算法设计实验报告

(2017 学年春季学期)

课程名称：Operating System　　任课教师：**张子臻**　　**批改人(此处为 TA 填写)：**

年级+班级	1507	专业（方向）	**软件工程（移动信息工程）**
学号	15352155	姓名	**赖贤城**
电话	13727024851	Email	754578682@qq.com
开始日期	2017/6/1	完成日期	2017/6/4

1. 实验目的

1. 将算法的知识运用于实际问题当中
2. 对 NP-hard 组合优化问题学会使用启发式算法求得较优解

2. 实验过程

（一）实验思路

a) 从贪心的方法得到启发

贪心算法是按照到达时间的顺序进行安排，对于每一艘船的安排，尽量使其等待时间最少。也就是说，贪心的解就是船按照到达时间顺序产生的一个排列所得到的一个解。那么，船的每个排列组合都有其对应的解。由此得到启发，启发式算法就是为组合优化问题而生的，既然本题可以转化为组合优化问题，那么就能使用启发式算法对其进行优化。

b) 选择一种启发式算法

较常用的启发式算法包括禁忌搜索，模拟退火，遗传算法。我认为遗传算法能更好的利用之前得到的结果，而且在解的变化，即跳出局部最优的能力上来看遗传算法也不亚于前两者。因此使用遗传算法作为做贪心结果的优化算法。

c) 遗传算法中的步骤以及其策略

1) 初始种群的生成

使用贪心算法对应的排列作为初始种群，也就是初始种群的全部个体都是贪心的结果。这样做的原因是使得遗传算法能有个好的方向，而不是全局的盲目的搜索，全局随机的初始种群会使得遗传算法耗费太多时间在找大方向上。

2) 淘汰策略

产生完子代之后，由于种群数量是原来的两倍，所以应该要淘汰掉一半的个体。采用**轮盘赌**的方式进行淘汰能同时兼顾到 ①尽可能保留优良个体 ②让一些较差的个体有一定机会存活下来，因为差的个体也可能产生好的子代或者产生好的变异，因此差的个体也要保存下来一些，才能最大程度的保证进化方向的多样化。

3) 交叉策略

由于在这个题目中，染色体是对一串编号不重复的船的排列，因此普通的交叉方法不能保证交叉出的子代中没有重复的基因。因此这里采用针对全排列的交叉算法，Order Crossover 算法，这种方法既适合于排列问题，又能很好地保留父代的优良性状，也就是基因的相对位置被较好的保留了。例如以下的例子

父代 A: 872 | 139 | 0546

父代 B: 983 | 567 | 1420

交叉后：

子代 A: 872 | 931 | 0546

子代 B: 983 | 756 | 1420

4) 变异方法

由于不同的变异方法适用的场景不同，同时为了实验不同的变异方法跳出局部最优的能力，因此我总共写了 4 种具体的变异方法，以及将这四种方法混合使用的混合法。

i 交换基因法，随机找出染色体中的两个不同的基因，进行交换

i 翻转染色体片段法，随机取两个点，将这两个点之间的染色体片段进行翻转，从而使得染色体片段内的基因顺序改变

ii 交换染色体片段法，随机选取一个点，将这个点左右两边的染色体片段交换，也就是左边片段变为右边片段，右边片段变为左边片段

iii 基因插入法，随机选取一个基因将它从原位置删除，然后再随机找一个位置将这个基因插入

iv 混合法，变异的时候随机选择上述的任一方法进行变异

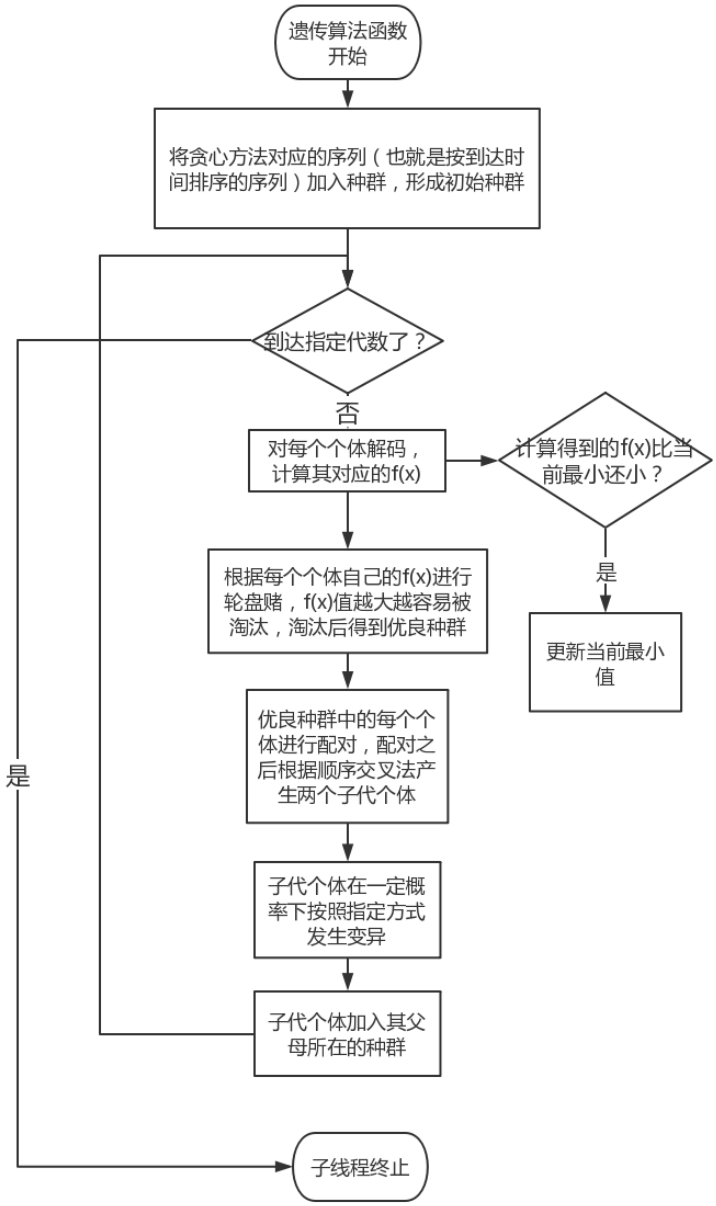
5) 解码过程

最开始说了，我的启发式算法的编码方式是从贪心算法得到的启发。每一个编码好的序列都对应着一种方案（每艘船的排放位置）。从编码序列得到方案就是解码。很简单，其实就是使用贪心的算法，对一个序列按从左到右的顺序依次摆放，每一次摆放就是一次贪心，这样贪心完之后就得到了可以输入到网站的解。

6) 对遗传算法的优化

为了使得搜索的范围更发散，采用较大的变异率进行变异。但是有时发散的代数多了就会变得很难收敛。因此我采用一种粗暴的方法：过一定的代数（可设置的一个参数）之后，则抛弃当前的种群，将种群的全部个体都设置为当前找到的最好个体，然后从这个结果再开始进行繁殖遗传。

（二）流程图



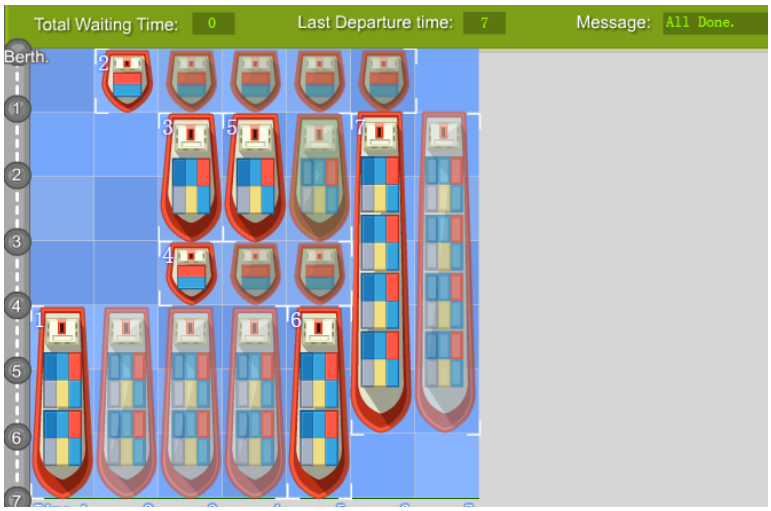
3. 实验结果

第一个游戏：

数据量很小，种群大小 5，迭代 3 代的情况下就得到了 22 的解

第二个游戏：

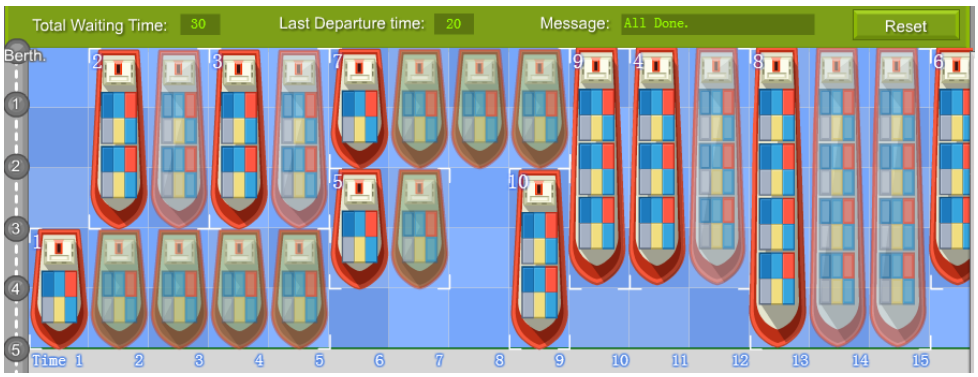
同样是很快得到最优解 7



第三到第六个游戏同样很快得出答案，应该是数据的设计问题，导致很容易收敛到全局最优解

第三个游戏

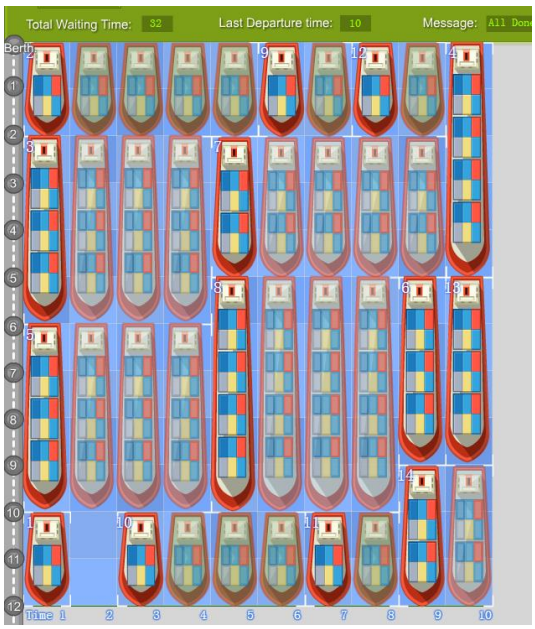
第四个游戏：



第五个游戏和第六个游戏图略

分析第 7 个游戏和第八个游戏：这两个样例跑的时候能明显感觉到结果优化到 100 多的时候就很难再继续优化，因此对于这两个样例我修改了遗传参数，将变异率提高，变异方法也从交换基因法改成混合法，目的是使其能尽可能的跳出局部最优的解，找到更优解。改完参数之后，第七个游戏没能得到全局最优解，第八个游戏能得到最优解 92

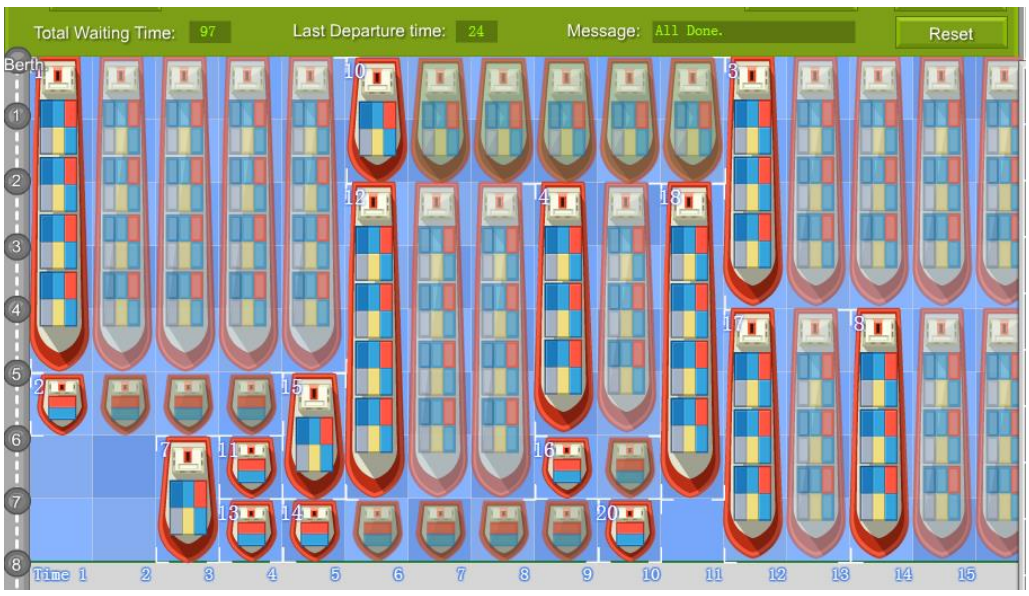
第七个游戏结果：



第八个游戏结果：

第九个游戏到第十二个游戏采用同样的变异率 50%，迭代代数 110 代，每隔 20 代重置一次种群，将其设置为当前最优解。最终得到的结果，第九第十第十一能得到全局最优解，最后一个只得到较优解 192，结果如下（第九第十略）：

第十一个游戏：



第十二个游戏：

4. 遗传算法和贪心算法的比较

1. 用时上的对比

   贪心算法是确定的算法，其用时只与船的数量和泊位的大小有关系，在这十二个游戏里面，贪心的用时都很短（<1s）。而遗传算法属于不确定算法，并且对于不同的问题需要设置不同的参数，例如迭代的代数，变异率的大小。一旦数据量较大，通常遗传的时间也要相应的增大才能找到较优解。因此在用时上，贪心是远优于遗传的

2. 结果上的比较

	Game 1	Game 2	Game 3	Game 4	Game 5	Game 6	Game 7	Game 8	Game 9	Game 10	Game 11	Game 12
遗传	22	7	55	80	17	24	74	92	146	137	218	192
贪心	115	106	55	185	117	124	162	182	213	306	420	351

   可见遗传算法比贪心算法得出的解优秀很多，大部分还得到了全局最优，说明付出的时间得到了回报。不过不像枚举那样的全局搜索需要大量的甚至是无法承受的时间，遗传算法是能在较短的时间得到较优解（甚至最优解）的。

5. 实验感想

这次实验算是对自己的一个挑战吧，也或许可以说是实现了自己的小小梦想。之前数学建模的时候看了很多优化算法，深深为之着迷。但是实现过的就只有模拟退火一个。一直以来很想将遗传算法实现出来，应用到实际的问题中。没想到真的等到了机会。

说实话，比起模拟退火，遗传算法的实现来的复杂得多。又有轮盘赌，又有交叉，又有各种不同的变异方式的。前前后后花了一天半的时间才将它实现了出来。期间又做了多次改良，例如写了四种变异方法（不算混合法的话），隔一定的代数对种群进行干预使得其不会发散太开等等。虽然感觉自己其实还不是很会设置遗传的那些参数，但是在这过程中也算是对遗传算法有了理性和感性的全方面认识。同时，这两天的编程中，在实现这些复杂的操作的时候也进一步提高了自己的代码能力，对 C++的掌握程度也是感觉进步很多。