
[单机调度-localsearch]

谭超 (学号: 1120161874)

摘要: 通过 localsearch 解决单机调度问题

关键词: 局部搜索

1 引言

问题: N 个元件要在一台机器上加工, 已知这 N 个元件的释放时间、加工时间、交货时间, 要求总延迟时间最小的调度方案。(延迟时间=实际完成时间-交货时间)

本实验主要是利用局部搜索解决单机调度问题, 达到延迟时间总和最小的目的

2 算法设计

本算法的完整过程分为三步, 第一步生成一个随机序列, 第二步由生成的随机序列达到局部最优, 第三步由局部最优的结果推出全局最优的结果(这里不一定是最优的, 严格来讲是较优的)。细节描述为: 生成一个 0 到 $n-1$ 的不重复随机序列, 表示 0 到 $n-1$ 号工件的加工顺序, 定义一个未转化时的延迟时间 $last_mintime$ 和这一轮更新完的延迟时间 $mintime$, 将这二者是否相等作为状态转移结束的标志即当 $last_mintime == mintime$ 时说明此随机序列达到了局部最优, 跳出循环, 否则将最后一个工件插入第 i ($0 \leq i < n-1$) 号工件的前面, 如果转化后的加工顺序延迟时间小于原本的延迟时间, 则更新。我们设计跳出局部最优 10000 次, 最后选择这些结果中最小的作为全局最优解。

3 实验

3.1 实验设置

开发软件: dev c++

开发设备 联想小新 700

跳出局部最优次数: 10000 次

3.2 实验结果

运行结果截图

```

1 0 3 4 2
initial_mintime=332
case 9993 end
case 9994 start
0 1 2 4 3
initial_mintime=99
case 9994 end
case 9995 start
1 4 2 3 0
initial_mintime=381
case 9995 end
case 9996 start
3 4 1 0 2
initial_mintime=483
case 9996 end
case 9997 start
3 0 4 2 1
initial_mintime=329
case 9997 end
case 9998 start
0 4 2 3 1
initial_mintime=221
case 9998 end
case 9999 start
1 4 2 3 0
initial_mintime=381
case 9999 end
局部搜索完成!
58
请按任意键继续. . .

1 3 4 0 9 8 2 5 7 6
initial_mintime=739
case 9993 end
case 9994 start
4 9 5 0 2 3 1 7 8 6
initial_mintime=1254
case 9994 end
case 9995 start
9 7 3 0 1 5 8 2 4 6
initial_mintime=1446
case 9995 end
case 9996 start
6 4 9 2 5 3 8 7 0 1
initial_mintime=1377
case 9996 end
case 9997 start
1 9 4 6 0 5 3 7 2 8
initial_mintime=1254
case 9997 end
case 9998 start
4 9 5 7 6 3 8 0 2 1
initial_mintime=1509
case 9998 end
case 9999 start
1 2 8 6 0 9 5 7 3 4
initial_mintime=726
case 9999 end
局部搜索完成!
367
请按任意键继续. . .

```

```

C:\Users\ic\Desktop\study\练习\项目9.exe
7 5 4 14 2 0 13 12 1 11 10 8 9 6 3
initial_mintime=5994
case 9993 end
case 9994 start
2 8 10 9 12 13 4 14 7 0 6 3 5 1 11
initial_mintime=4893
case 9994 end
case 9995 start
11 4 3 10 0 12 13 6 1 5 8 14 7 9 2
initial_mintime=5249
case 9995 end
case 9996 start
6 0 12 2 8 14 3 7 4 1 13 11 5 10 9
initial_mintime=4360
case 9996 end
case 9997 start
1 8 9 6 5 0 7 14 13 10 4 12 11 3 2
initial_mintime=5456
case 9997 end
case 9998 start
4 6 3 1 5 14 11 10 9 8 0 2 13 12 7
initial_mintime=5074
case 9998 end
case 9999 start
12 9 8 1 11 0 5 13 14 6 2 7 3 10 4
initial_mintime=5492
case 9999 end
局部搜索完成!
3053
请按任意键继续. . .

8 14 11 17 15 7 9 2 18 4 13 5 12 6 0 3 19 10 1 16
initial_mintime=8282
case 9993 end
case 9994 start
6 8 9 12 19 13 4 3 11 18 17 10 15 7 14 1 2 16 5 0
initial_mintime=8701
case 9994 end
case 9995 start
9 18 12 0 14 6 5 10 19 15 13 17 3 1 8 16 11 2 7 4
initial_mintime=9948
case 9995 end
case 9996 start
16 5 7 4 9 2 12 11 0 3 17 14 8 6 1 19 15 18 13 10
initial_mintime=10211
case 9996 end
case 9997 start
4 7 5 13 0 17 8 1 11 9 10 6 15 16 18 12 2 3 19 14
initial_mintime=9042
case 9997 end
case 9998 start
12 6 15 19 14 8 13 11 9 3 10 16 5 2 1 0 18 17 7 4
initial_mintime=8470
case 9998 end
case 9999 start
17 2 15 6 1 11 8 5 12 14 4 10 0 13 9 7 16 19 3 18
initial_mintime=8678
case 9999 end
局部搜索完成!
5724
请按任意键继续. . .

```

```

18 17 13 5 12 24 1 14 21 15 10 22 7 0 9 20 6 4 19 2 8 23 11 16 3
initial_mintime=15611
case 9993 end
case 9994 start
5 2 22 14 15 9 11 0 21 13 19 10 24 17 12 23 20 3 6 4 8 16 1 18 7
initial_mintime=15021
case 9994 end
case 9995 start
4 6 14 24 10 8 2 12 3 11 7 5 0 13 23 1 21 20 16 18 22 9 17 19 15
initial_mintime=13505
case 9995 end
case 9996 start
8 4 17 6 19 14 21 5 18 12 13 0 3 2 9 11 23 7 16 1 20 10 22 15 24
initial_mintime=14035
case 9996 end
case 9997 start
14 20 15 19 8 3 9 22 5 1 10 6 18 24 13 4 21 17 11 23 12 0 2 16 7
initial_mintime=15249
case 9997 end
case 9998 start
9 12 4 18 13 19 8 23 6 22 21 10 5 2 11 20 15 24 0 3 16 14 1 17 7
initial_mintime=13410
case 9998 end
case 9999 start
15 0 8 20 14 4 9 5 17 24 1 3 6 13 11 22 16 18 2 19 21 10 7 23 12
initial_mintime=15955
case 9999 end
局部搜索完成!
9711
请按任意键继续. . .

```

实验结果:

用例	运行结果	运行时间
1	58	31.2s
2	367	32.2s
3	3053	37.6s
4	5724	45.0s
5	9711	49.5s

实验结果分析：从实验结果来看，随着工件数量的增加，寻找最优调度花费的时间也在增加，这是由于工件数量的增加造成每一个加工序列的领域扩大，那么转移到领域去所需要的时间也随之增加。例如当有 5 个工件时，每个序列的领域有 5 个可行的序列，当加工序列增加到十个，每个序列的领域也可行的序列也增加到 10 个。另外一个较为显著的结论是工件数量越少，得到的结果越接近最优结果，这是因为我设定的跳出局部最优的次数为固定的 10000 次。

4 总结

我在本实验中设置的跳出局部最优次数为固定的 10000 次，因而对于工件数不同的测试用例，实验结果与最优的调度之间的差距也不同，例如测试用例的前两个实验结果和最优调度几乎一致，当工件数增大时，结果与最优调度之间的差距就慢慢变大了。可以设置跳出局部最优的次数和由元件个数决定。但这样当工件数增加时，花费的时间就会成倍的增长。