昵称：victorchew
园龄：1年10个月
粉丝：1
关注：0
+加关注

< 2018年5月 >

日	一	二	三	四	五	六
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

随笔档案
2016年7月 (13)

阅读排行榜
1. 蚁群算法简介(part3: 蚁群算法之更新信息素)(2313)
2. 遗传算法在JobShop中的应用研究（part1: 绪论）(499)
3. 遗传算法在JobShop中的应用研究(part 2：编码)(294)
4. 蚁群算法简介(part2: 蚁群算法之构造路径)(278)
5. 蚁群算法简介(part 1：蚁群算法之绪论)(271)

遗传算法在JobShop中的应用研究（part 5：解码）

解码操作是整个遗传算法最重要的一步，在这步里面我们利用配置文件中的信息将染色体解码成一个有向无环图。

在介绍解码操作之前我们先来看一下配置文件，在part1绪论中我们已经介绍了一个车间调度问题的基本信息可以用一个表格来表示：

	$\theta_{1,1}$	$\theta_{1,2}$	$\theta_{2,1}$	$\theta_{2,2}$
$r_{i,j}$	1	2	2	1
$D_{i,j}$	3	2	5	1

表5.1

在编程时我们要把这个表格用一个配置文件来存储，这个配置文件可以是一个记事本文件，后缀名是.txt。

下面我们给出上面那个表格对应的配置文件：

Init.txt

```
2 2
2 1 3 2 2
2 2 5 1 1
```

可以看到这个配置文件有三行，第一行中的第一个“2”表示有2个工件，第二个“2”表示有2台机器。第二行代表工件1的信息，有五个数字，它们的含义如下：

“2”表示工件1有2道工序，

“1”表示工件1的第一道工序需要在机器1上加工，

“3”表示工件1的第一道工序在机器"1"上加工的时间为3个时间单位

“2”表示工件1的第二道工序需要在机器2上加工

“2”表示工件1的第二道工序在机器2上的加工时间为2个时间单位

第三行表示工件2的全部信息，大家可以参考第二行把第三行的数字各自的含义写出来，在此不再赘述。

在进行解码之前，我们首先要写一个函数把配置文件的信息读到一个python类中，最终输出的类实例I应该有着如下的成员：

jobs=[[(1,3),(2,2)],[(2,5),(1,1)]]

（list中又嵌套两个子list,每个子list对应一个工件的信息，子list中的一个有序偶对应表格中的一列）

n=2(工件的个数)

m=3（机器的个数）

这个python类的代码如下所示：

```
⊞ View Code
```

我们还需要写一个函数来把配置文件txt中的信息读到python中对应的成员变量中即

I = LoadInstance("配置文件.txt");

loadInstance的python代码如下：

```
def LoadInstance(fname):
    """Load instance file."""
    f = open(fname, 'r')"""打开配置文件 Init.txt"""
    head = f.readline().split()"""读入第一行"""
    n, m = int(head[0]), int(head[1])"""将工件数和机器数存在n和m两个变量中"""
    I = []""" 构造上文中的"jobs列表""""
    for l in f:"""从文件中依次读入每行数据，一行对应一个工件"""
        l = l.split()
        if len(l) < 3: continue """如果少于3个数则跳过该行"""
        ntasks = int(l[0])"""ntasks存放该工件包含的工序道数，Init每行对应一个工件"""""
        I.append([])"""每行一个子list代表一个工件的信息"""
        for j in xrange(ntasks):
            mid = int(l[j*2+1])
            dur = int(l[j*2+2])
            I[-1].append((mid, dur))"""每个工件有多道工序，每个工序对应一个有序偶（机器数，加工时间）"""
    return Instance(I, m)"""返回Instance类的一个实例"""
```

我们在part1绪论部分已经介绍了一个JobShop问题的一个解可以用一个有向无环图来表示，而解的质量即makespan则是这个有向图中从起点S到终点F中所有路径中总完成时间最长的那条路径所对应的完成时间。

通过一条染色体和表格5.1来构造相应的有向无环图的代码如下：

```
def ComputeDAG(s, I):
    """s是染色体即工件号的乱序排列，I存储表5.1的信息，格式在上面刚刚介绍过"""
    G = [] """构造一个空list来存储图"""
    for t in s: G.append([])"""为每道工序在图中用一个节点表示，即每个节点对应一个子list"""
    G.append([])"""添加终止节点"""
    T = [0 for j in xrange(I.n)]"""初始化一个长度为工件数的全0list"""
    last_task_job = [-1 for j in xrange(I.n)]"""构造存储每个工件上一次加入图的工序号，初始化为-1 """"
    tasks_resource = [[-1 for j in xrange(I.n)] for m in xrange(I.m)]"""见后面正文部分"""
    st = [] "记录每个节点对应的工序在其工件内是第几道工序"
    for i in xrange(len(s)):""" 遍历染色体，染色体的下标即为图节点的编号，依次处理图的节点"""
        j = s[i]"""提取编号为i的节点对应的工件号"""
        t = T[j]""" 辅助变量,记录编号i的节点是j工件内的第几道工序"""
        st.append(t)""" 将t存入st"""
        r = I[j][t][0]""" 将j工件的第t道工序对应有序偶中的第一个数字，即该工序的机器号存入变量r"""
        # 判断j的相对工序号的计数器t的值是否等于j的工序数，若相等则将j的最后一道工序指向终止节点
        if t + 1 == len(I[j]): G[-1].append(i)
        # 若不是j的第一道工序则将该工序的前一道工序指向代表当前工序的节点i
        if t > 0: G[i].append(last_task_job[j])
        # 如果之前加入的节点也在使用和j的第t道工序相同的机器r,则将这些节点编号指向当前处理节点i
        G[i].extend([tasks_resource[r][j2] for j2 in xrange(I.n)
                     if j2 != j and tasks_resource[r][j2] > -1])
        T[j] = T[j] + 1"""j工件的工序号累加器加1"""
        last_task_job[j] = i"""将j工件的上一道工序的节点编号置为当前节点的编号i """
        tasks_resource[r][j] = i""" 将j工件中正在使用机器r的工序对应的节点编号置为i"""
    return G, st""" 返回图G,和记录每个节点对应的工序在其工件内是第几道工序的 list st"""
```

特别说明一下第9行task_resource这个list,以表5.1为例它的形式是：

task_resource=[[-1，-1]，[-1,-1]]

因为有两台机器，所以task_resource里面有两个子list,又因为有两个工件，所以每个子list里面有两个数字，初始化为-1,以后这个数字存放上一次使用这台机器的工件中的相对工序号。

在构造完有向图之后我们就可以利用这个有向图来计算每个解的适应度，具体算法是动态规划的算法。

```
1 def ComputeStartTimes(s, I):
2     """该函数计算图中每个节点（一个节点对应一道工序）的开始时间，最后一个节点存放总的makespan"""
5     G, st = ComputeDAG(s, I)
6     C = [0 for t in G]"C用来存放图G中每个节点的开始时间，长度为G的节点个数，初始化为全0"
7     for i in xrange(len(G)):"""遍历G中每一个节点"""
8         if len(G[i]) == 0: C[i] = 0""" 如果节点i的前面没有节点指向它，则i的开始时间为0"""
9         """否则，节点i的开始时间为所有前面指向它的节点的开始时间+处理时间中最大的那个节点的开始时间
+处理时间"""
          else: C[i] = max(C[k] + I[s[k]][st[k]][1] for k in G[i])

10     return C"""返回每个节点的开始时间，最后一个终止节点的开始时间即为makespan"""
```


victorchew
关注 - 0
粉丝 - 1
+加关注

« 上一篇：遗传算法在JobShop中的应用研究（part4:变异）
» 下一篇：遗传算法在JobShop中的应用研究（part 6：结果显示）

posted @ 2016-07-23 10:04 victorchew 阅读(160) 评论(0) 编辑 收藏