利用模拟退火算法解决泊位分配问题

数据科学与计算机学院 软件工程（移动信息工程）15352138 黄慰欣

摘要：泊位是集装箱码头的稀缺资源，合理制定靠泊计划，提高泊位利用率，是集装箱码头不断发展的重要途径。港口泊位分配问题（Berth Allocation Problem），实质是为到达集装箱港口的船舶安排最佳停靠位置与停靠时间。本文将采用贪心算法和模拟退火算法来解决泊位分配问题，并比较两种算法的效率。

关键词：泊位分配问题；贪心算法；模拟退火算法

1 泊位分配问题介绍

随着经济贸易全球化，科学技术的进步与发展，集装箱港口在国际物流与国民经济中具有重要的地位。然而，随着吞吐量日益增长，港口泊位的分配成为了决定港口运行效率的一个重要因素。合理的泊位调度方案，既能有效提高港口的通过能力，又能满足客户对服务水平提高的要求。

目前大多数港口采取先到先服务（FCFS）的人工调度方式，这种方法易操作，但当船型及其作业量有明显差别时，这种方法不利于总的船舶在港停留时间以及运营成本的降低。

因此，我们想要解决的问题就是在时间空间一定的条件下，如何才能让泊位能够合理地被分配，达到效率的最大化。港口泊位分配问题（Berth Allocation Problem，以下简称 BAP）就是基于这个背景提出来的，解决 BAP，实际上就是为到达港口的船舶寻找到最佳的停靠位置与停靠时间。

以下是 BAP 的数学模型的约束：

1) 若按照合理安排，理论上每个到港船舶都可被服务一次
2) 任何一个到港的船舶若被服务则仅被服务一次，船舶到达之后才能被服务。
3) 每个泊位在同一时间只能服务一艘船舶，且不能移泊。
4) 船舶到港时间，服务时间占用泊位数事先知道

模型的符号说明如下：

I 到港船舶集合，$i \in I$
J 港口泊位集合，$j \in J$
K 工作时间集合，$k \in K$
N 被服务船集合，$n \in N$
I 到港船舶集合，$i \in I$
A_i，船舶 i 到港时间，$i \in I$
S_i，船舶 i 需被服务时间，$i \in I$
T_i，船舶 i 开始被服务时间，$i \in I$
X_1，在 K 内没有被服务到的船舶数量
X_2，在 K 内所有船等待时间之和

X_3，最后一艘被服务完的船的离开时间

模型的目标变量为：

$$f(x) = 100 * X_1 + 2 * X_2 + X_3$$

由上可推导出：

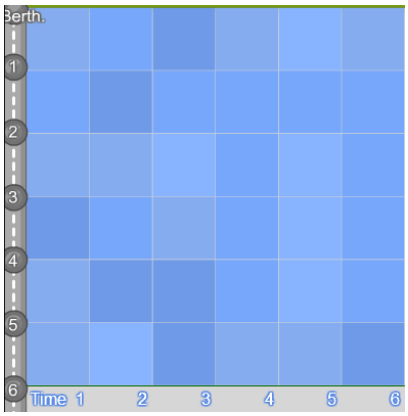
$$X_1 = |\mathrm{I}| - |\mathrm{N}|$$
$$X_2 = \sum_{n \in N} T_n - A_n,\ n \in N$$
$$X_3 = \max(T_n + S_n), n \in N$$

我们的任务就是在满足约束条件的情况下，使得$f(x)$尽可能地小。

2 贪心算法介绍

贪心法，又称贪心算法、贪婪算法、或称贪婪法，是一种在每一步选择中都采取在当前状态下最好或最优（即最有利）的选择，从而希望导致结果是最好或最优的算法。贪心算法在有最优子结构的问题中尤为有效。最优子结构的意思是局部最优解能决定全局最优解。简单地说，问题能够分解成子问题来解决，子问题的最优解能递推到最终问题的最优解。

在此问题中，以第一关为例子。总工作时间是 6，可用泊位为 6。有 5 艘船。每艘船的具体信息如图所示。



Vessel index	Arrival time	Service Time	Berths occupied	Icon
1	0	3	5	
2	1	1	2	
3	2	1	2	
4	2	1	5	
5	2	2	4	

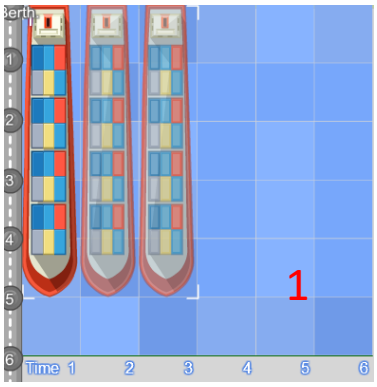
如果是使用基于先到先排，能上则上的贪心算法，是下面的情况。

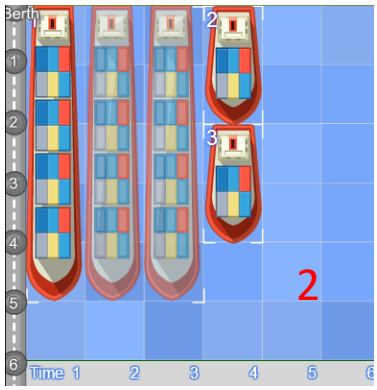
1.第一艘船 time0 到达，所以从 time0 开始排，占三个时间单位。

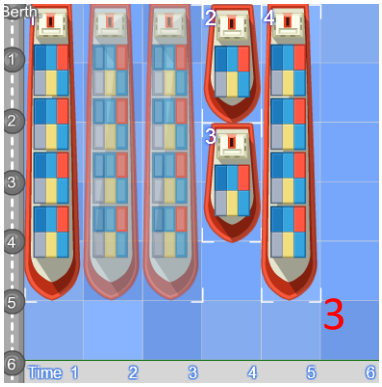
2.第二艘船 time1 到达，但是 time1 到 time3 之间都没有泊位了，所以只能从 time3 开始摆。第三艘也是，在 time2 到达，但是 time2 到 time3 之间都没有足够的泊位了，所以只能从 time3 开始摆。

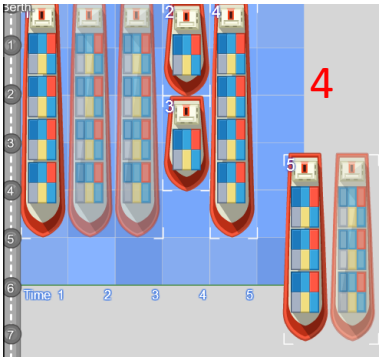
3.船 4 也是 time2 到达的，但是 time4 之前都没有泊位了，所以只能从 time4 开始摆。

4.这时候船 5 来了，可是纵观全局，已经没有容得下它的地方，不管怎样都摆不进了。



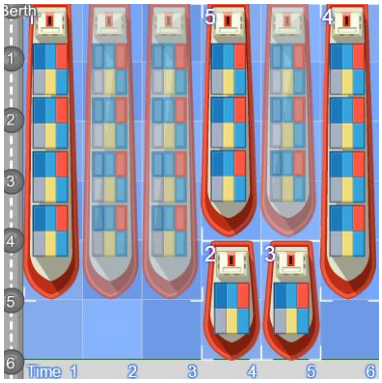






可以看出，贪心算法的优点是方式很简单，很容易实现。

但其实这个 case 船是可以全部安排好的，只要把排船顺序换成 1->5->2->3->4 再用上面的贪心就好了。



所以原始的贪心算法缺点也显而易见，就是太 naive 了。在这个简单的情况下原始的贪心得出的结果都不能把船全放进去，更别说现实中纷繁复杂得多的情况了。由此可见，原始的贪心算法是不能充分地利用好时间空间的。

那么是不是还有其他放置方法可以达到最优解呢呢？第一个想到的就是深搜了，在这个 case1 是可行的，毕竟数据量不是很大，但如果是往后动辄 20 艘船，用深搜来安排也是不切实际了。

那么又可以想，深搜最根本的无非就是把所有情况都走了一遍，那么就肯定存在一种安排顺序，根据那个安排顺序，用贪心也可以达到最优解。所以说到底，就是可以把所有排船方案都遍历一次，就能找到最优解。于是问题又来了，假设有 20 艘船，那么排船方案多达 20!种。把那么多方案都过一遍，显然也不切实际。

那还有办法吗？肯定是有的，就是利用启发式算法生成船的排列顺序，即可在合理时间内生成最优解或者不是最优但是也不错的答案。

3 启发式算法之模拟退火算法

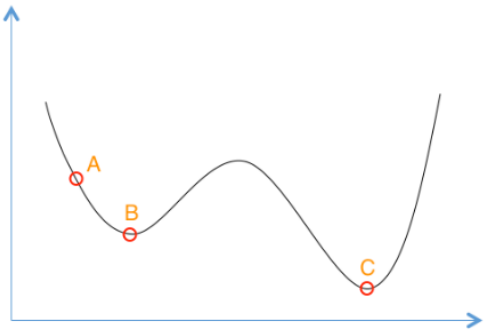
3.1 模拟退火算法描述

启发法源自古希腊语的 εὑρίσκω，是指依据有限的知识（或“不完整的信息”）在短时间内找到问题解决方案的一种技术。它是一种依据关于系统的有限认知和假说从而得到关于此系统的结论的分析行为。由此得到的解决方案有可能会偏离最佳方案。通过与最佳方案的对比，可以确保启发法的质量。

模拟退火的原理也和金属退火的原理近似：我们将热力学的理论套用到统计学上，将搜寻空间内每一点想像成空气内的分子；分子的能量，就是它本身的动能；而搜寻空间内的每一点，也像空气分子一样带有“能量”，以表示该点对命题的合适程度。算法先以搜寻空间内一个任意点作起始：每一步先选择一个“邻居”，然后再计算从现有位置到达“邻居”的概率。可以证明，模拟退火算法所得解依概率收敛到全局最优解。

模拟退火其实也是一种贪心算法，但是它在搜索过程中加入了随机的因素。这是为了用来在固定时间内寻求在一个大的搜寻空间内找到的最优解。

以下图为例，假设要找函数的最小值。从 A 点开始，如果采用传统的贪心算法（爬山算法），那么只能一直往下降的方向去，到 B 点就搜索停止了。但是模拟退火算法可以以一定概率接受比当前解差的解，这也就是说，用模拟算法搜索到 B 点时，是有可能接着往右边找从而越过 BC 之间的峰点到达 C 点，从而找到全局最优解的。



那么问题来了，和传统贪心相比，模拟退火在具体实现中，是怎么接受新解的呢？

$$p=\begin{cases}1 & if \quad E(x_{new})<E(x_{old}) \\ \exp(-\dfrac{E(x_{new})-E(x_{old})}{T}) & if \quad E(x_{new})\geq E(x_{old})\end{cases}$$

在温度为 T 时，出现能量差为 dE 的降温的概率为 P(dE)，表示为：P(dE) = exp(dE/(kT))。其中 k 是一个常数，exp 表示自然指数，且 dE<0。所以 P 和 T 正相关。这条公式就表示：温度越高，出现一次能量差为 dE 的降温的概率就越大；温度越低，则出现降温的概率就越小。又由于 dE 总是小于 0（因为退火的过程是温度逐渐下降的过程），因此 dE/kT < 0 ，所以 P(dE) 的函数取值范围是(0,1) 。随着温度 T 的降低，P(dE)会逐渐降低。

我们将一次向较差解的移动看做一次温度跳变过程，我们以概率 P(dE)来接受这样的移动。也就是说，在用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 *f*，温度 T 演化成控制参数 *t*，即得到解组合优化问题的模拟退火演算法：由初始解 *i* 和控制参数初值 *t* 开始，对当前解重复“产生新解→计算目标函数差→接受或丢弃”的迭代，并逐步衰减 *t* 值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值 *t* 及其衰减因子 Δ*t* 、每个 *t* 值时的迭代次数 L 和停止条件 S。

以下是模拟退火的代码基本思想：

(1) 初始化：初始温度 T(充分大)，初始解状态 S(是算法迭代的起点)，每个 T 值的迭代次数 L

(2) 对 k=1, …, L 做(3)至(6)：

(3) 产生新解 S′

(4) 计算增量 ΔT=C(S′)-C(S)，其中 C(S)为评价函数

(5) 若 ΔT<0 则接受 S′作为新的当前解，否则以概率 exp(-ΔT/T)接受 S′作为新的当前解.

(6) 如果满足终止条件则输出当前解作为最优解，结束程序。终止条件通常取为连续若干个新解都没有被接受时终止算法。

(7) T 逐渐减少，且 T->0，然后转(2)。

3.2 模拟退火的数学模型

模拟退火的数学模型由三部分构成：解空间、目标函数和初始解。

解空间:对所有可能解均为可行解的问题定义为可能解的集合,对存在不可行解的问题,或限定解空间为所有可行解的集合,或允许包含不可行解但在目标函数中用罚函数(Penalty Function)惩罚以致最终完全排除不可行解;

目标函数:对优化目标的量化描述,是解空间到某个数集的一个映射,通常表为若干优化目标的一个和式,应正确体现问题的整体优化要求且较易计算,当解空间包含不可行解时还应包括罚函数项。

初始解:是算法迭代的起点,试验表明,模拟退火算法是健壮的(Robust),即最终解的求得不十分依赖.初始解的选取,从而可任意选取一个初始解。

模拟退火算法新解的产生和接受可以分为如下四个步骤：

1) 第一步有一个产生函数从当前解产生一个位于解空间的新解
2) 计算与新解所对应的目标函数差
3) 判断新解是否被接受
4) 当新解被确定接受时，用新解代替当前解

3.3 模拟退火解决 BAP 的算法

本文针对 BAP，提出具体的退火模拟算法。模拟退火算法的初始解采用网站上贪心产生出来的解。停止准则设置为温度小于某个很小的参数β 。

目标函数——$f(x) = 100 * X_1 + 2 * X_2 + X_3$。

X_1为未被安排的船舶数量，X_2为全部船舶的等待时间之和，X_3为最后一艘被服务完的船的离开时间：

接受准则——若新解优于原解，则接受它；否则以一定的概率接收它，如下图公式所示，E_{new}为新产生的权值，E_{old}为原来的权值，T 为当前温度。

$$p = \begin{cases} 1 & if \quad E(x_{new}) < E(x_{old}) \\ \exp(-\dfrac{E(x_{new}) - E(x_{old})}{T}) & if \quad E(x_{new}) \geq E(x_{old}) \end{cases}$$

构造邻域——随机交换两艘船的安排顺序，再将新的安排顺序放回地图，计算权值。

冷却机制——对于不复杂的情况，$T \leftarrow 0.999 * T$；对于复杂的情况，$T \leftarrow 0.9999 * T$.

模拟退火部分的伪代码如下：

```
simulateAnneal()
    choose an initial greedy solution X0
    give an initial temperature T0, X←X0,T←T0,X' ←X0
        whileT>β
            for i← 1 to 5 do
                pick a solution X'∈N(X) randomly
```

generate f(X') with greedy algorithm

Δ f ← f(X') – f(X)

if (Δ f < 0) X←X '

else X←X ' with probability $P(\Delta f,T)$

if f(X) > f(X*) then X*←X

T ← 0.999∗ T

4 实验结果

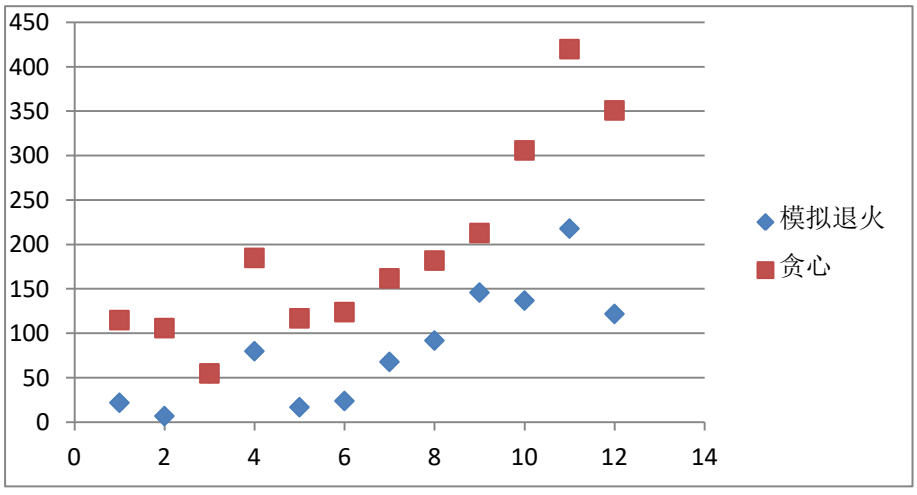
以下是模拟退火算法产生的结果，初始温度为 1，Game7,10,11,12 的降温系数为 0.9999，其余为 0.99，停止温度为 1e-50。

	Unassigned vessels	Total waiting times	Last departure time	f(x)=w1x1+w2x2+w3x3
Game 1	0	8	6	22
Game 2	0	0	7	7
Game 3	0	19	17	55
Game 4	0	30	20	80
Game 5	0	4	9	17
Game 6	0	7	10	24
Game 7	0	29	10	68
Game 8	0	38	16	92
Game 9	0	63	20	146
Game 10	0	59	19	137
Game 11	0	97	24	218
Game 12	0	46	30	122

以下是贪心产生的结果

	Unassigned vessels	Total waiting times	Last departure time	f(x)=w1x1+w2x2+w3x3
Game 1	1	5	5	115
Game 2	1	0	6	106
Game 3	0	19	17	55
Game 4	1	33	19	185
Game 5	1	4	9	117
Game 6	1	7	10	124
Game 7	1	26	10	162
Game 8	1	33	16	182
Game 9	1	47	19	213
Game 10	1	93	20	306
Game 11	2	98	24	420
Game 12	2	61	29	351

两个算法的 f(x)的图标对比，很直观地可以看出不管从安排的角度还是等待的角度，模拟退火都有更优更稳定的表现。而且当情况变得复杂的时候，虽然模拟退火耗时多一点，但是得到的结果是传统的贪心算法无法比拟的。



5 实验总结与感想

通过这次的实验，我进一步理解了模拟退火算法并体会到了启发式算法的优势所在。在之前的总结中我没有特别说明但是在实验过程中明显感觉到的就是冷却系数对模拟退火的影响也是挺大的，当冷却系数趋近于一时，迭代次数增多，耗时会比较久但是找到的答案往往会更好。但是很可惜由于这次实验开始太晚没有足够的时间去用具体数据反映这一关系。

6 附录

C++代码，测试数据，实验结果。