[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=实际完成实际-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，第二部由生成的随机序列达到局部最优，第三部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==miintime 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<=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。

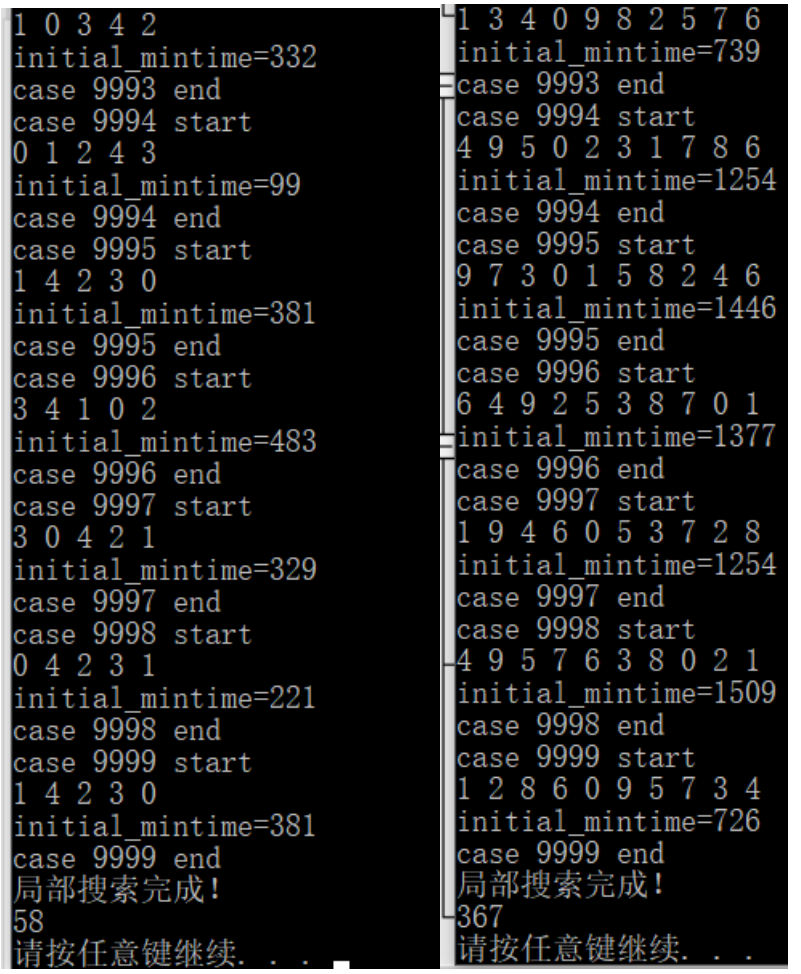
3　实验

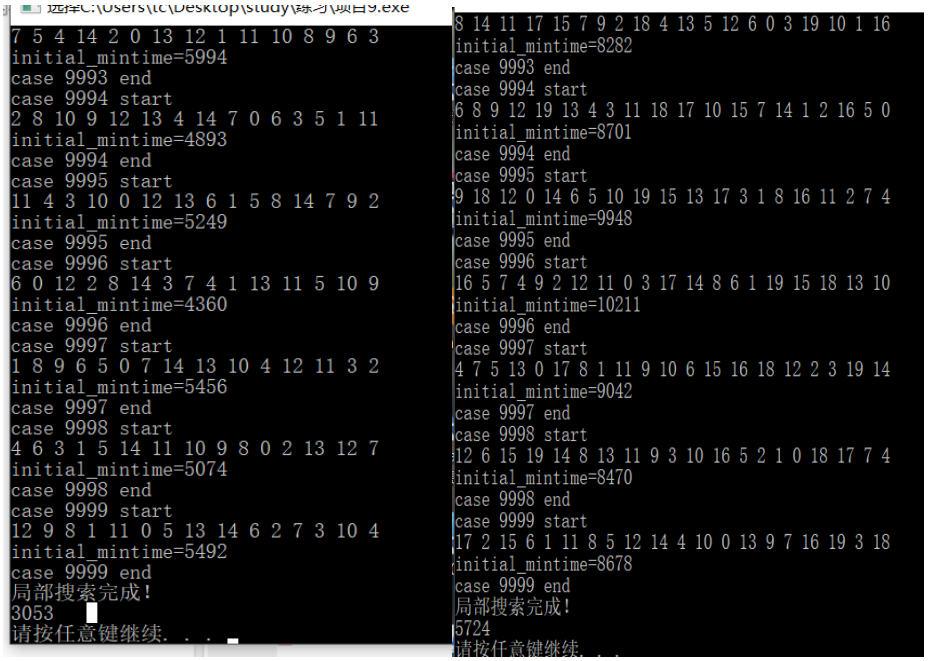
3.1 实验设置

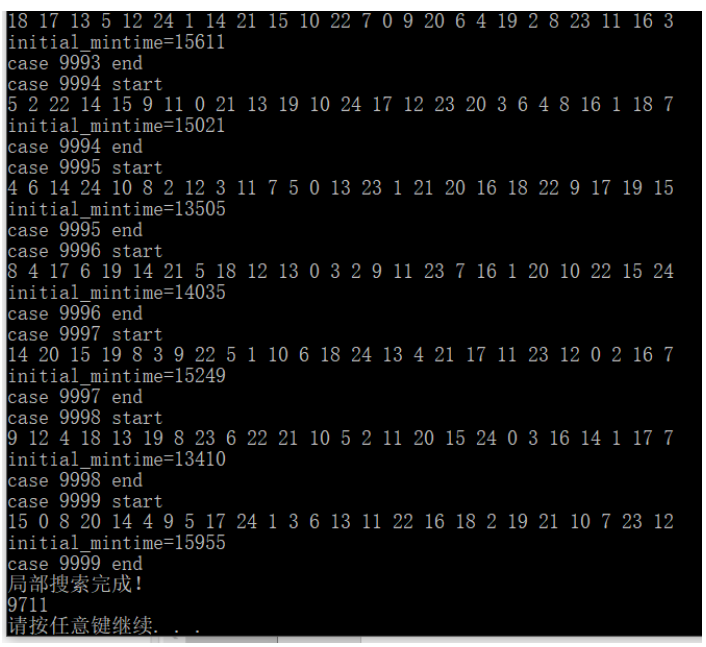
开发软件：dev c++
开发设备 联想小新 700
跳出局部最优次数：10000 次

3.2 实验结果

运行结果截图







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。