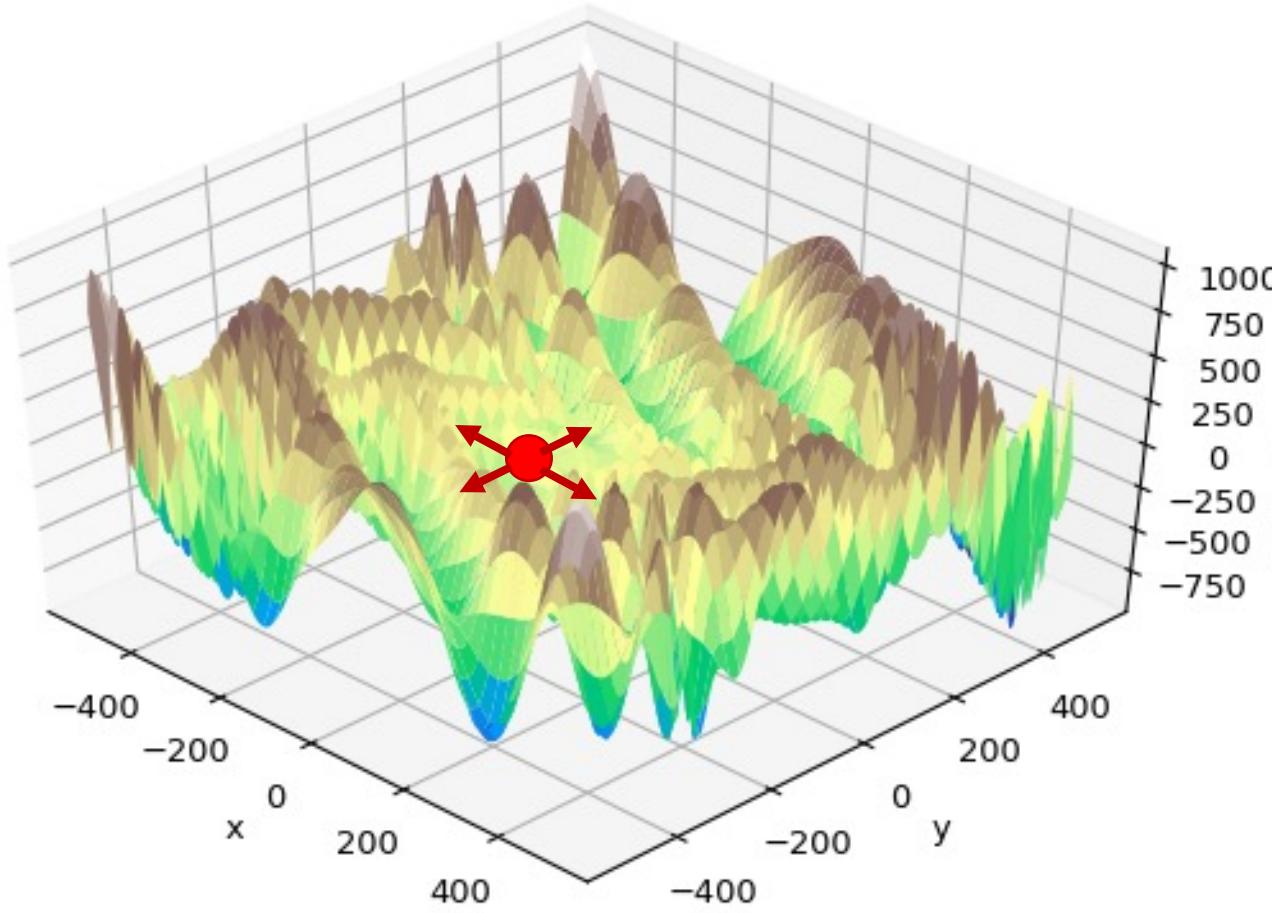


登山搜索

"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$\boldsymbol{x}_{\text{current}}$

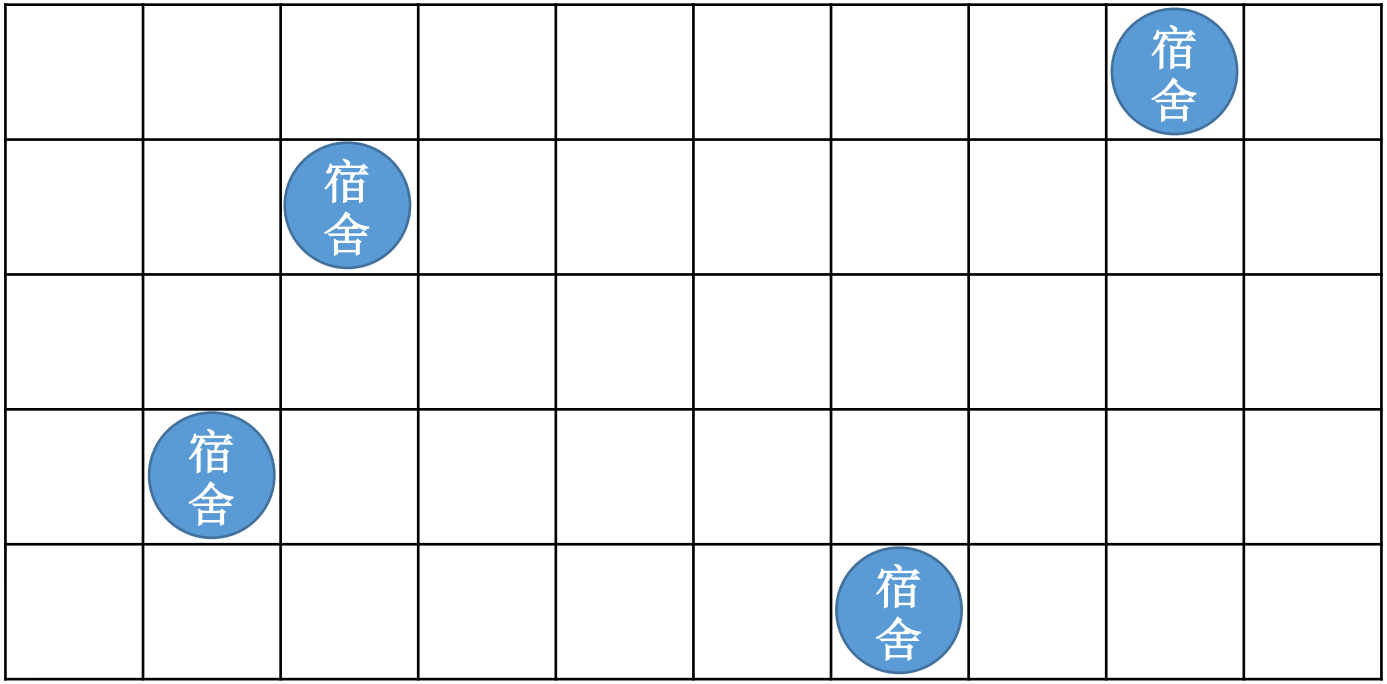
1 寻找当前解的所有邻域解 自行定义邻域

2 得到对应目标函数值$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，返回1

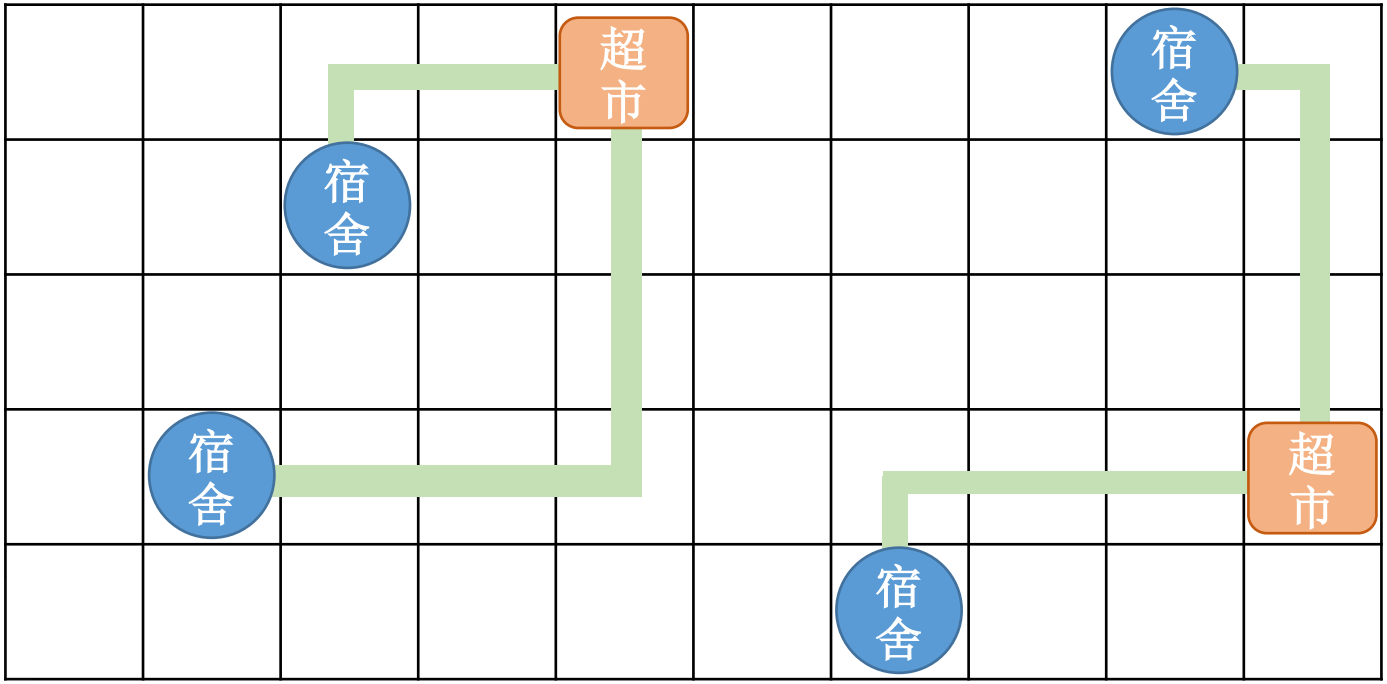
例子回顾

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?



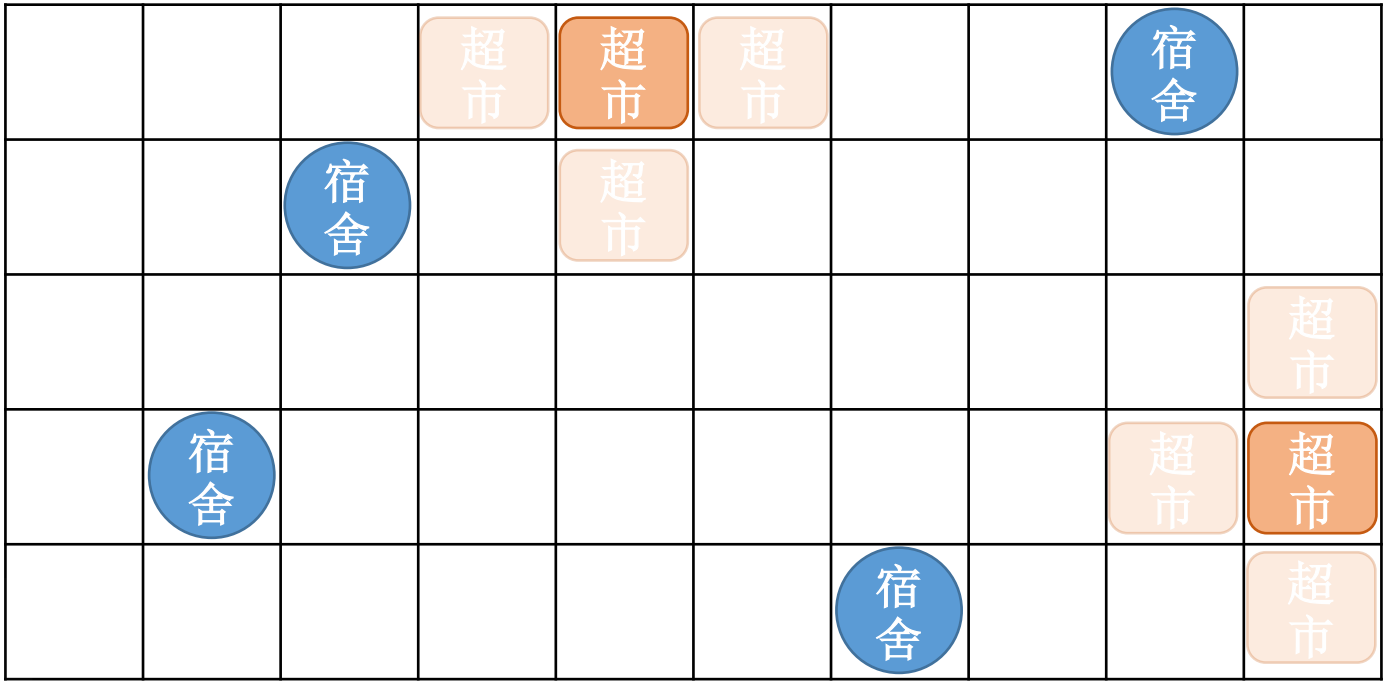
例子回顾

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

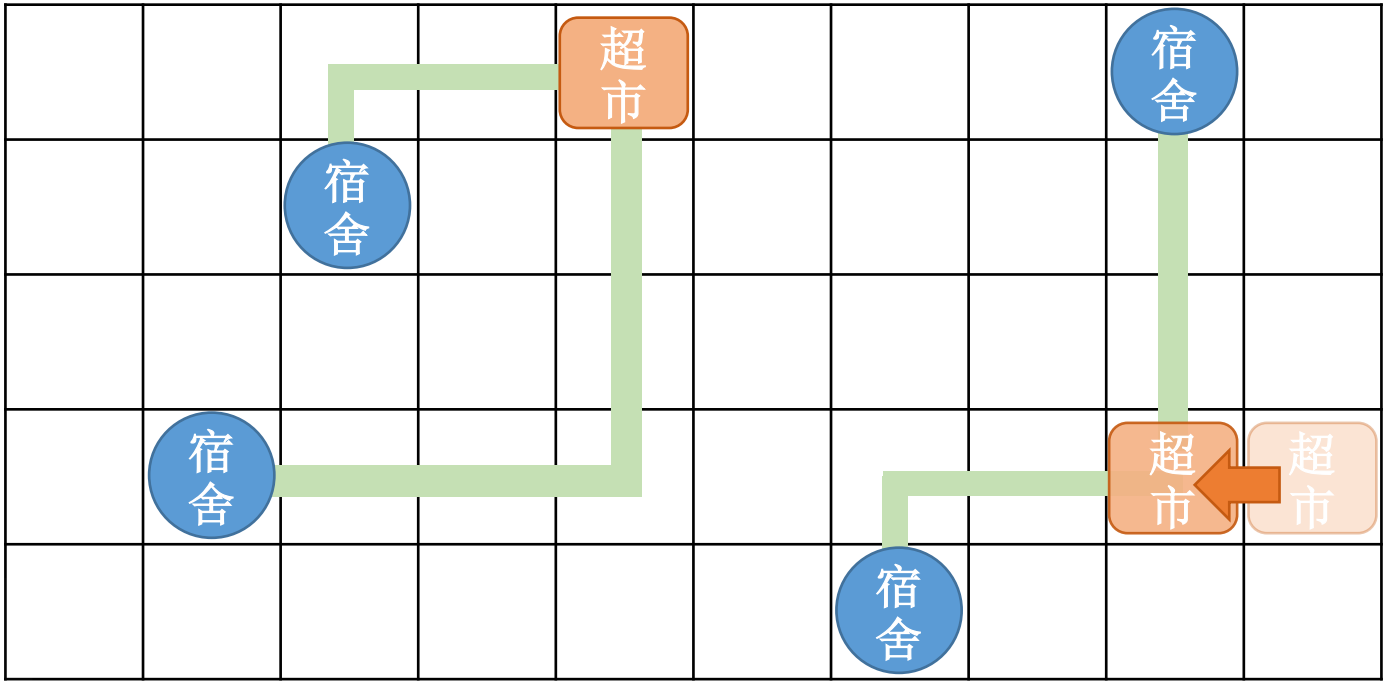
例子回顾



以上选址方案对应的目标函数值为17

例子回顾

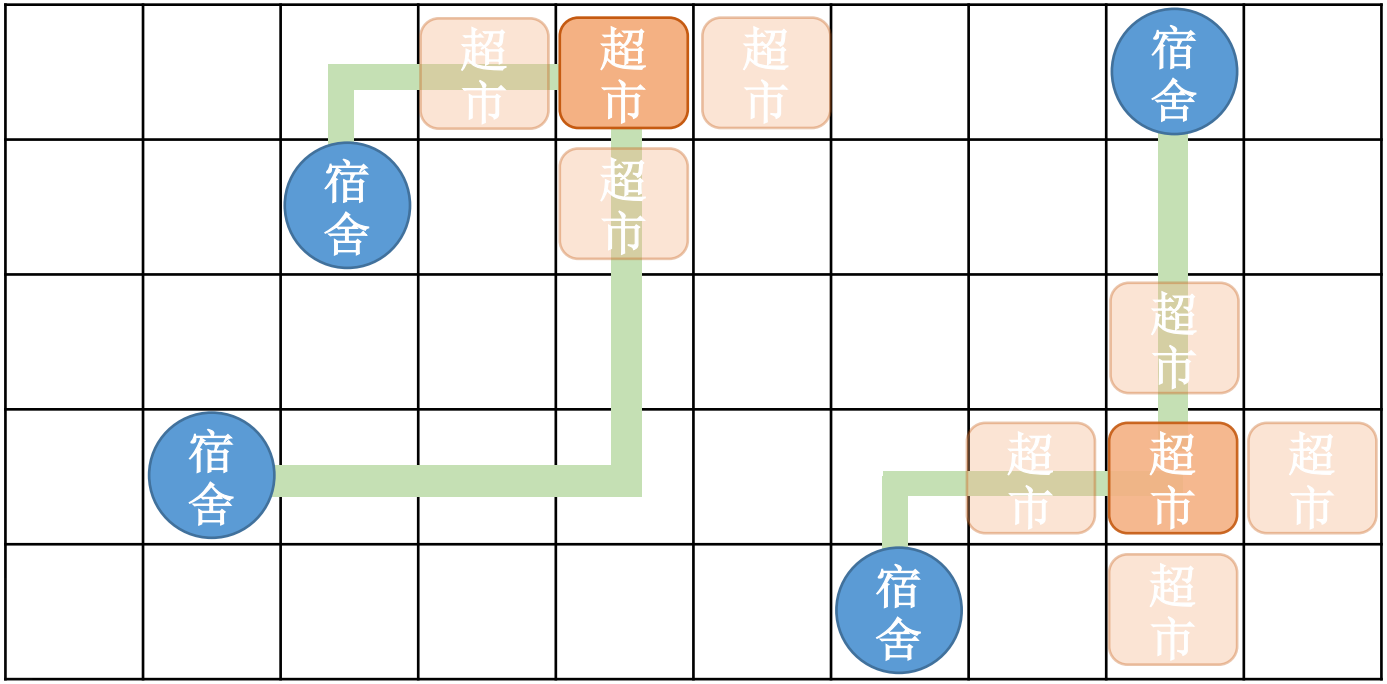
从6种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从17降低为15

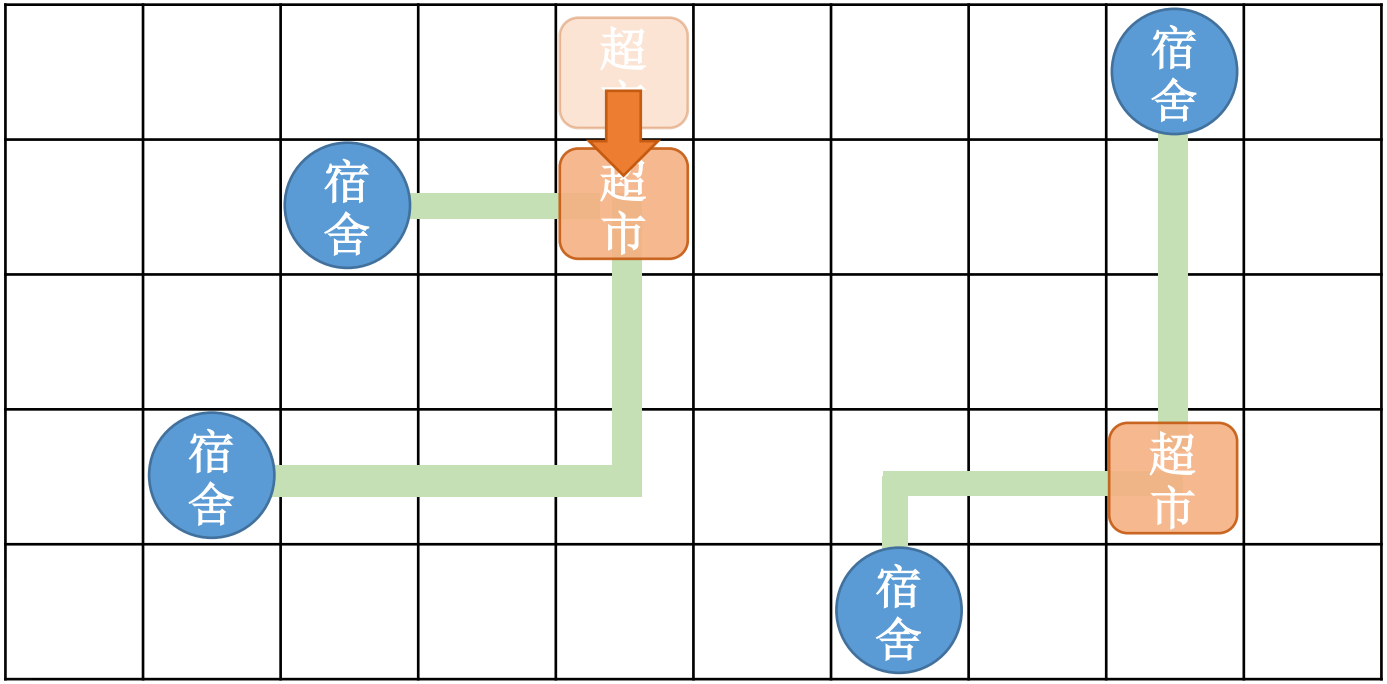
例子回顾

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

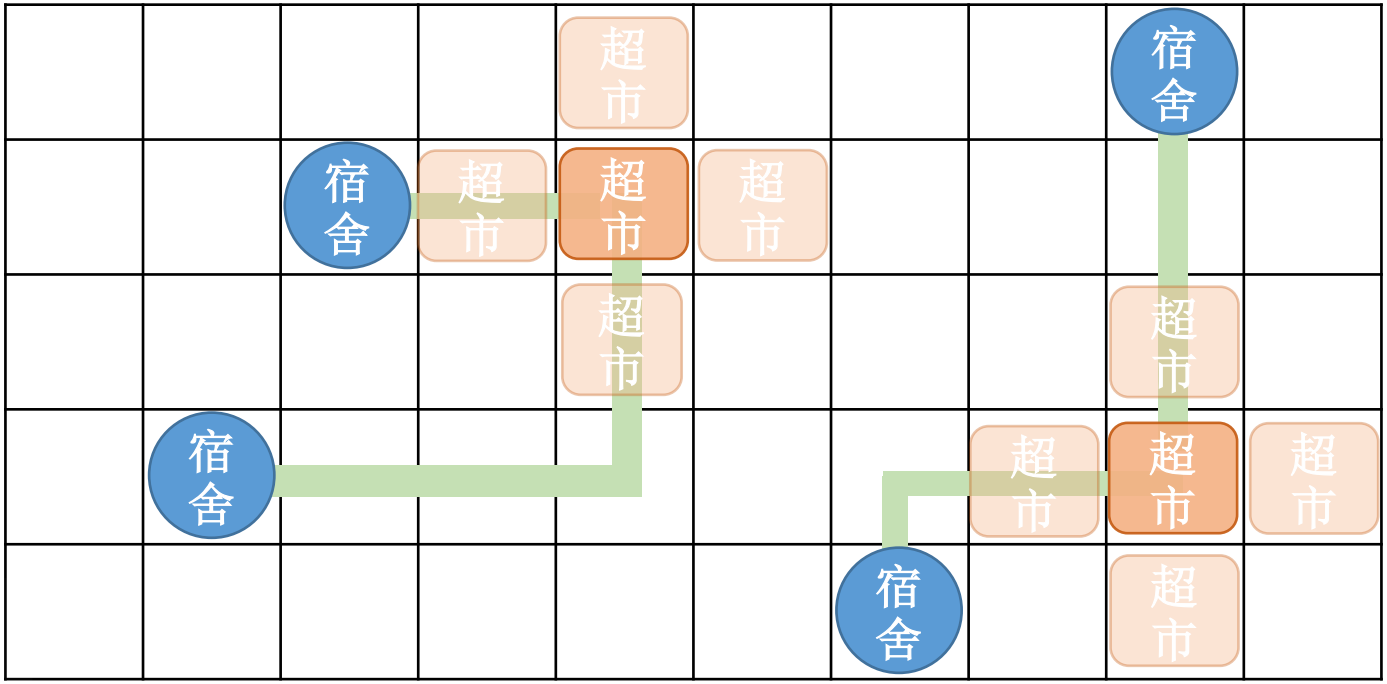
从7种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从15降低为13

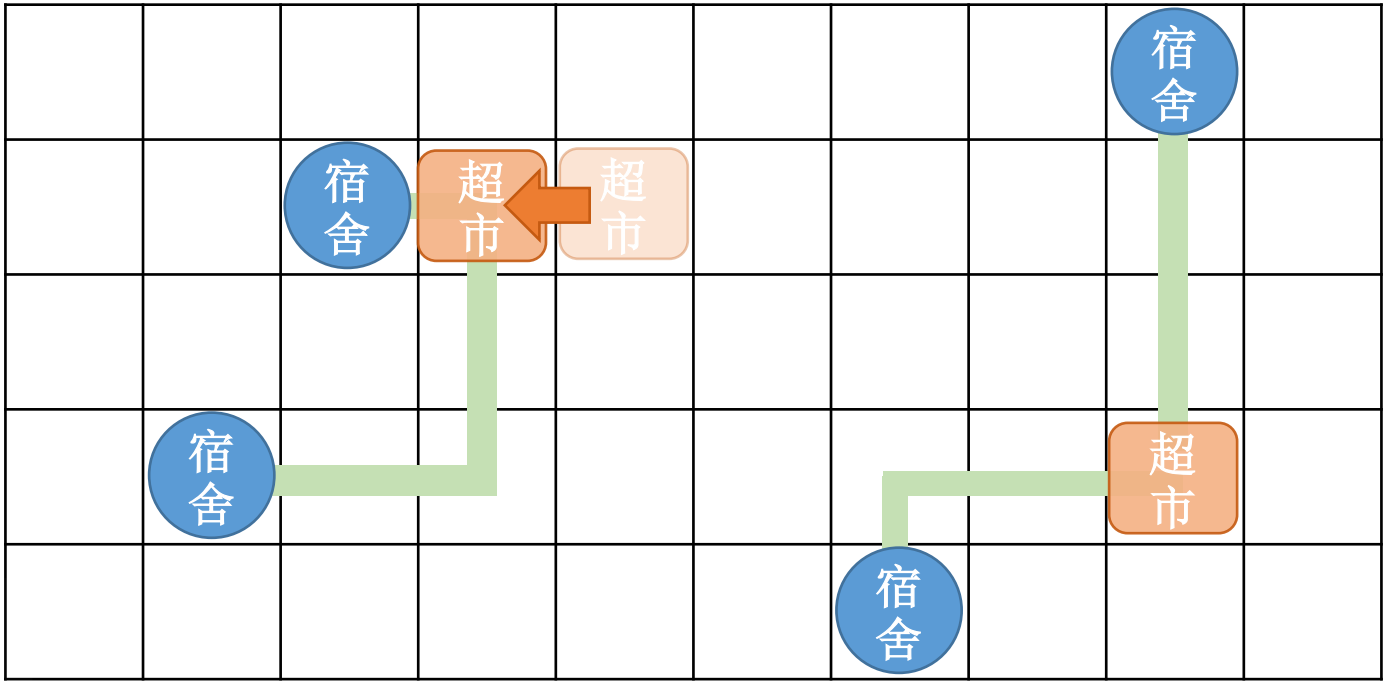
例子回顾

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

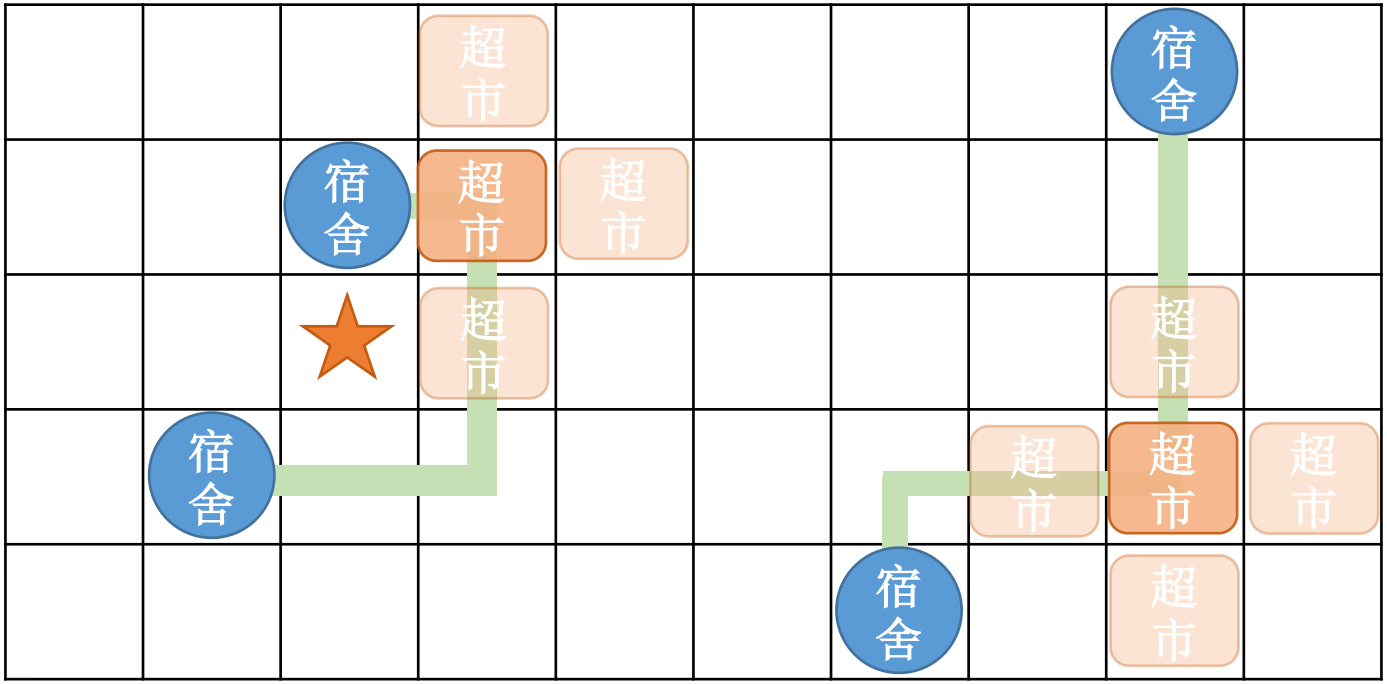
从8种该类更改方案中选择对应目标函数最低的方案



选址方案对应的目标函数值从13降低为11

例子回顾

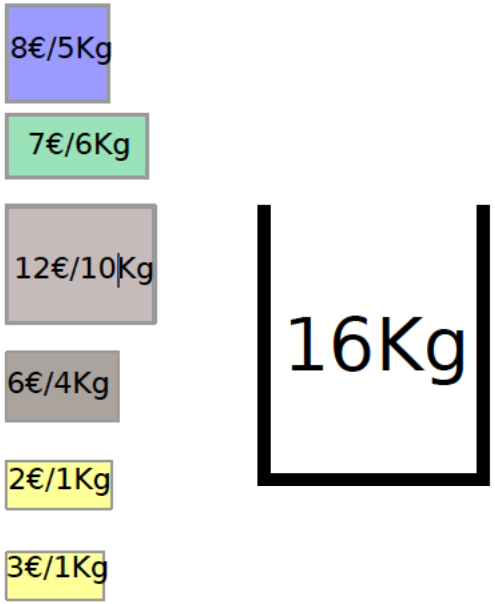
针对新方案有7种更改方案，但都不能进一步降低目标函数



该算法停止搜索，输出结果，对应目标函数值为11

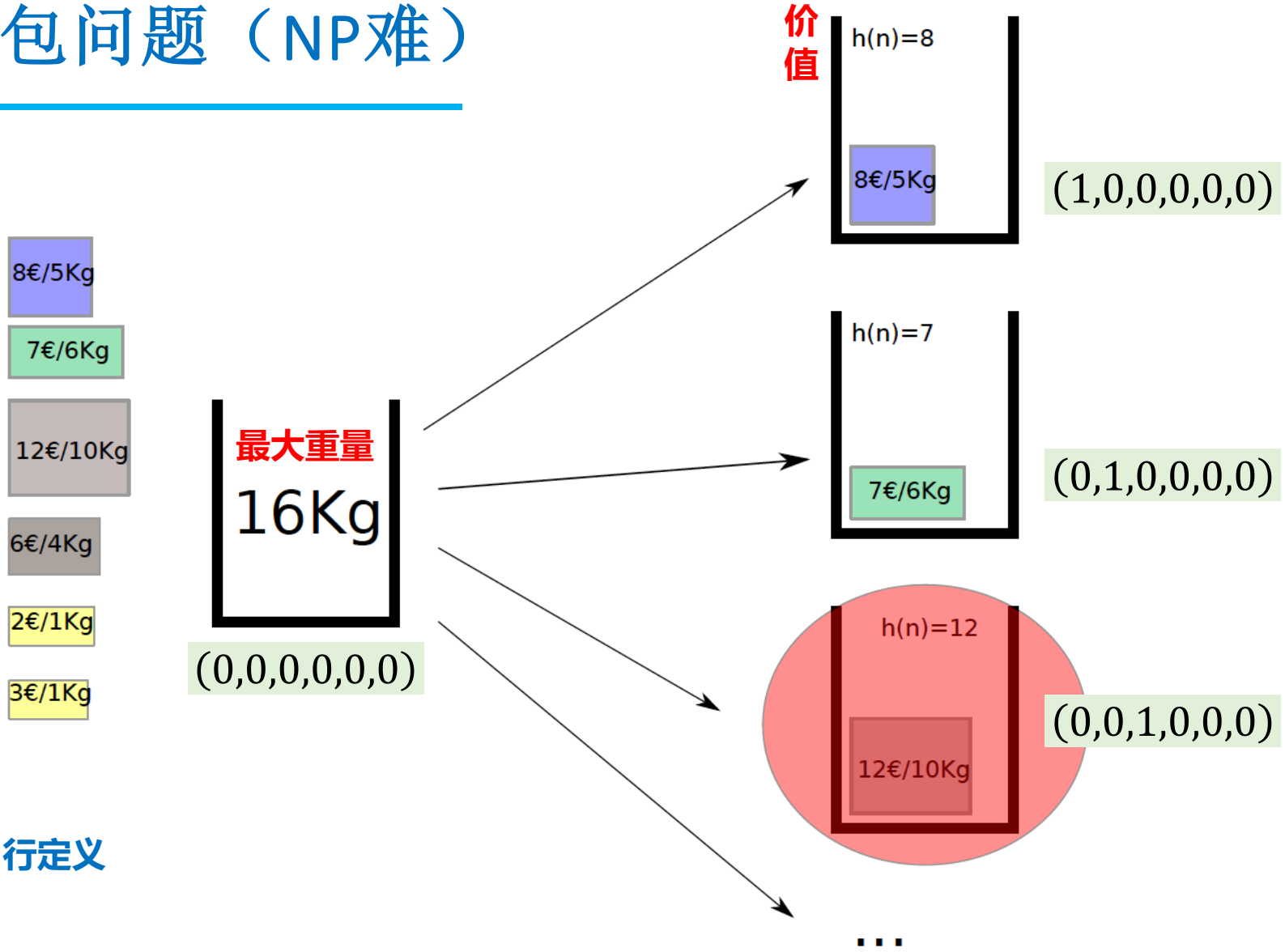
0-1背包问题（NP难）

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



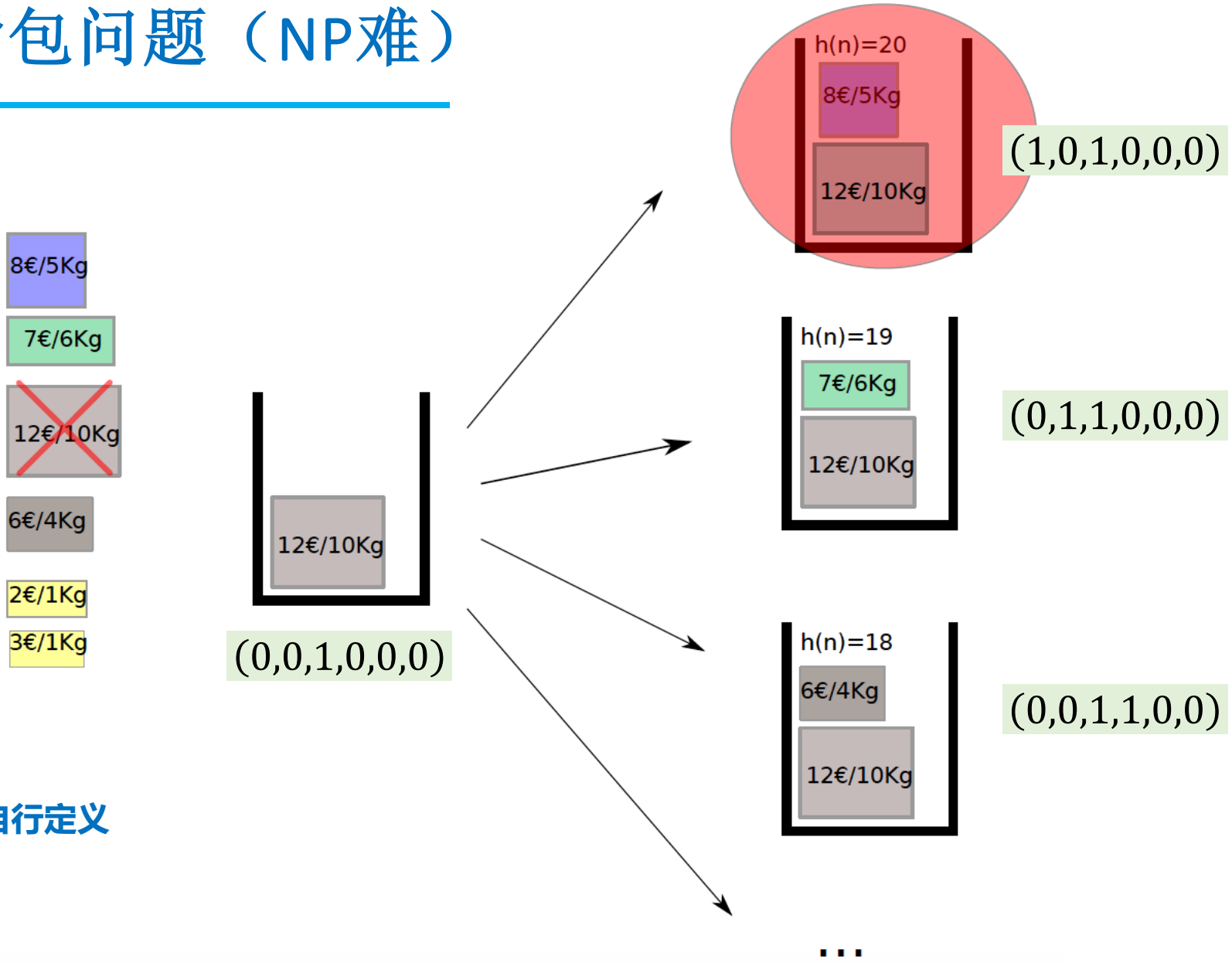
例题取自LSI-FIB-UPC <Artificial Intelligence>

0-1背包问题（NP难）



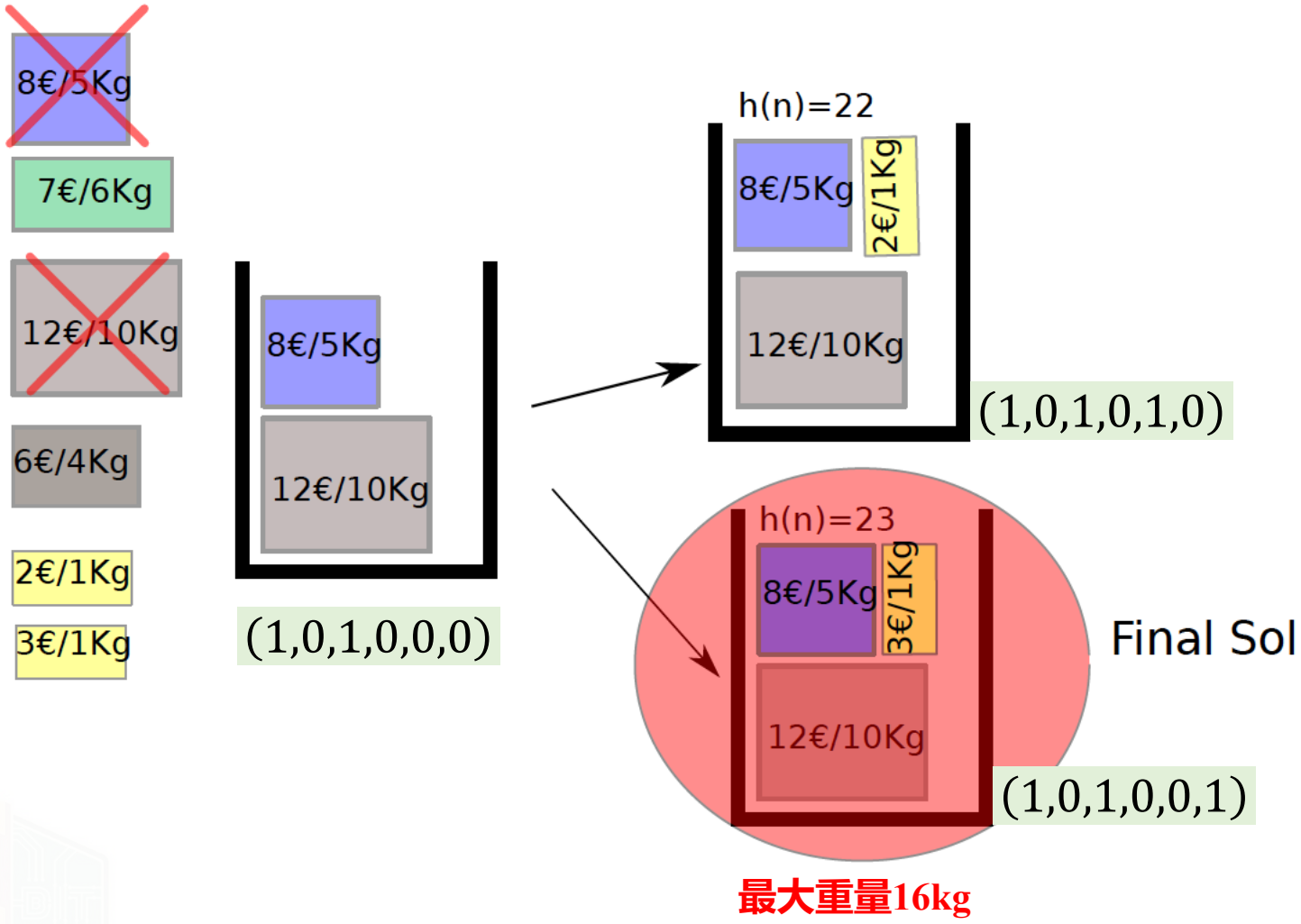
邻域自行定义

0-1背包问题（NP难）

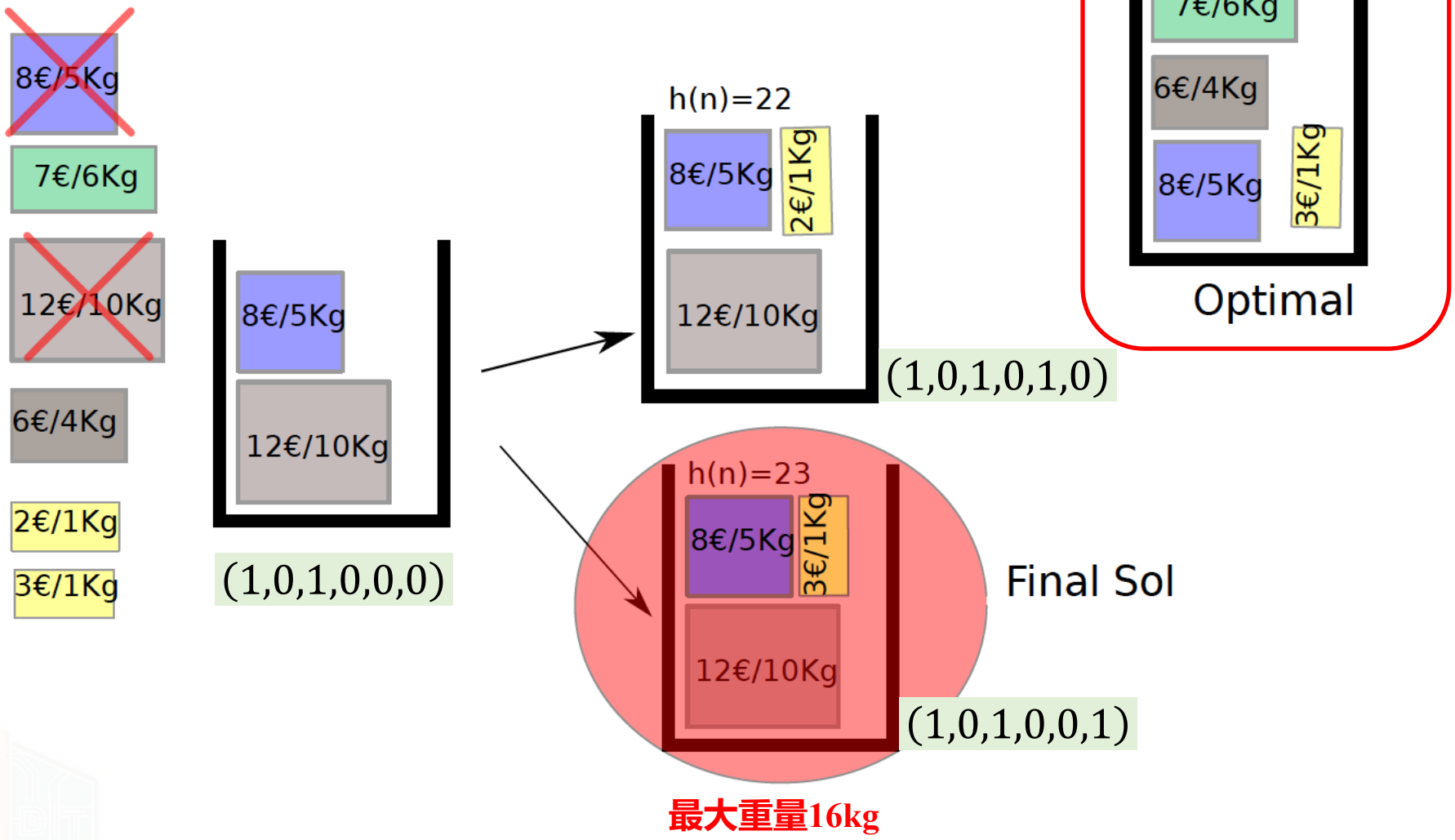


邻域自行定义

0-1背包问题（NP难）

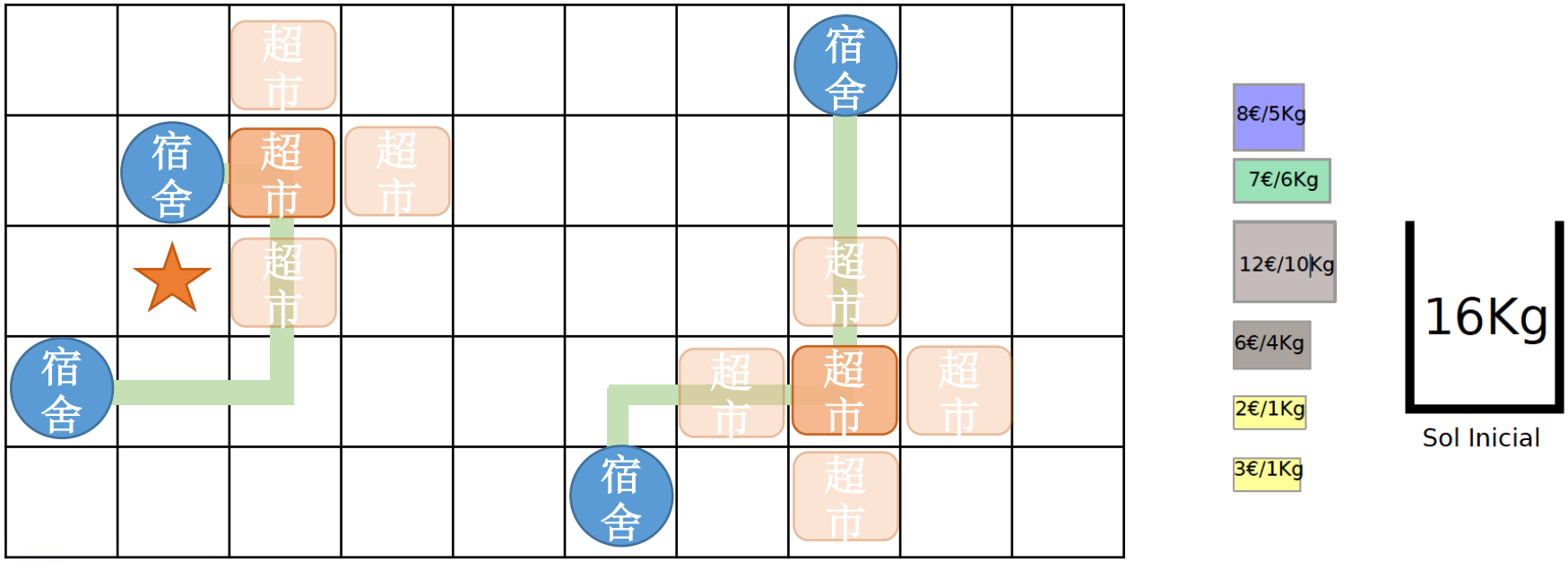

8€/5Kg
7€/6Kg
12€/10Kg
6€/4Kg
2€/1Kg
3€/1Kg
8€/5Kg
12€/10Kg
(1,0,1,0,0,0)
h(n)=22
8€/5Kg
2€/1Kg
12€/10Kg
(1,0,1,0,1,0)
h(n)=23
8€/5Kg
3€/1Kg
12€/10Kg
(1,0,1,0,0,1)
Final Sol
最大重量16kg

0-1背包问题（NP难）


8€/5Kg
7€/6Kg
12€/10Kg
6€/4Kg
2€/1Kg
3€/1Kg
8€/5Kg
12€/10Kg
(1,0,1,0,0,0)
h(n)=22
8€/5Kg
2€/1Kg
12€/10Kg
(1,0,1,0,1,0)
h(n)=23
8€/5Kg
3€/1Kg
12€/10Kg
(1,0,1,0,0,1)
Final Sol
全局最优解
h(n)=24
7€/6Kg
6€/4Kg
8€/5Kg
3€/1Kg
Optimal
最大重量16kg

0-1背包问题（NP难）

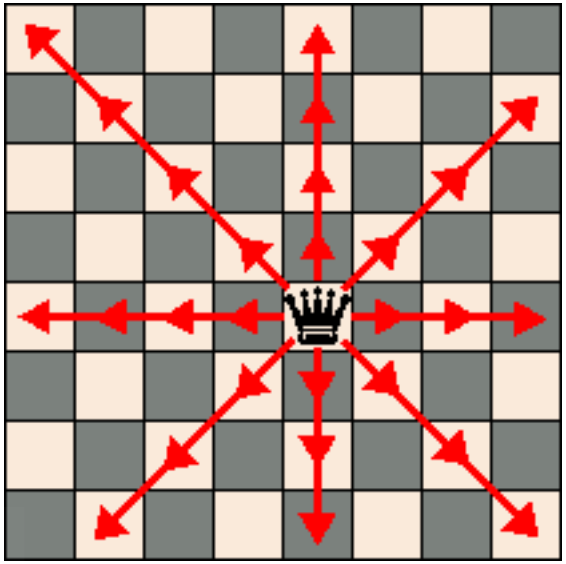
超市选址、背包问题都有明确的待最大化的目标函数



约束满足问题

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

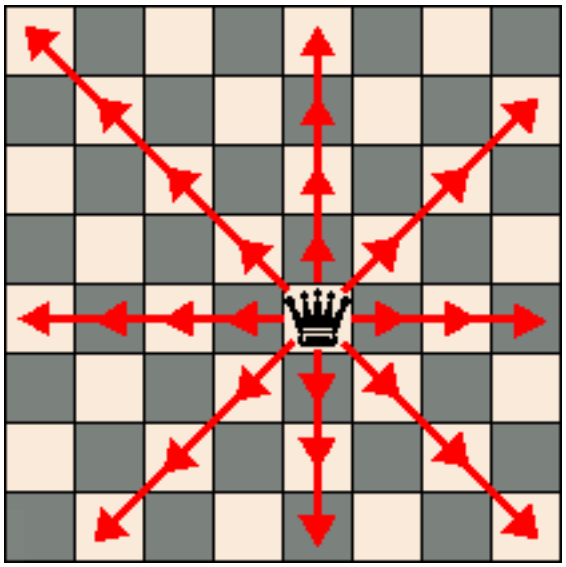
皇后问题（Queens）：在NxN的棋盘中放置N个皇后，找到皇后的摆放位置使得任意两个皇后不能互相攻击

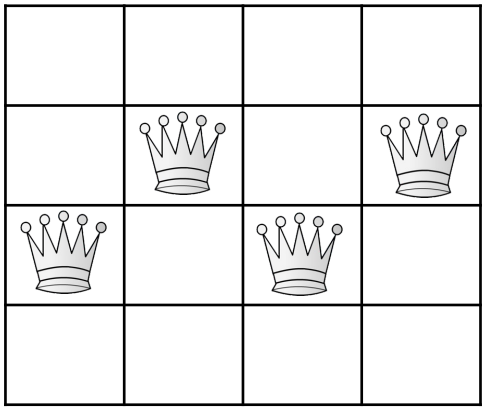


约束满足问题

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

皇后问题（Queens）：在NxN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





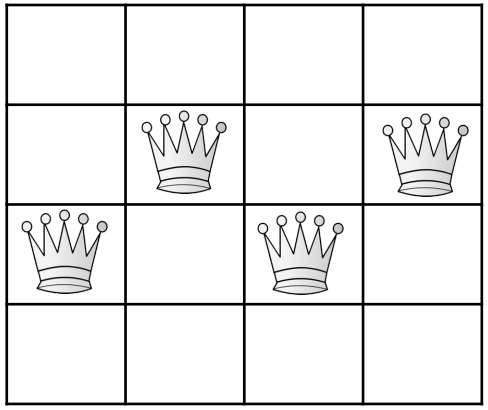
比如以上摆放位置是当前解
如何用登山搜索找到满足要求的解？

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NxN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？

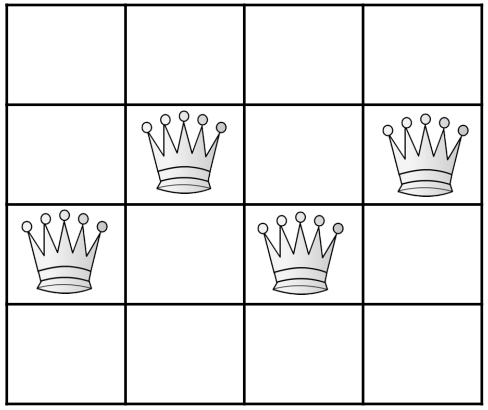
自行定义为当前冲突对的总数
如左图的目标函数值为5

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NxN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

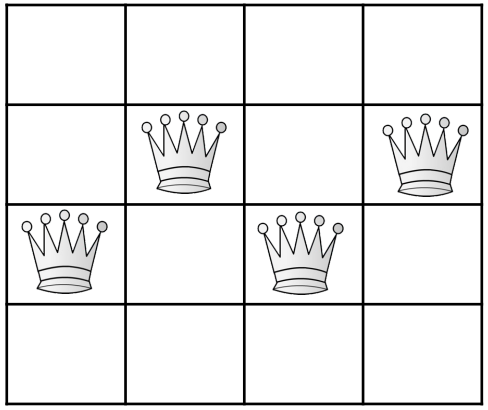
2 如何定义邻域解？

- 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NxN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

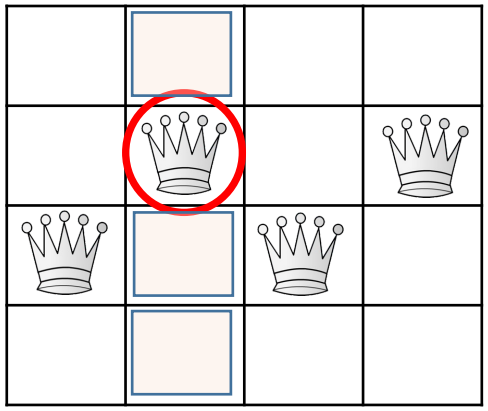
2 如何定义邻域解？

- 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NxN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

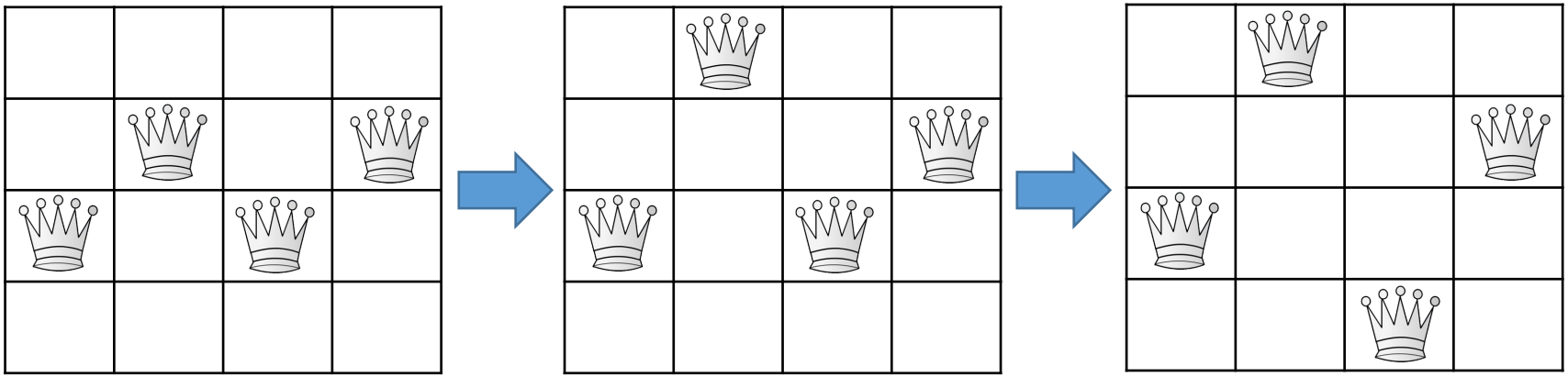
2 如何定义邻域解？

- 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NxN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



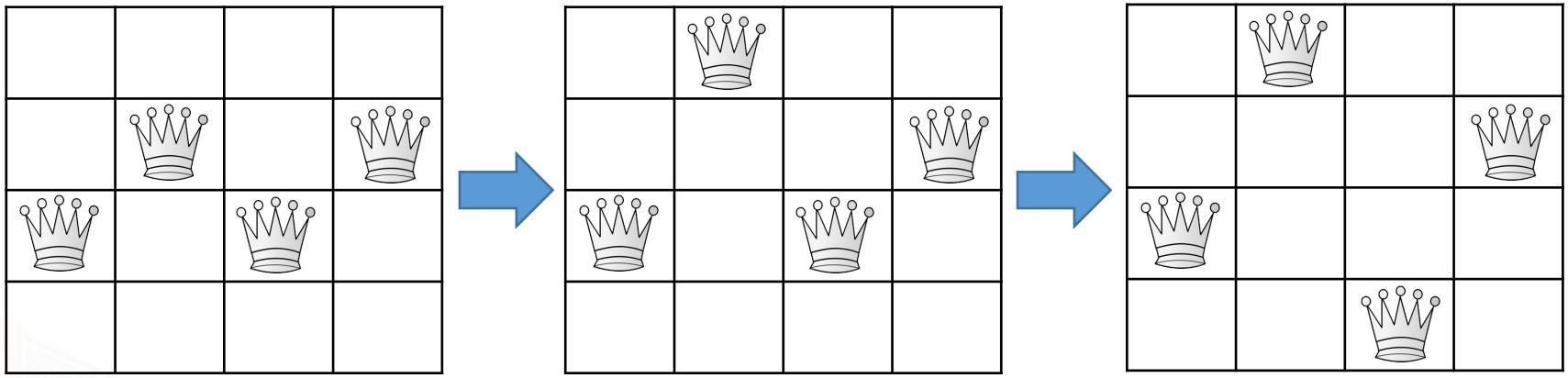
冲突对：5　　冲突对：2　　冲突对：0

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

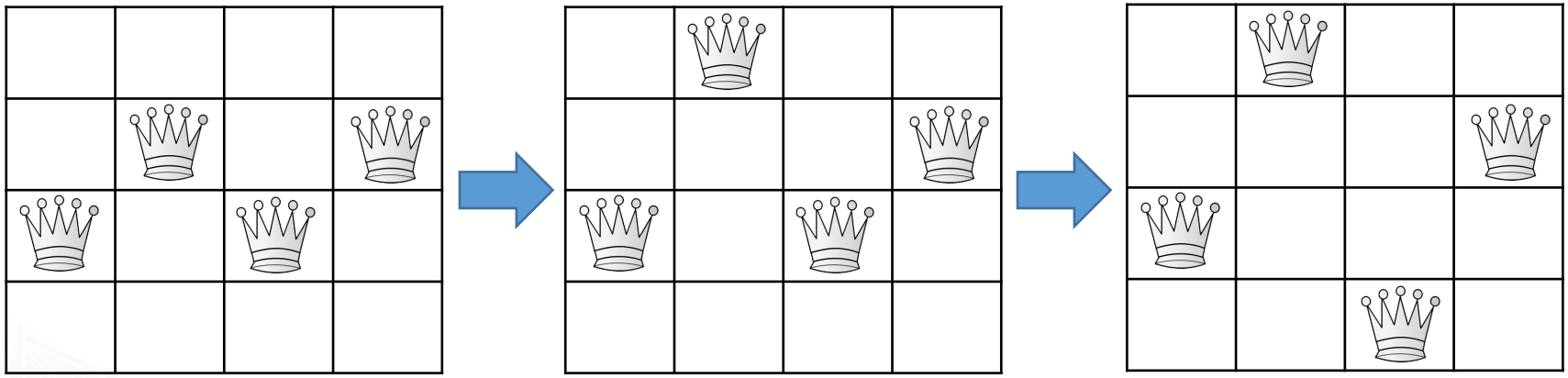
约束满足问题

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$f(\boldsymbol{x}_{\text{current}}) = f(\boldsymbol{x}_{\text{neighbor}})$，
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

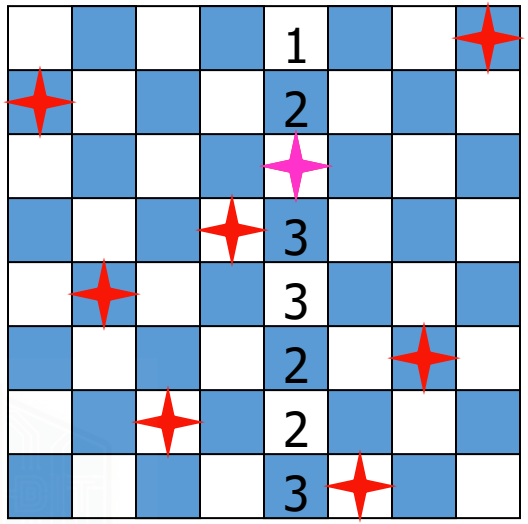
最小冲突启发式（Min-Conflicts Heuristic）算法

生成初始状态（随机）
重复以下过程

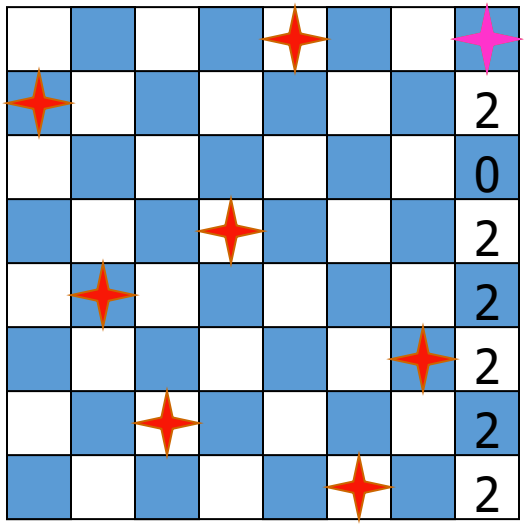
- 选取一个冲突变量
- 重新为该变量赋值，使得冲突次数最小
- 如果新状态没有冲突产生，那么返回该赋值

允许侧向移动（sideways moves）

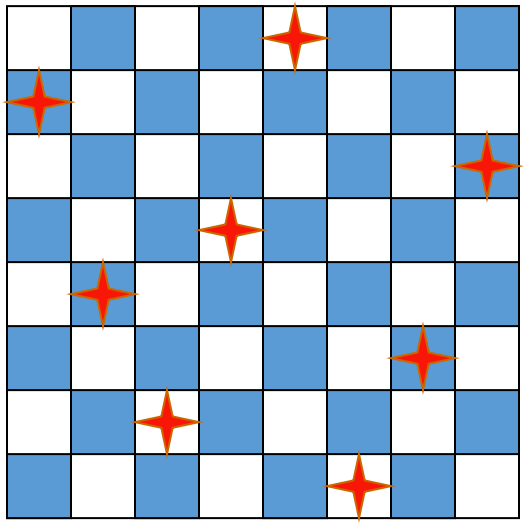
只要还有冲突没有消除，
即便$f(\boldsymbol{x}_{\text{current}}) = f(\boldsymbol{x}_{\text{neighbor}})$，
仍要继续执行算法



冲突对数 1



冲突对数 1

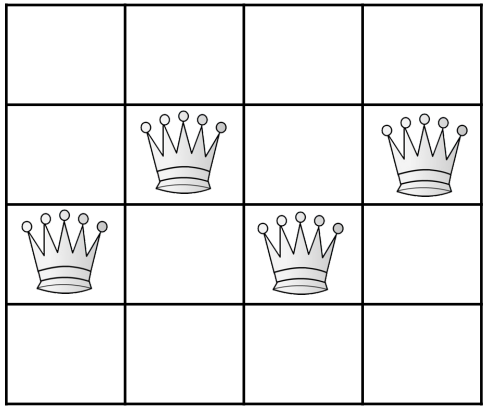


冲突对数 0

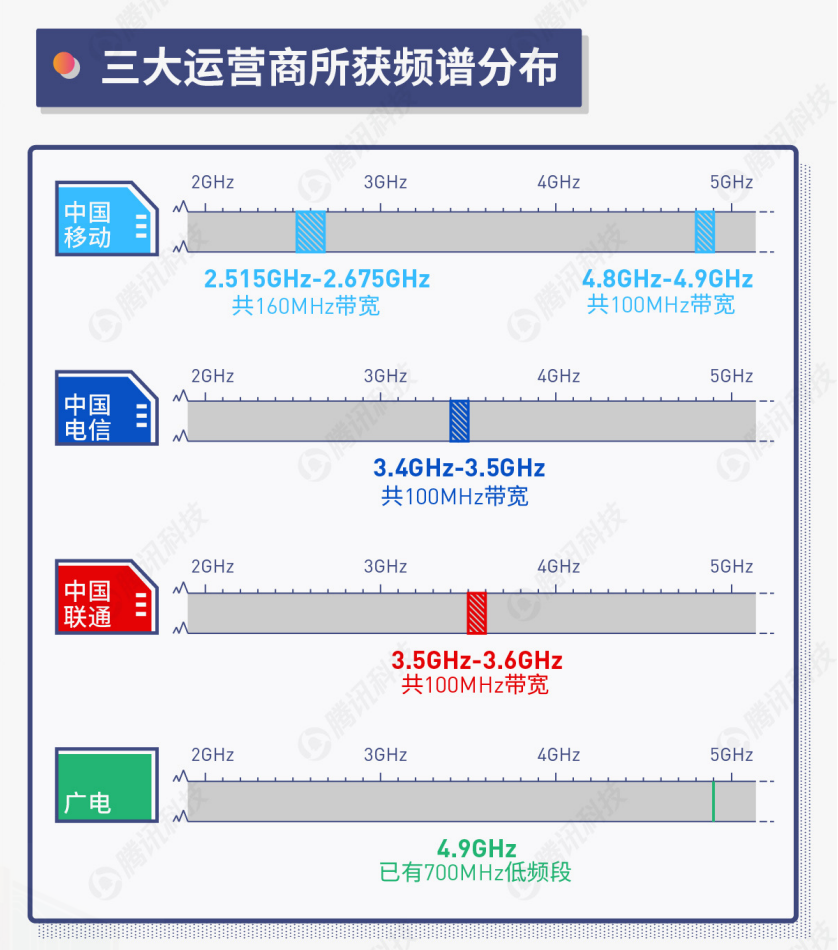
约束满足问题

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

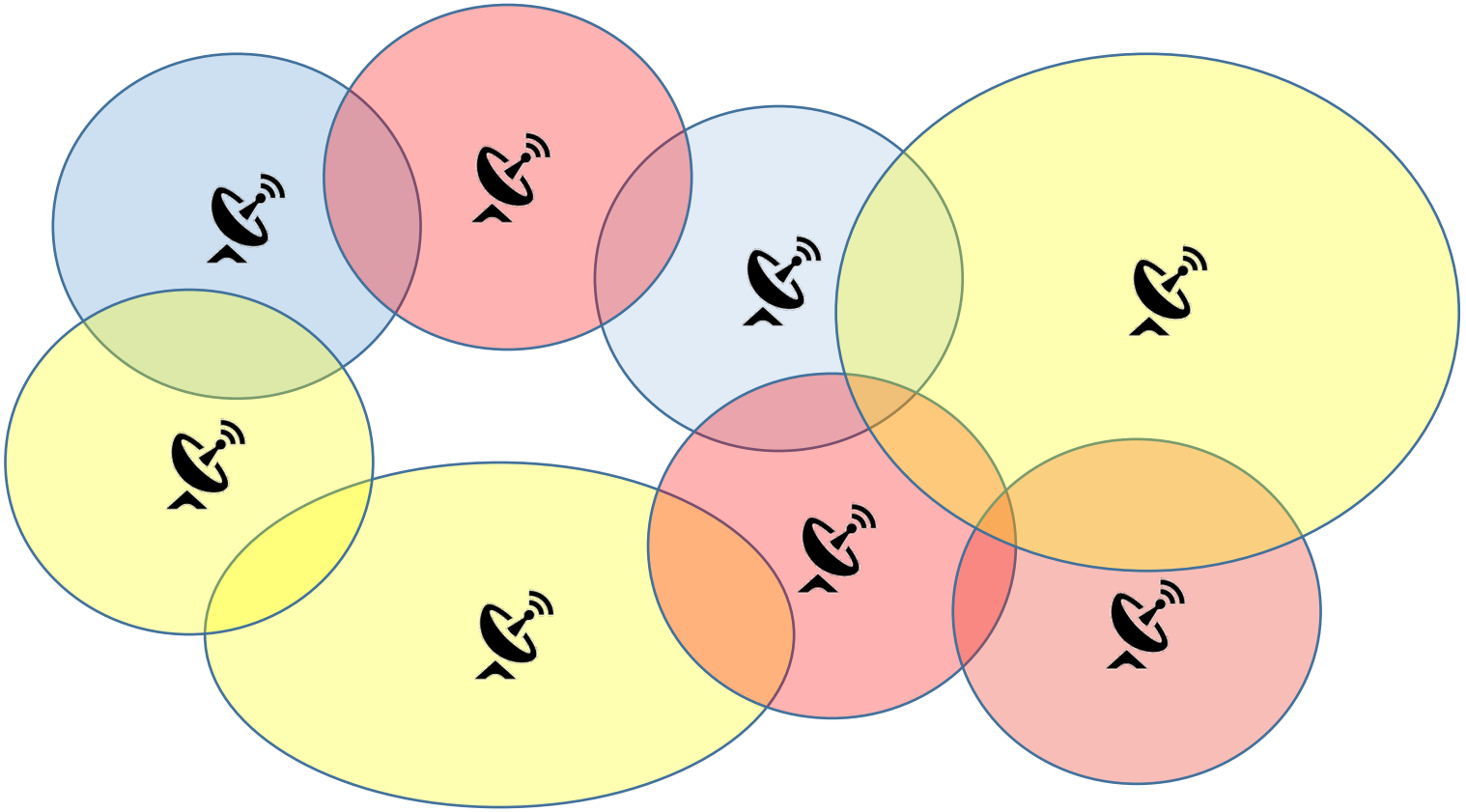


约束满足问题


三大运营商所获频谱分布
中国移动
2GHz
3GHz
4GHz
5GHz
2.515GHz-2.675GHz
共160MHz带宽
4.8GHz-4.9GHz
共100MHz带宽
中国电信
3.4GHz-3.5GHz
共100MHz带宽
中国联通
3.5GHz-3.6GHz
共100MHz带宽
广电
4.9GHz
已有700MHz低频段

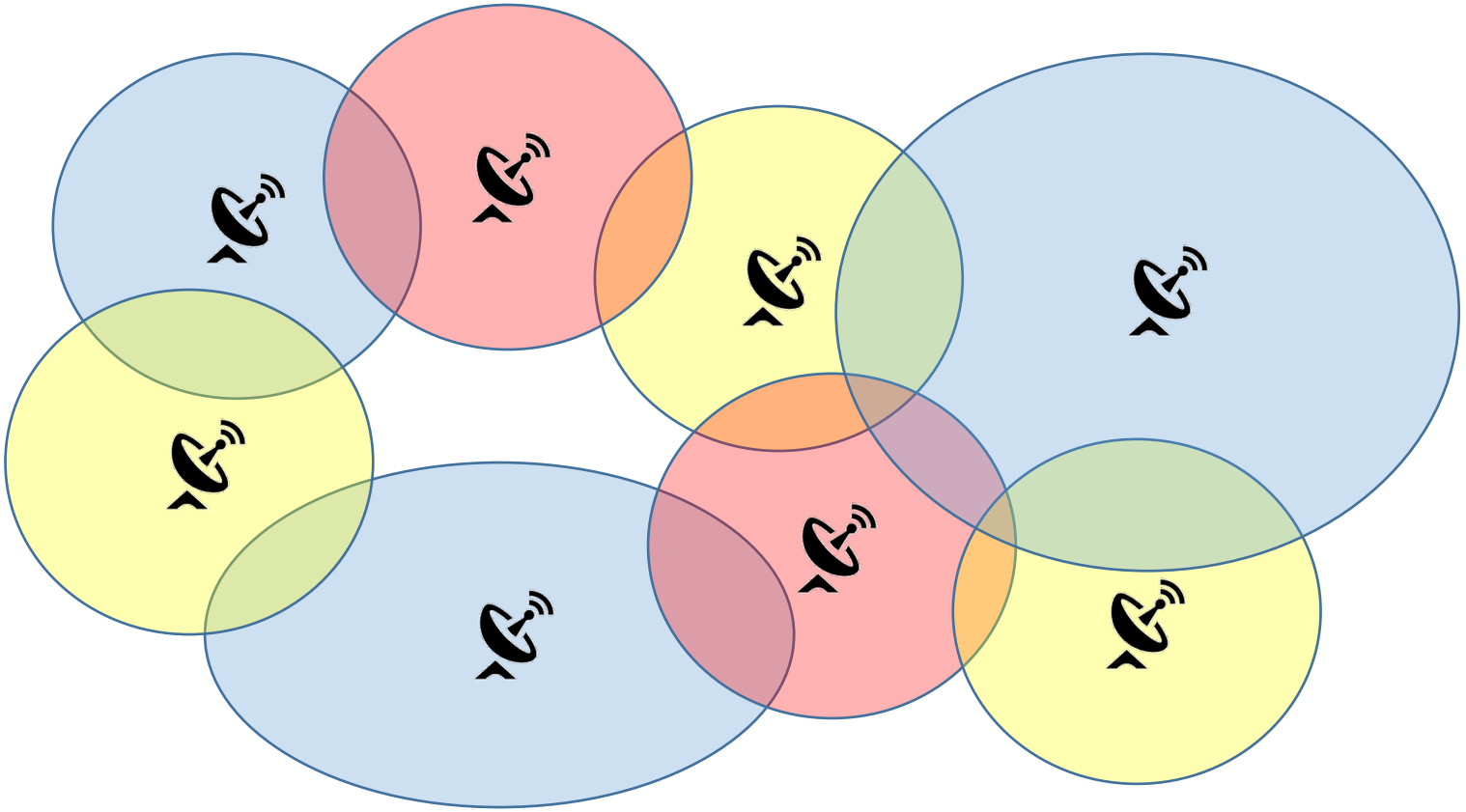
基站

约束满足问题



频谱分配问题

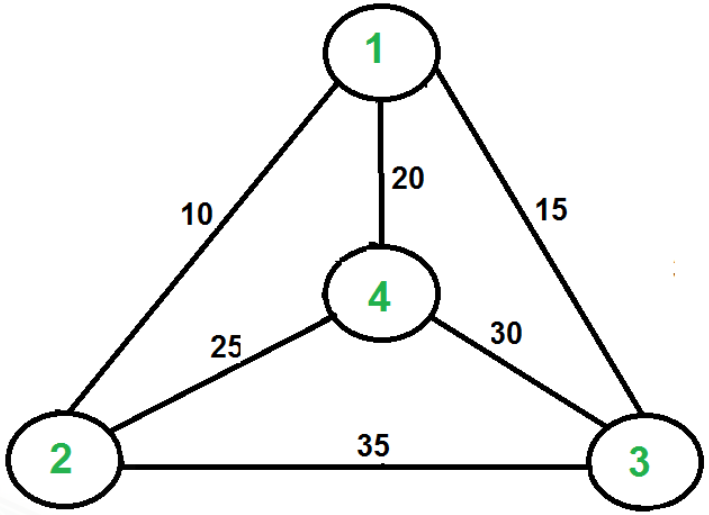
约束满足问题



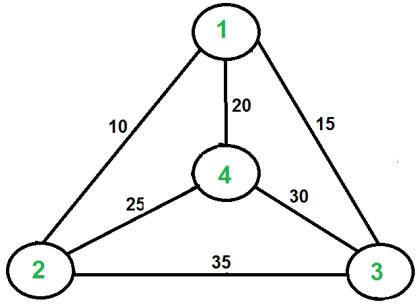
频谱分配问题

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）

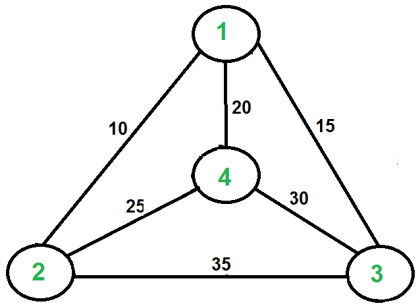
0 初始化当前解$\boldsymbol{x}_{\text{current}}$

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$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$, 结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$, 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$\boldsymbol{x}_{\text{current}}$

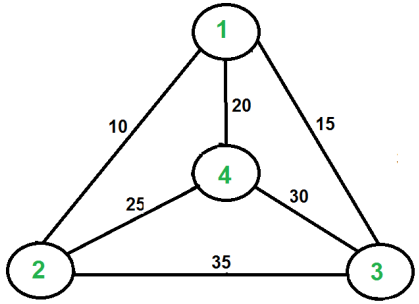
1 寻找当前解的所有邻域解

2 得到对应目标函数值$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，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）

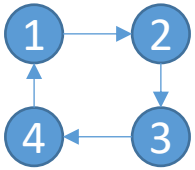
0 初始化当前解$\boldsymbol{x}_{\text{current}}$

1 寻找当前解的所有邻域解

2 得到对应目标函数值$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，返回1

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

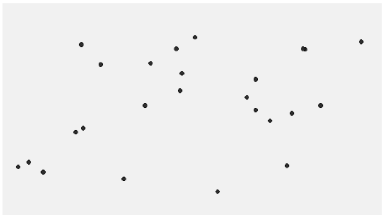


$2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2$
$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$
$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$
$4 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4$

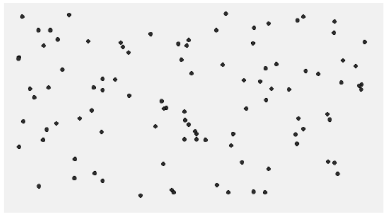
旅行商问题

四个实验场景

a) 15 cities



a) 25 cities



b) 100 cities

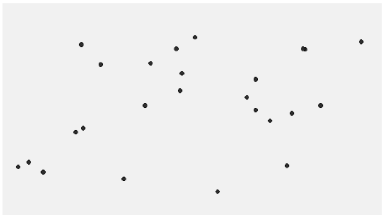
d) 25 cities alternative

实验结果取自Kagan Tumer <ROB537-Learning Based Control>

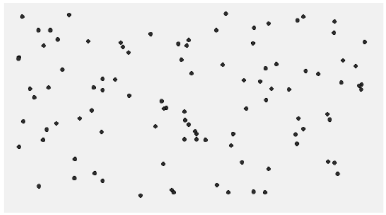
旅行商问题

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.

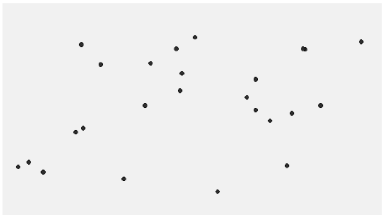
Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	**6200**
	25 Cities	0.188	**9221**
	25 Cities A	0.190	13362
	100 Cities	0.671	**47231**
Evolutionary Algorithm 进化（遗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

实验结果取自Kagan Tumer <ROB537-Learning Based Control>

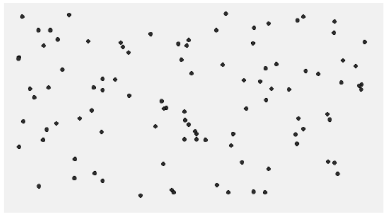
旅行商问题

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.

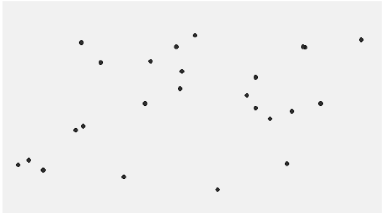
Algorithm	TSP	Time (s)	Cost (m)
Hill Climbing 登山搜索	15 Cities	0.003	8755
	25 Cities	0.013	12541
	25 Cities A	0.014	17567
	100 Cities	0.179	61802
Simulated Annealing 模拟退火	15 Cities	0.105	**6200**
	25 Cities	0.188	**9221**
	25 Cities A	0.190	13362
	100 Cities	0.671	**47231**
Evolutionary Algorithm 进化（遗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

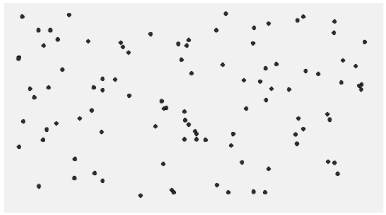
旅行商问题

四个实验场景

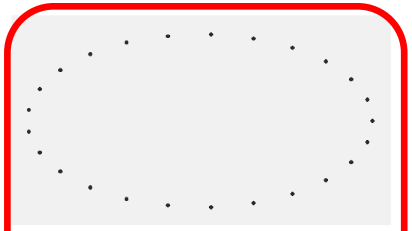
a) 15 cities



a) 25 cities



b) 100 cities



d) 25 cities alternative

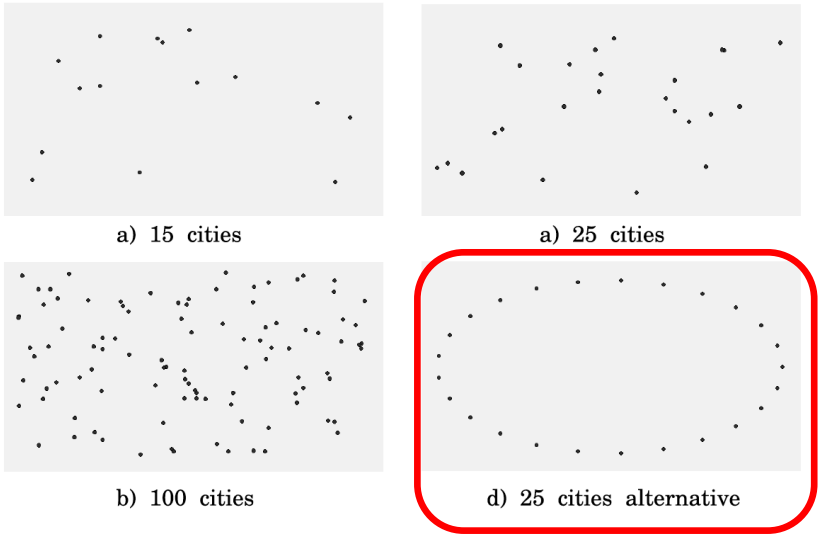
四个实验场景下三个算法的结果

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



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

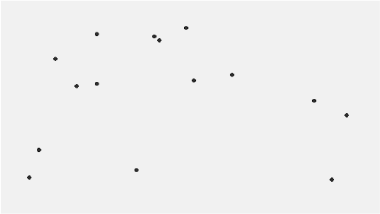
Table 1: Experimental Results. Smallest costs are in bold.

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

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.

评论取自Kagan Tumer <ROB537-Learning Based Control>

旅行商问题



a) 15 cities

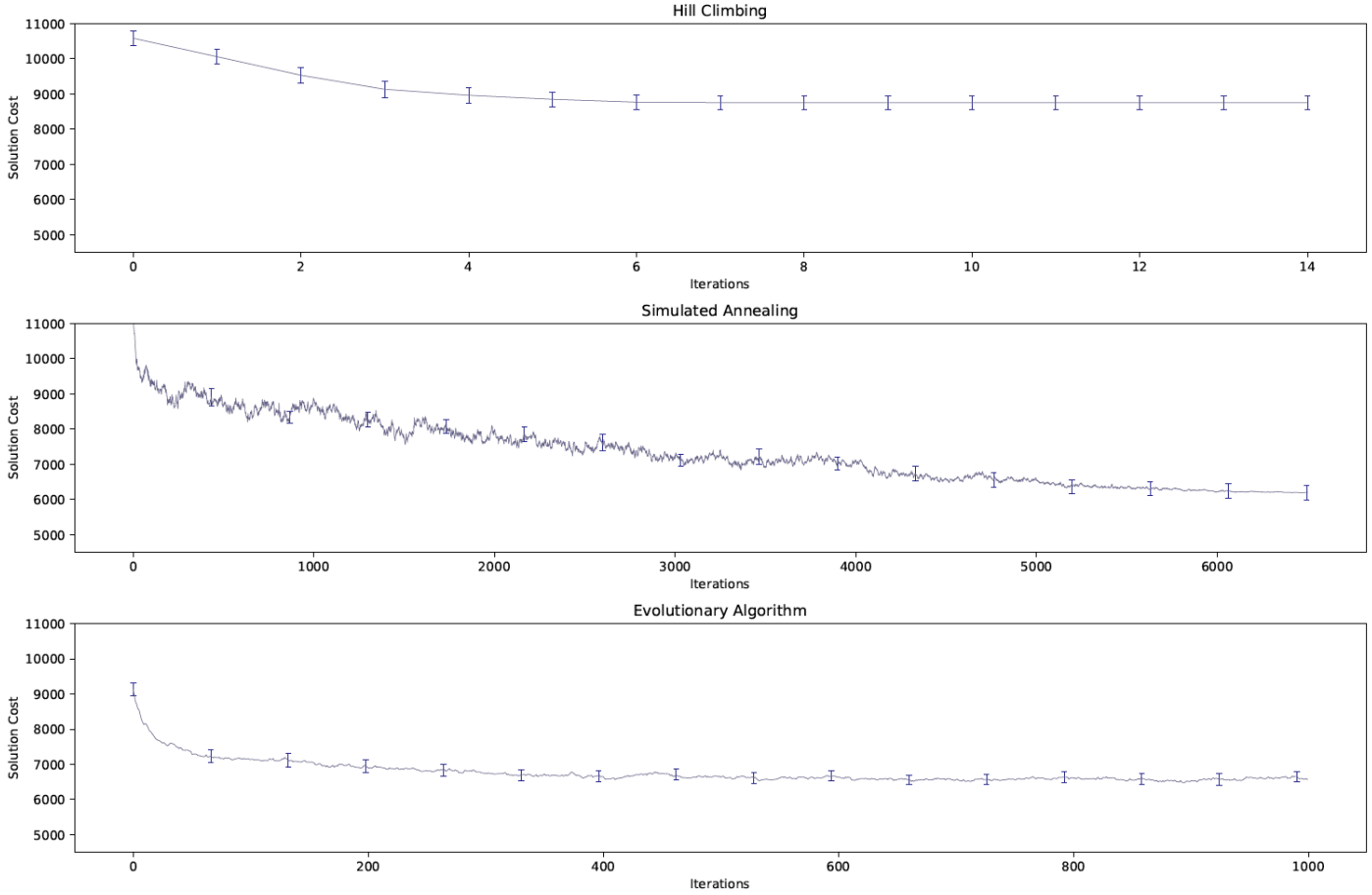
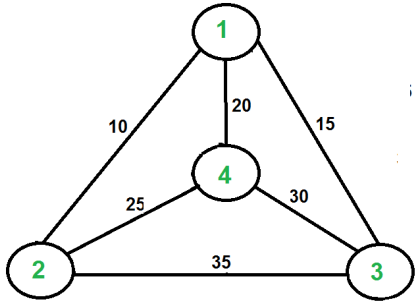


Figure 2: Performance curves for the three algorithms for a TSP with 15 cities.

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$\boldsymbol{x}_{\text{current}}$

1 寻找当前解的所有邻域解

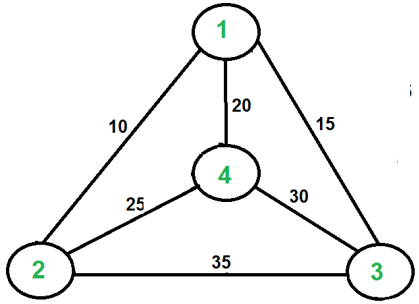
2 得到对应目标函数值$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，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

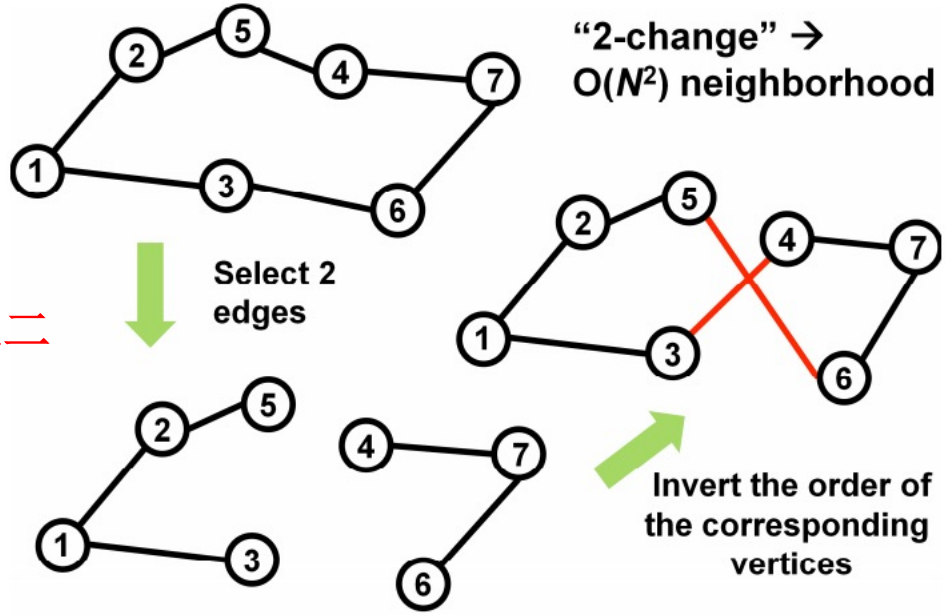
有没有其它定义邻域的方法？

旅行商问题



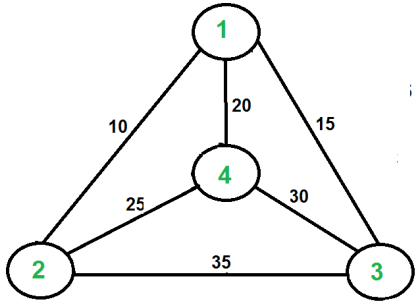
旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。找到总长度最小的路径（从一个城市出发、不重复的遍历所有城市并回到起点）

定义邻域方法之二



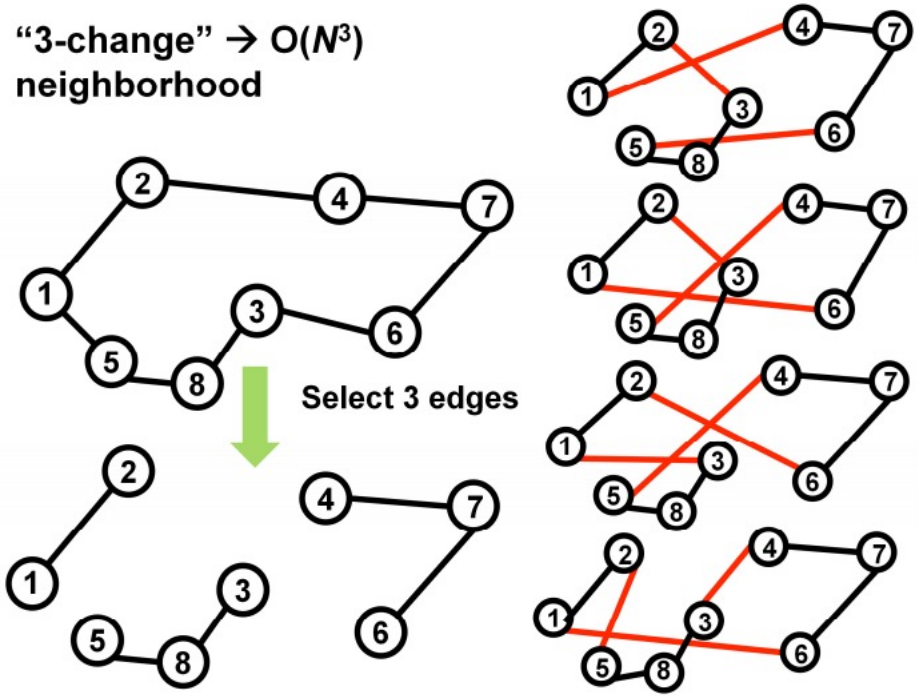
图片取自Doina Precup COMP-424<Artificial Intelligence>

旅行商问题

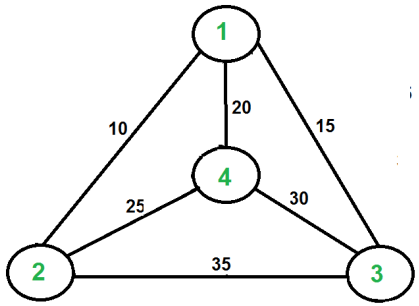


旅行商问题：给定n个城市，任意两城市间有路相连且路程距离已知。找到总长度最小的路径（从一个城市出发、不重复的遍历所有城市并回到起点）

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）

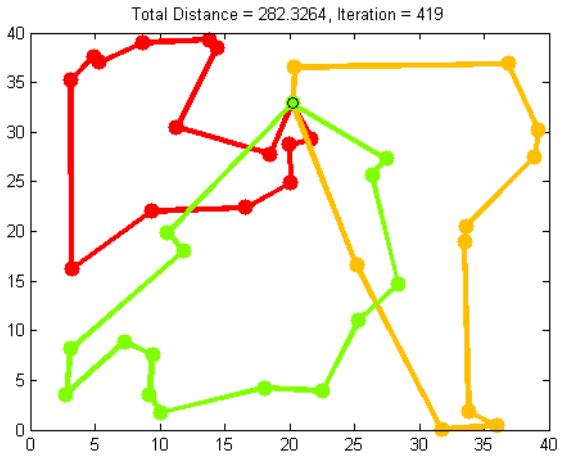
- A smaller neighborhood means fewer neighbors to evaluate (so cheaper computation, but possibly worse solutions)
- A bigger neighborhood means more computation, but maybe fewer local optima, so better final result
- Defining the set of neighbors is a *design choice* (like choosing the heuristic for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

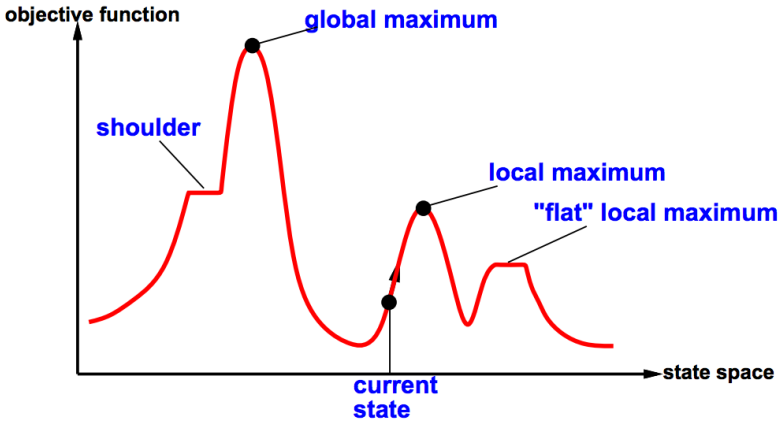
Part1 如何从复杂度的角度衡量算法好坏？
Part2 针对一些有特定形式的最优化问题，如何找到最优解？
Part3 针对较复杂的最优化问题（如NP难问题），如何找到较好解？
局部搜索、模拟退火、遗传算法、差分进化算法、蚁群算法、粒子群优化算法、人工蜂群算法
Part4 针对较复杂的多目标最优化问题，如何找到较好解？

登山搜索的局限

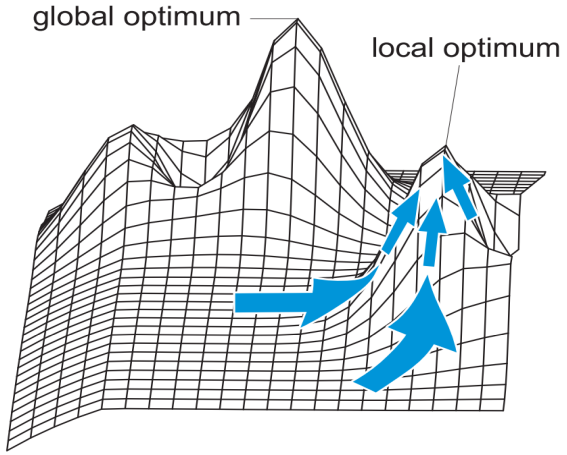
算法最后输出的解不一定是全局最优解，那么具体有哪些可能？

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？

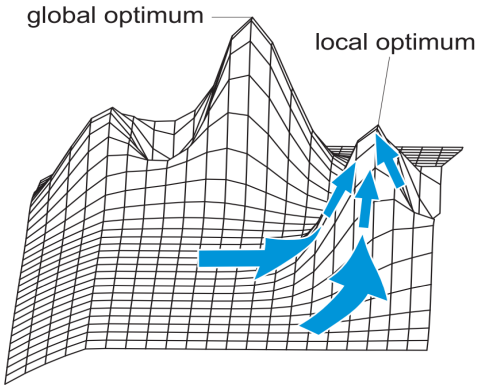


单个变量的情况



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$\boldsymbol{x}_{\text{current}}$

1 寻找当前解的所有邻域解

2 得到对应目标函数值$f(.)$最高的邻域解$\boldsymbol{x}_{\text{neighbor}}$

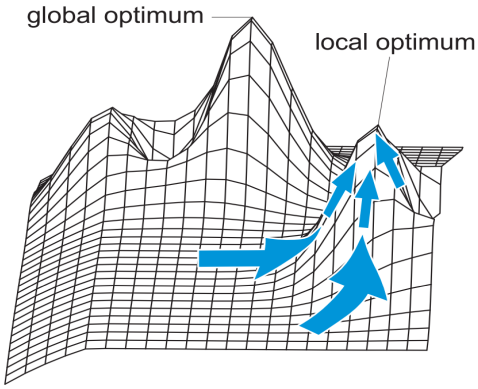
3 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，返回1

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

2 若$f(\boldsymbol{x}_{\text{current}}) \geq f(\boldsymbol{x}_{\text{neighbor}})$，结束搜索；否则，$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$，返回1

下一步不一定会跳到邻域中质量最高的解

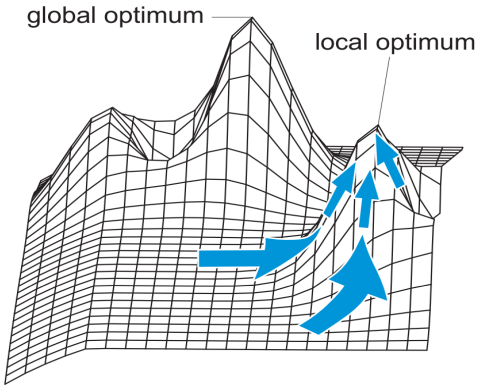
当邻域的解数目很多时，效果比较好

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

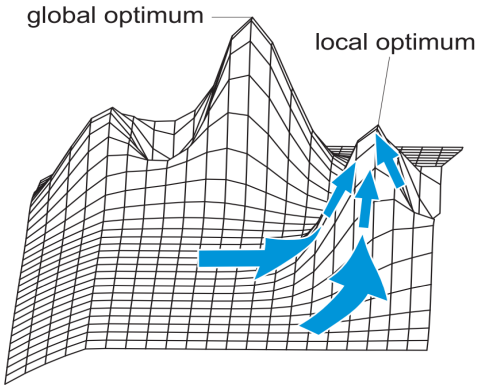
0 初始化当前解$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

2 以概率$\frac{1}{1+e^{\frac{f(x_{\text{current}})-f(x_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

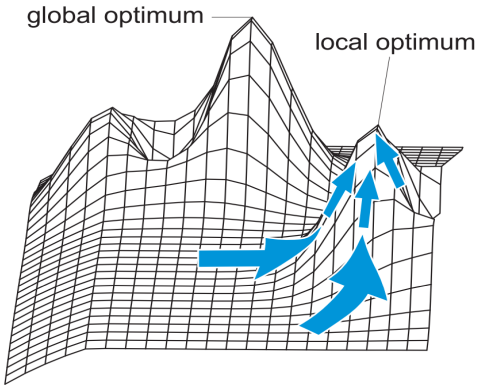
2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

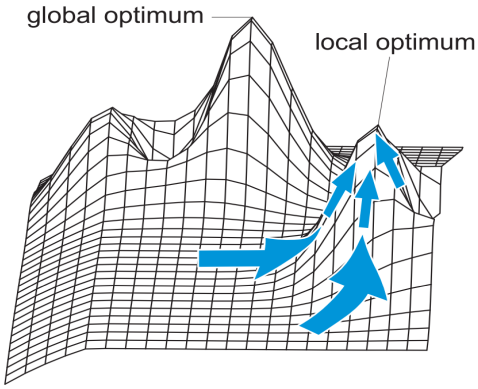
2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

3 若算法终止条件不满足，返回1

怎么推导而来的？

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

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

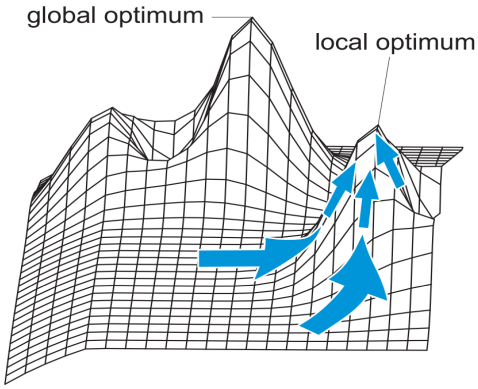
2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

3 若算法终止条件不满足，返回1

替换的概率 $\frac{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}}{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}+e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}$

不替换的概率 $\frac{e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}+e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}$

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

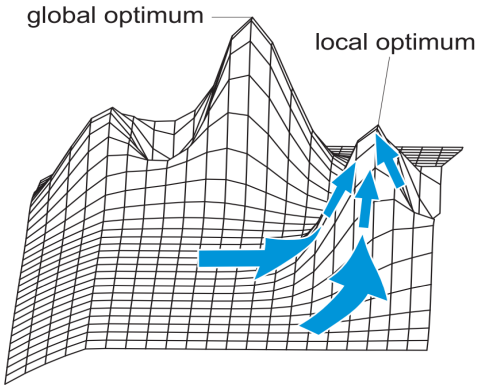
2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

3 若算法终止条件不满足，返回1

替换的概率 $\frac{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}}{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}+e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}$　　不替换的概率 $\frac{e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}{e^{\frac{f(\boldsymbol{x}_{\text{neighbor}})}{T}}+e^{\frac{f(\boldsymbol{x}_{\text{current}})}{T}}}$

用指数形式而非$\frac{f(\boldsymbol{x}_{\text{neighbor}})}{f(\boldsymbol{x}_{\text{neighbor}})+f(\boldsymbol{x}_{\text{current}})}$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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

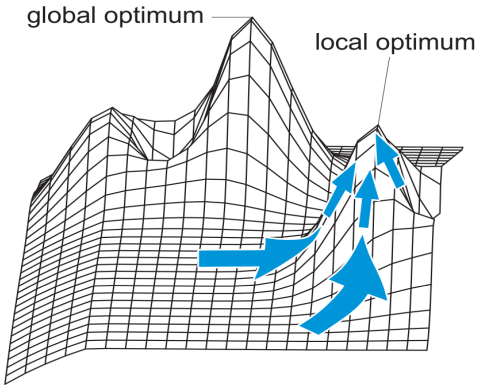
2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

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？

$\boldsymbol{x}_{\text{current}}, \boldsymbol{x}_{\text{neighbor1}}, \boldsymbol{x}_{\text{neighbor2}}$　　$f(\boldsymbol{x}_{\text{current}}), f(\boldsymbol{x}_{\text{neighbor1}}), f(\boldsymbol{x}_{\text{neighbor2}})$

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$\boldsymbol{x}_{\text{current}}$

1 随机选择一个邻域解$\boldsymbol{x}_{\text{neighbor}}$

2 以概率$\frac{1}{1+e^{\frac{f(\boldsymbol{x}_{\text{current}})-f(\boldsymbol{x}_{\text{neighbor}})}{T}}}$令$\boldsymbol{x}_{\text{current}} \leftarrow \boldsymbol{x}_{\text{neighbor}}$

3 若算法终止条件不满足，返回1

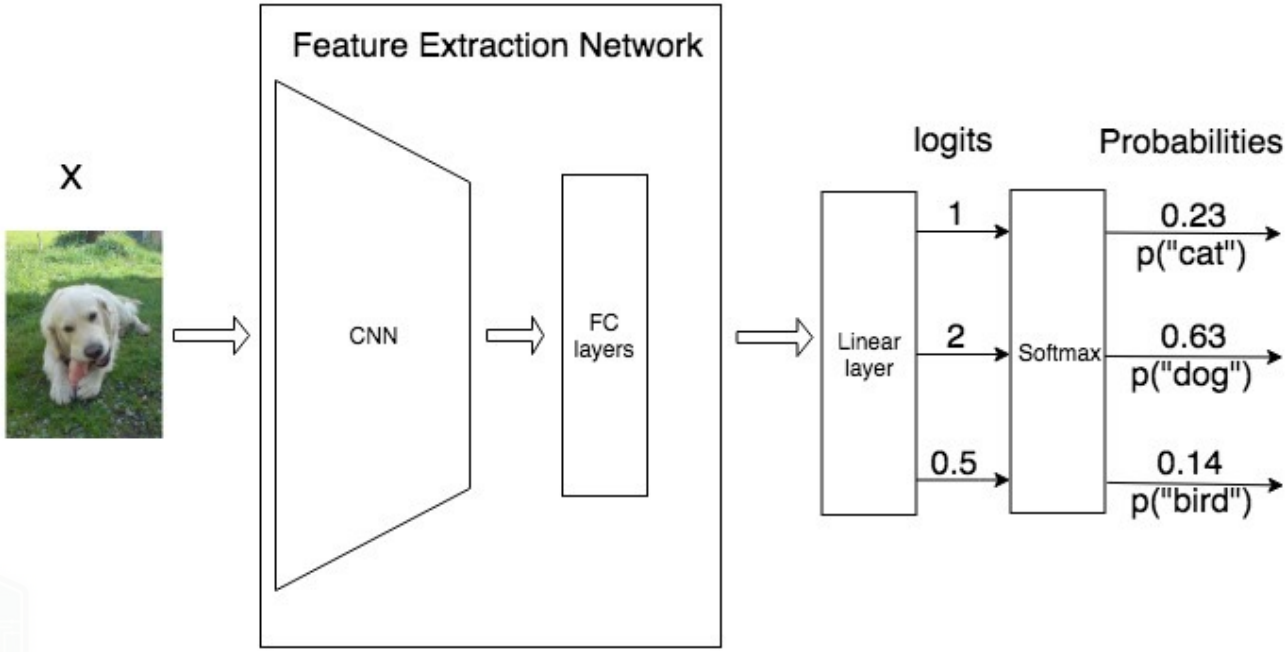
The standard (unit) softmax function $\sigma : \mathbb{R}^K \to [0,1]^K$ is defined by the formula

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

随机登山搜索

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

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



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

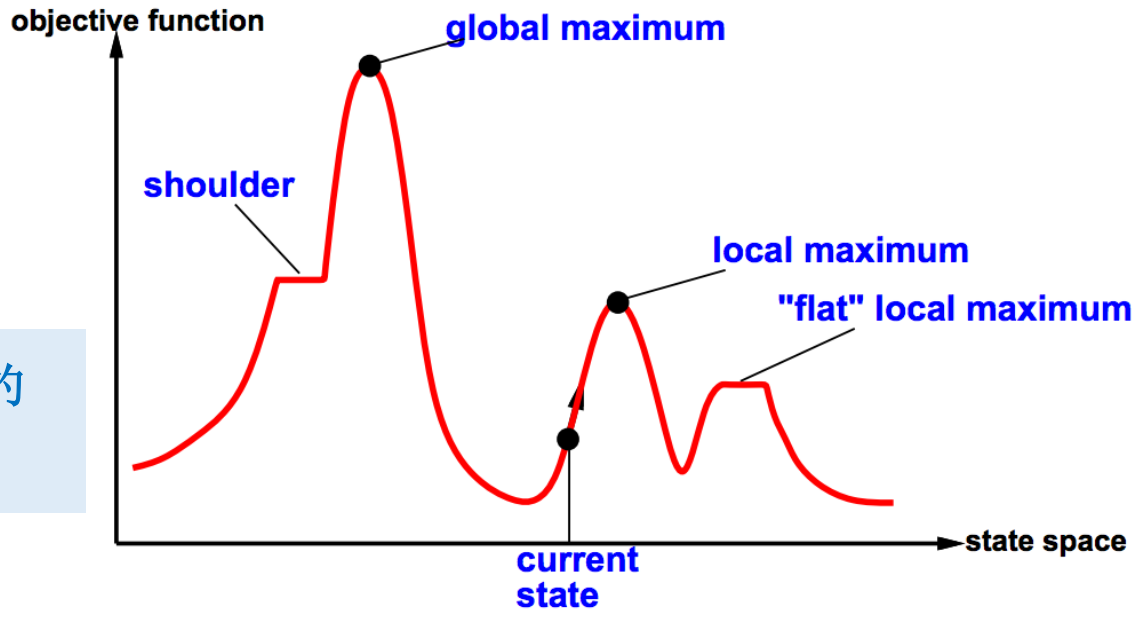
侧向移动（Sideways moves）

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



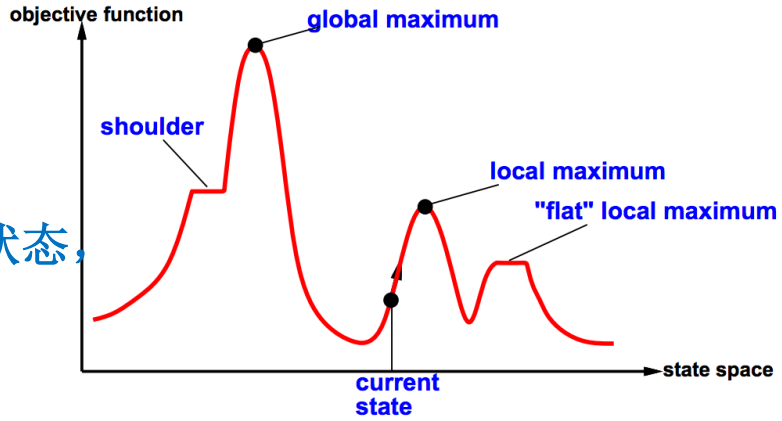
改进登山搜索

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）

sideways moves (M):
M=100 → 94% solved instances
for the 8-queens!
21 steps avg. on success
64 steps avg. on "failure"

侧向移动登山搜索

random restarts:
100% solved instances
28 steps avg.

随机重启登山搜索

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：$\boldsymbol{x}_{\text{current},1}, \boldsymbol{x}_{\text{current},2}, \cdots, \boldsymbol{x}_{\text{current},K}$

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

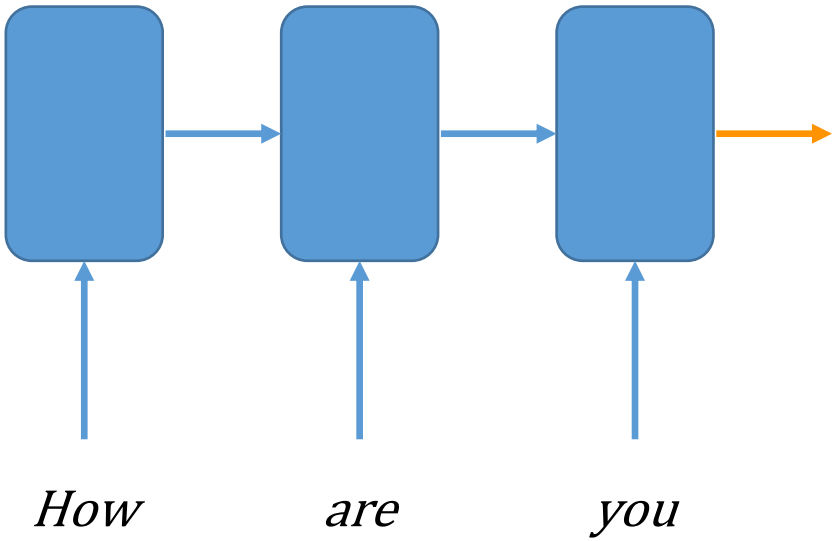
各个点之间共享信息（代价是存储空间）

随机重启的登山算法（分别）
迭代一次后解位置

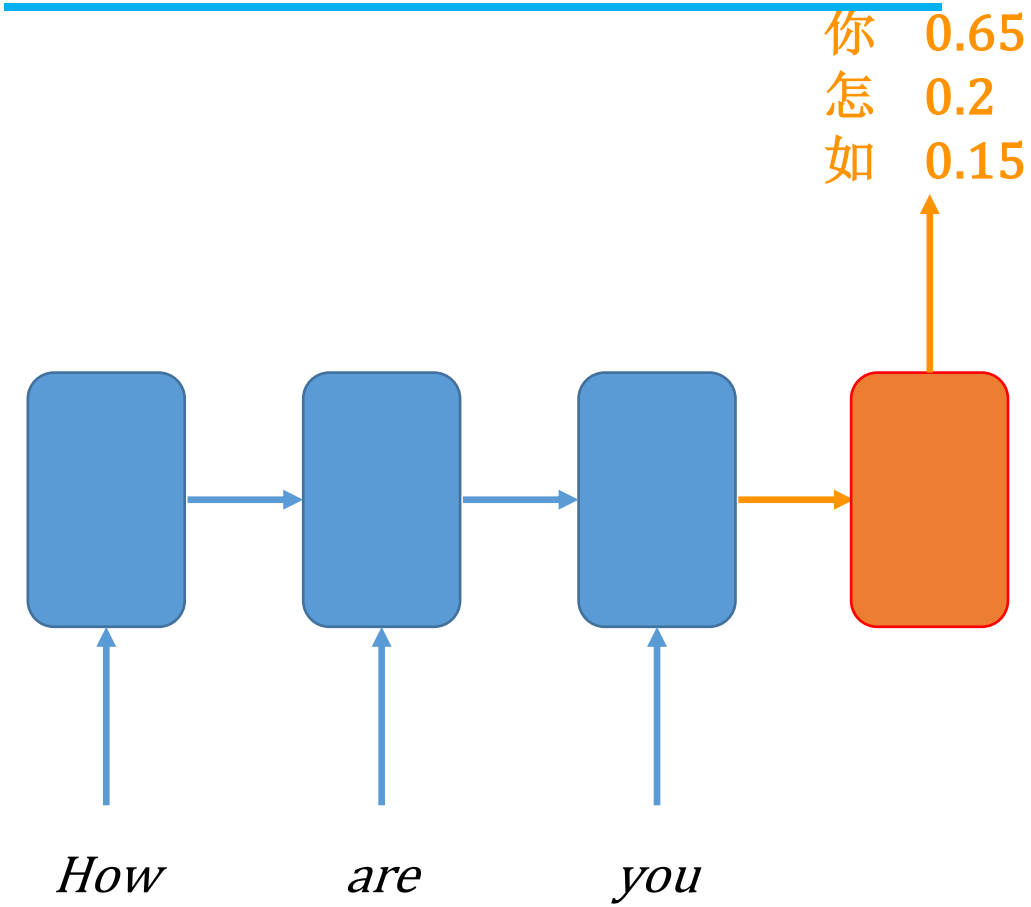
可以先后执行，不占额外的存储空间

束搜索在机器翻译中的应用

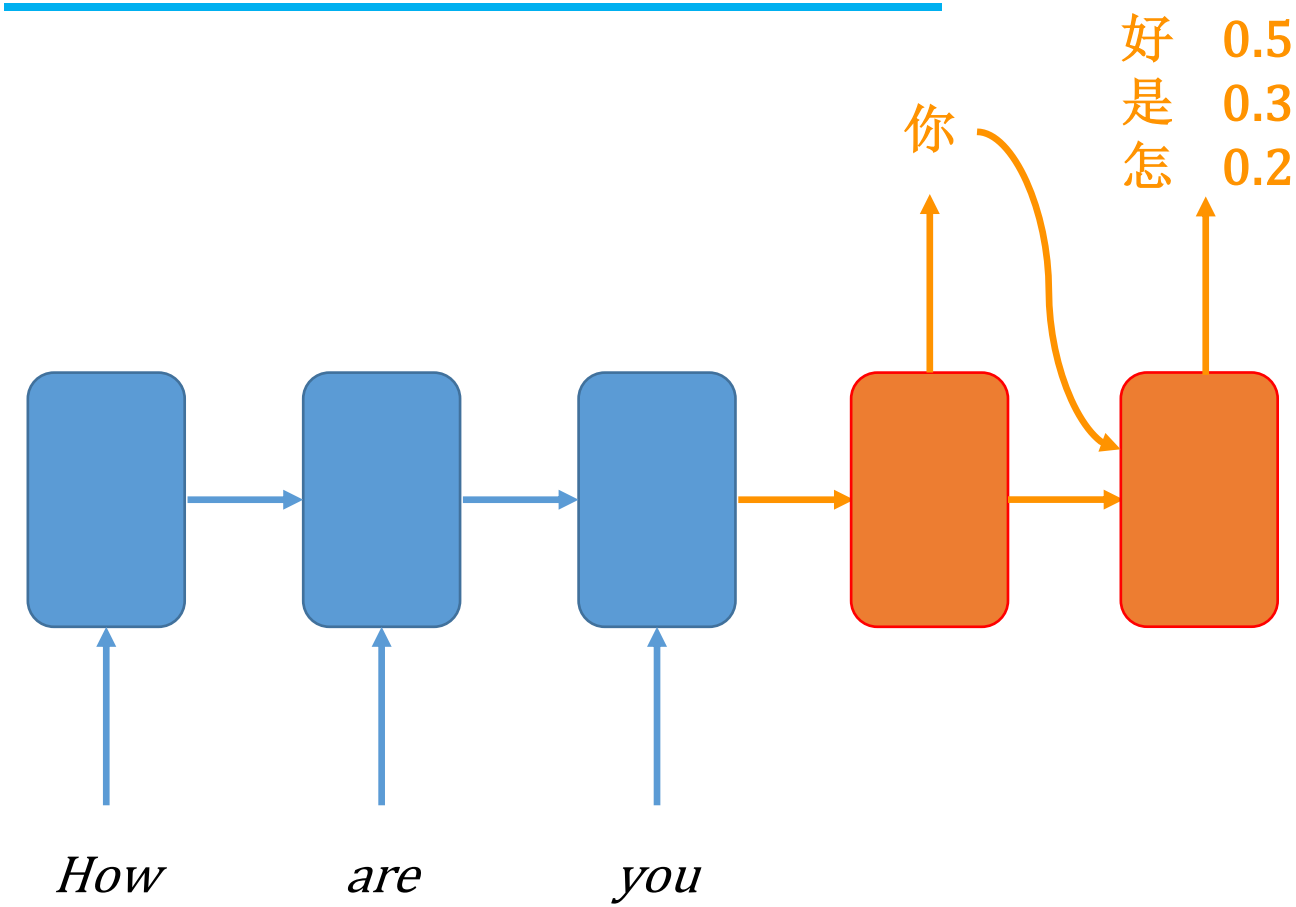
基于循环神经网络（RNN）的原始翻译方式：



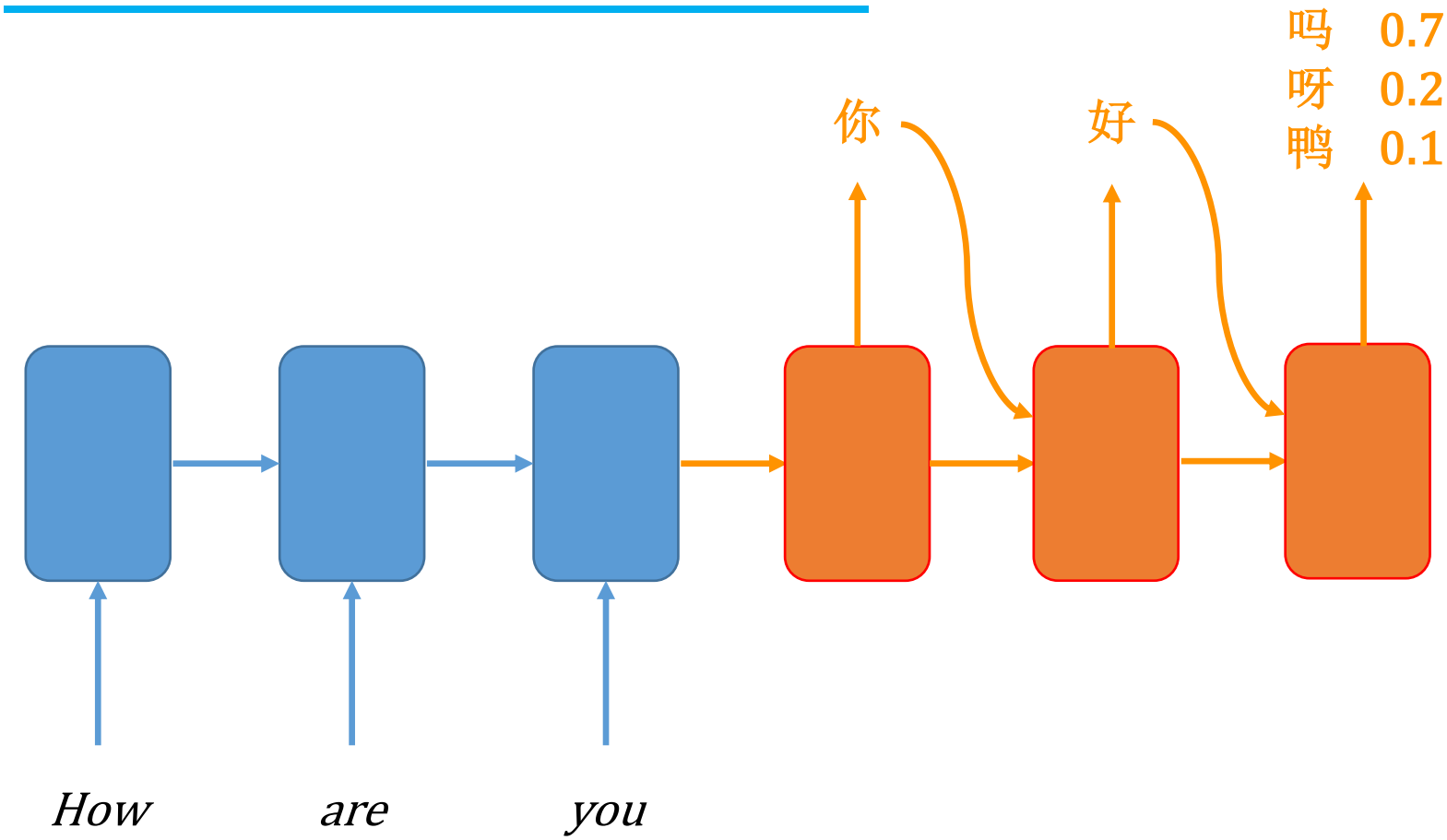
束搜索在机器翻译中的应用



束搜索在机器翻译中的应用



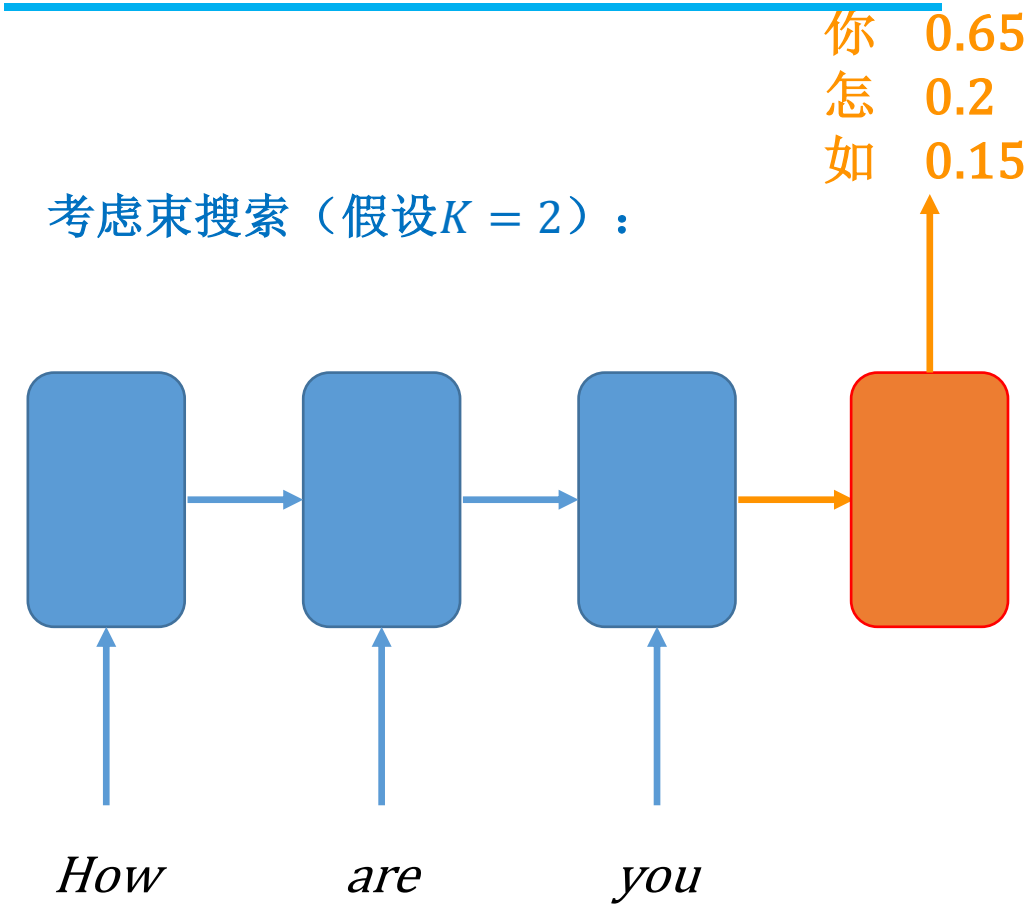
束搜索在机器翻译中的应用



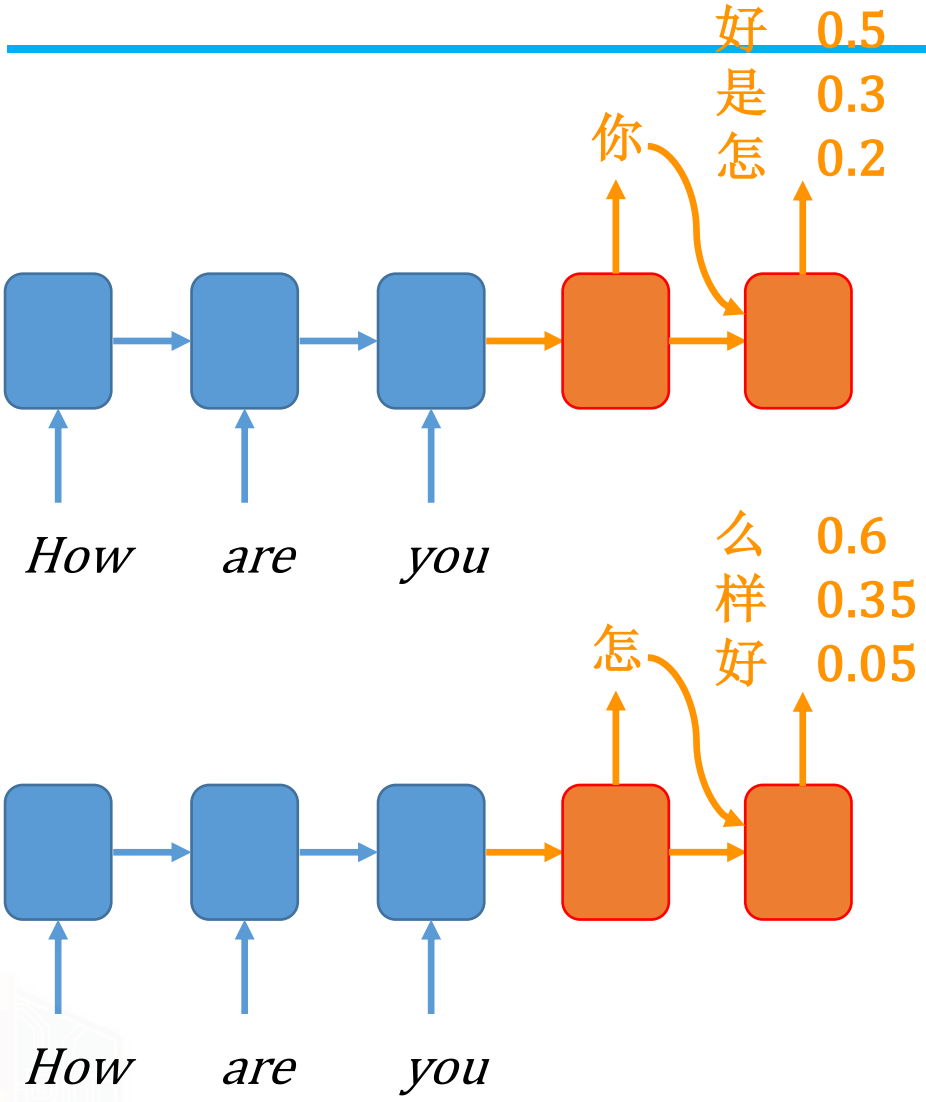
非常依赖对第一个字翻译的准确度

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



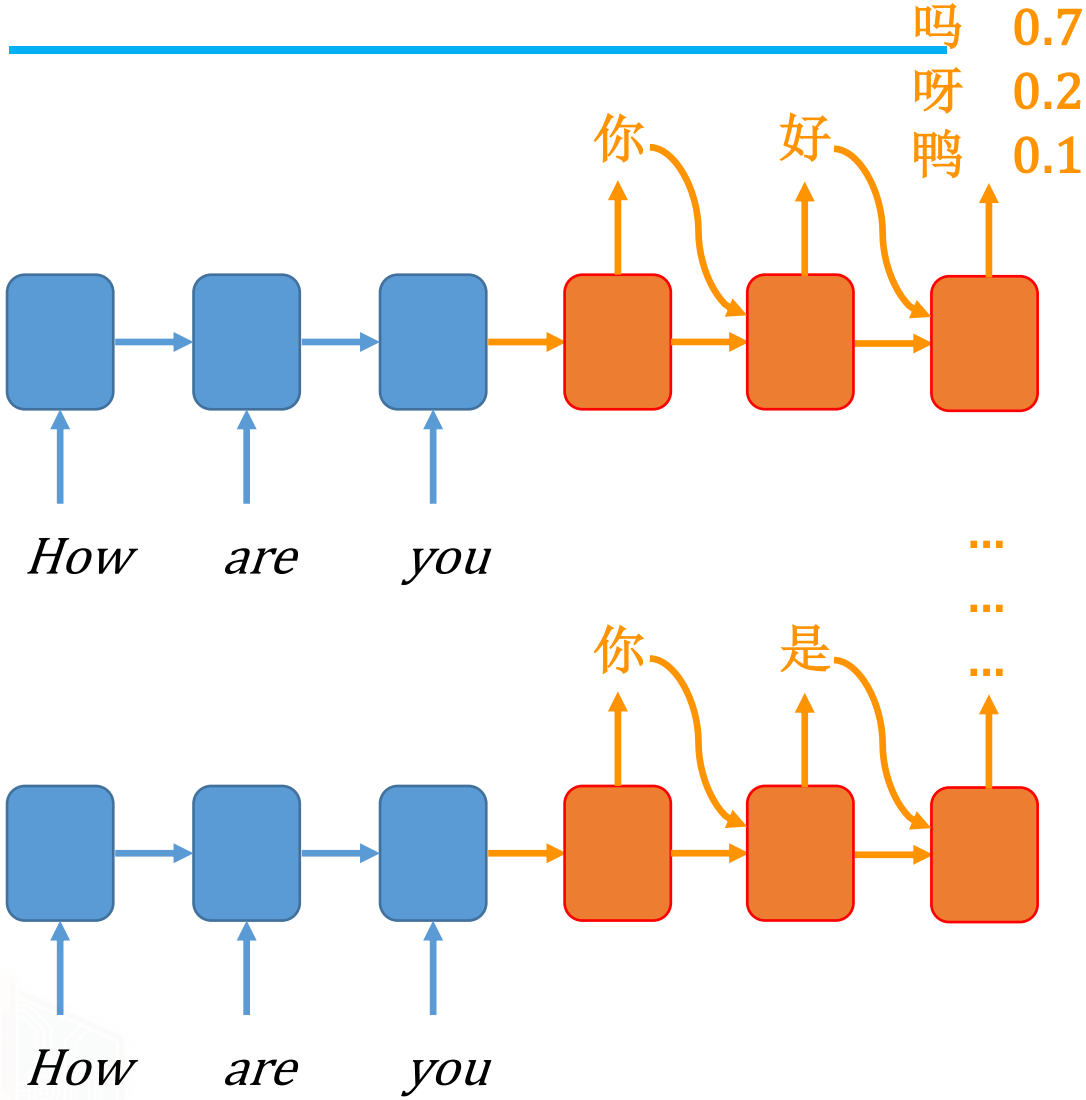
计算：

P(你好|How are you)
P(你是|How are you)
P(你怎|How are you)
P(怎么|How are you)
P(怎样|How are you)
P(怎好|How are you)

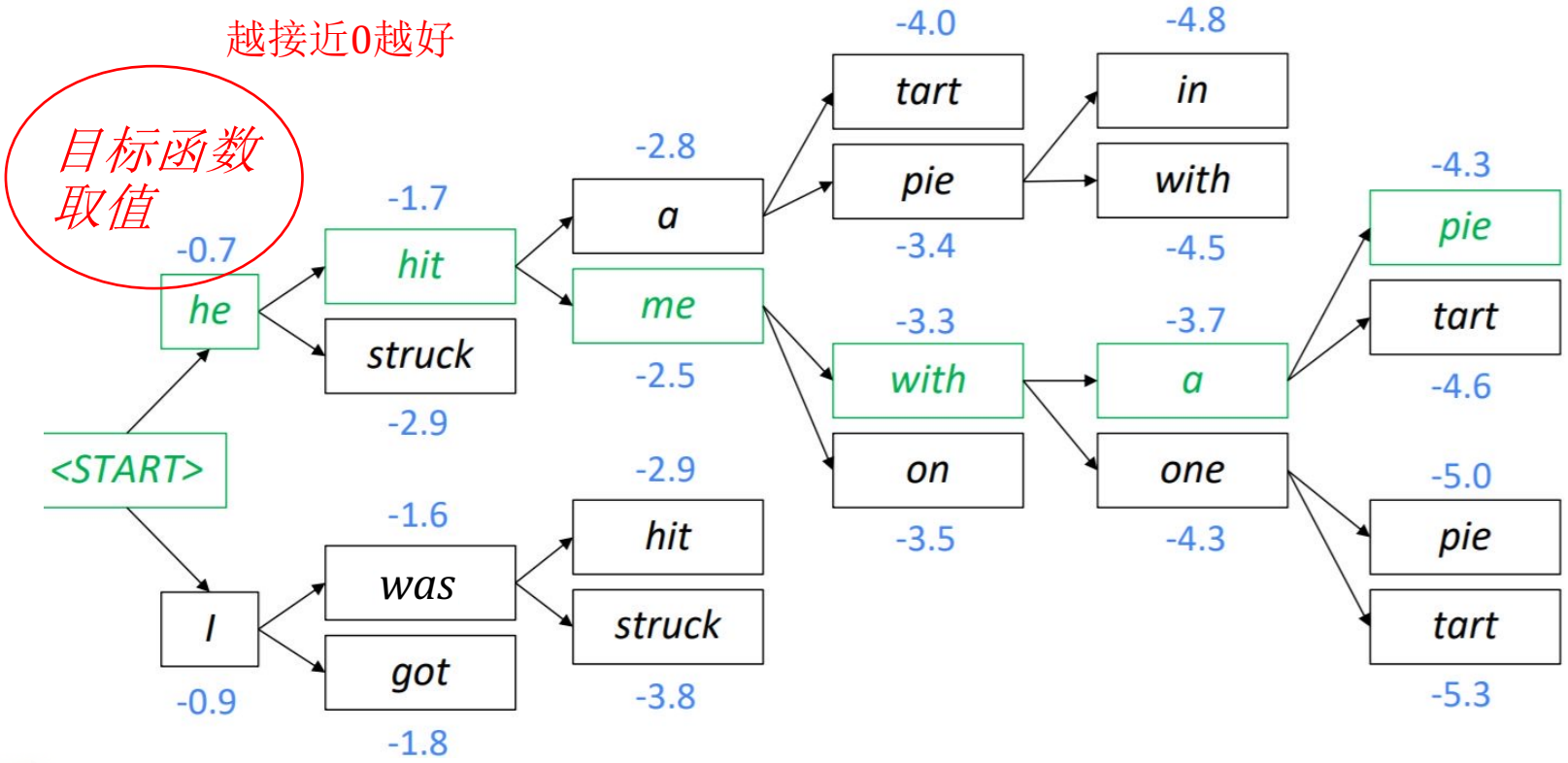
选出其中概率最高的两项

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

局部搜索之登山搜索、局部束搜索

计算机学院
余皓然
2024/3/25

主要参考资料

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！

计算机学院
余皓然
2024/3/25