闲谈 JSP

汤力遥

北京理工大学，计算机科学与技术，07111403 班 1120141832

摘要：

简要探讨了对传统标准 JSP 问题的三种不同处理方式，分别为 启发式算法，元启发式算法，以及超启发式算法。其中，元启发式算法和超启发式算法基于经典 PSO 算法。

关键词：JSP, PSO, hyper-heuristic

问题简介：

生产调度问题(JSP)属于典型的组合优化问题，可化归到 TSP 问题，属于 NP-hard 问题。

JSP 问题有许多变形，在此仅讨论标准 JSP 问题，定义如下：

n 个工件: J_i (i = 1, . . . , n)

m 个机器: M_j (j = 1,...,m)

第 i 个工件的工序: O_{i1}，O_{i2},...，O_{im}

包含如下假设：

1. 机器同时只能加工一个工件，且运行过程中无故障
2. 工件各工序加工严格按顺序执行且仅执行一次
3. 所有工件 0 时刻同时到达，工件在机器间转移时间忽略

目标函数：

Min Max($\sum_{i=1}^{n} C_i$)，C_i 为第 i 个工件完成所有工序的时间（从 0 时刻开始计时）

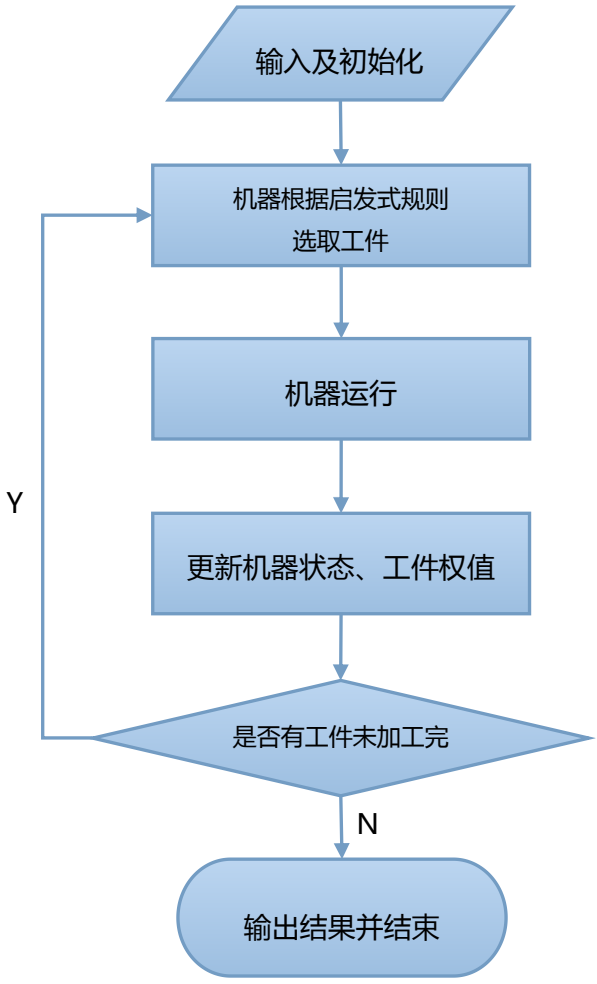
解决方案：

（一）：启发式规则

1、代码文件：启发式规则.cpp

2、基本框架：WSJF (Weighted shortest job first)

3、流程图：



4、简介及特点：

在传统的 WSJF 权值公式中加入机器的因素，在工序所需时间相差不大时，选择下一道工序所需机器上候选工件数较少的工件执行当前工序。

5、性能：（测试结果）

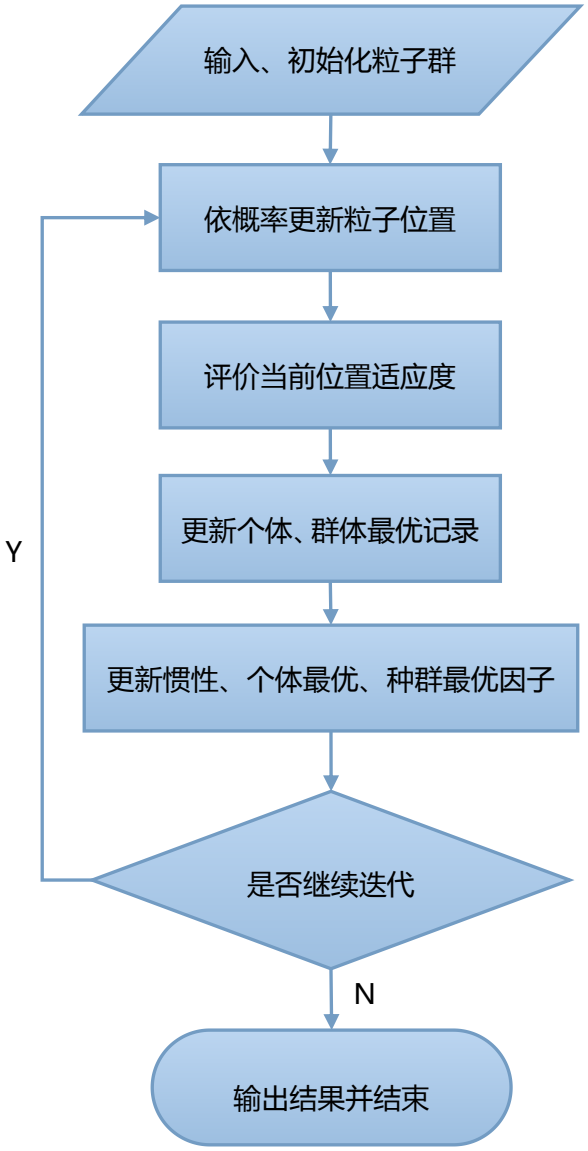
性能上与传统 WSJF 及 SJF 几乎相同，并没有显著的提升。与所有启发式算法一样，在处理较大规模的 JSP 问题时难以获得较好的解。并且，在测试过程中发现，解的质量在很大程度上依赖于原问题的结构，算法不具有广泛应用的潜力。

（二）：元启发式算法

1、代码文件：元启发式算法.cpp

2、基本框架：PSO (Particle swarm optimization)

3、流程图：



4、简介及特点：

假设有 n 个工件 m 台机器

编码：没有采用传统的 PSO 编码，而是采用基于工序的编码(operation-based representation)，即每个粒子的位置表示为一个 n*m 维的列表，列表中的元素为工件编号，每个列表可唯一对应一个有序的操作列表，一定程度上脱离了传统 PSO 中粒子的意义。编码保证任意一种编码都对应一个可行解，但仅具有半 Lamarckian 性。

例如：粒子[2 1 3 1 2 2 3 1 3] -> 操作列表[$O_{21}\ O_{11}\ O_{31}\ O_{12}\ O_{22}\ O_{23}\ O_{32}\ O_{13}\ O_{33}$]

其中，O_{ij} 表示第 i 个工件第 j 个操作(工序)

进化方程：上述编码方式无法采用传统的速度位置更新公式，但却能很好的契合如下离散的更新公式：$X_{i(t+1)} = c_2 \otimes g_{gBi}(c_1 \otimes g_{pBi}(\omega \otimes f_{a,b}(X_{it})))$

其中：$\omega, c_1, c_2 \in (0,1)$，$X_{it}$ 为粒子 i 在 t 时刻的位置

$$\omega \otimes f(X) = \begin{cases} f(X), r < \omega \\ X, r \geq \omega \end{cases}$$

其中：r 为(0,1)之间的随机数

$f_{a,b}(X)$ 为令 X 编码中 a,b 位之间的编码分别与该粒子中全部编码随机互换

其中：a,b 为(1, n*m)之间的随机数

$g_Y(X)$ 为按以下算法对 X, Y 中编码操作：

① 将 n 个工件随机分为两个集合 S_1，S_2

② 初始化 i = 1, j = 1, Z[n*m]

③ 若 X[i]$\in S_1$, Z[j] = X[i], j++;

若 Y[i]$\in S_2$, Z[j] = Y[i], j++;

i++;

④ 若 j$\leq$n*m 转③，否则，X = Z，结束算法

故，直观的，该进化公式使粒子 X 以 ω 的概率发生突变，以 c_1 的概率受个体最优记录(pB)影响，以 c_2 的概率受种群最优记录(gB)影响

惯性、个体最优、种群最优影响因子：分别表示为 ω ，c_1 ，c_2（即为进化公式中对应变量）使用更新公式使 ω 随着迭代次数的增加而减小，c_1，c_2 则随着迭代次数的增加而增大。相较传统的线性更新公式，算法中使用了如下更新公式：

$$\omega = 0.9 - 0.4 * \frac{i^2}{iteration^2} \quad , \quad c_1 = 0.1 + 0.5 * \frac{i^2}{iteration^2} \quad , \quad c_2 = 0.1 + 0.5 * \frac{i^2}{iteration^2}$$

其中，i 为当前迭代次数，iteration 为总迭代次数

通过引入平方项，使得粒子在前半段受个体最优、种群最优记录影响的概率维持在一个比较低的水平(<0.25)，同时突变的概率维持在比较高的水平(>0.75)，以充分探索解空间，而在算法迭代后半段由于平方项的存在可以使突变的概率较快的下降，同时个体、种群最优记录的影响概率较快的上升，使算法最终收敛。

迭代次数：表示为 iteration，根据问题的规模动态决定，具体的，算法中使用：

iteration = n*m*100

使算法能根据问题的规模相应的扩大搜索时间和范围。

5、性能：（测试结果）

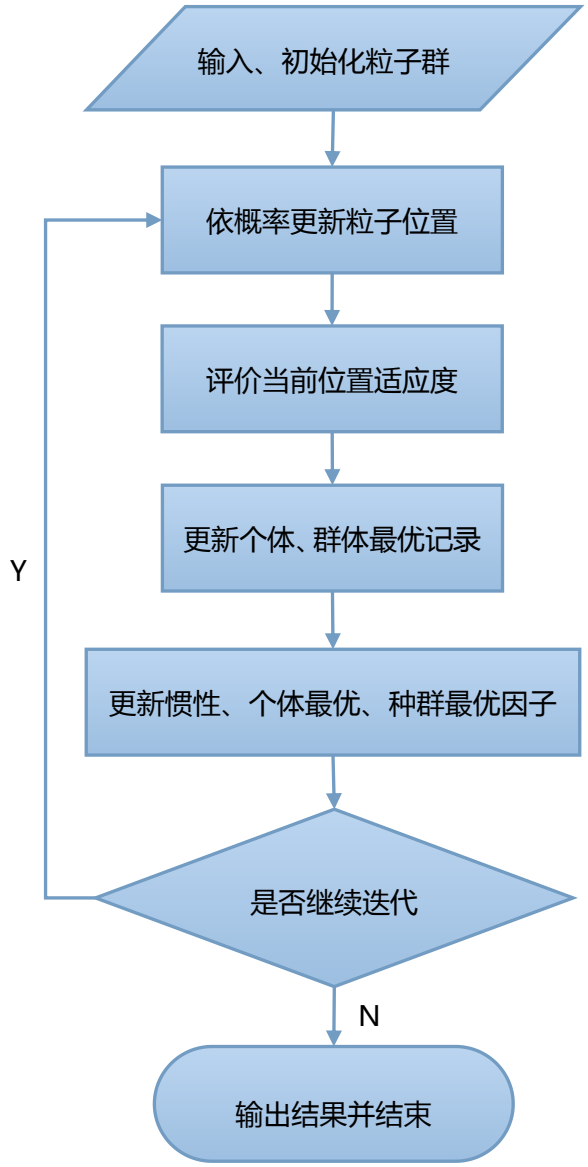
由于以上的优化，使得改进后的 PSO 能够一定程度上克服传统 PSO 收敛过快的缺点，在保留，充分的搜索解空间，并在小规模及大规模 JSP 问题中都取得较好的结果，但相对应的，随着问题规模的扩大，算法运行时间也随之增长。总体来说，算法更具有普适性，可以应用在不同结构的 JSP 问题上。

（三）：超启发式算法

1、代码文件：超启发式算法.cpp

2、基本框架：PSO (Particle swarm optimization)

3、流程图：



4、简介及特点：

可以发现，在高层结构与大体流程上与元启发式算法相比并没有发生改动，但实际上的改动发生在编码以及对应的进化方程上。设有 m 台机器。

编码：每个粒子的位置表示为一个 m 维的列表，列表中每个元素对应一个启发式规则，每个列表对应一组启发式规则在机器上的分布。本算法中共使用了 6 种启发式规则，分别为：

1.Rand (随机选择) 2.FIFO (First In First Out) 3.SJF (Shortest Job First) 4.WSJF(Weighted Shortest Job First) 5.LPT (Longest Processing Time) 6.EDF (Earliest Deadline First)

进化公式：由于编码方式及编码对应意义的改变，原进化方程不再适用。故在本算法中，采用 GA(Genetic algorithm) 中的 Crossover, Mutation 分别替换原进化公式中的 $f(X), g(X)$

其中，Crossover 采用 Two-point crossover

算法其余部分沿用前页元启发式算法中设计。

5、性能：（测试结果）

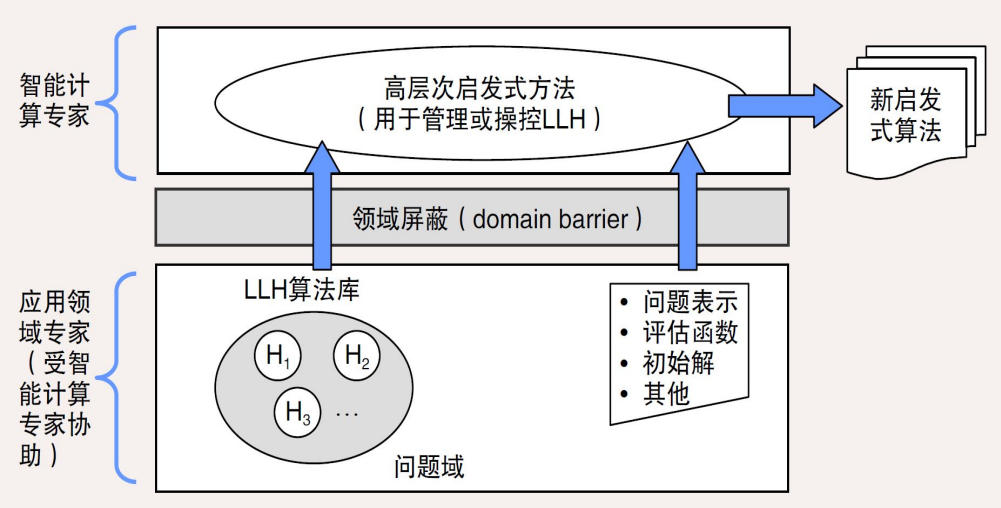
在实践过程中，相较于启发式规则，尤其是在求解较大规模问题上，算法得到了较优质的解。然而相较于元启发式算法，解的质量并没有出现大幅的提高，甚至有时在较大规模问题上表现略逊于元启发式算法。可能的原因有如下几点：

1） 本算法中仅使用了 6 种启发式规则，在得到较好解的超启发式算法中大多使用了 7-10 甚至更多的启发式规则。故本算法中的启发式规则数量或许不足以使其构造出得到较优解的规则分布。

2） 超启发式算法本身基于启发式规则，解空间由规则在机器上的分布构成，故超启发式算法的解空间为 JSP 问题解空间的子集，有可能存在优质解不在超启发式算法的解空间中的情况。出现这种情况的一个原因即如 1）所述：启发式规则数量不足

然而，超启发式算法设计的初衷是为了实现算法的普适性，这种普适性不仅仅是同类问题中不同问题结构，更是不同类问题之间的普适性。在实践中发现相较于前两种算法具有如下优点：

1） 如图所示，算法实现了领域屏蔽：



这种领域屏蔽使得算法具有良好的可移植性和可扩展性。即，仅需更换 LLH 算法库即可实现对不同类问题的求解，且领域专家在该问题任意分支上做出了算法的改进均可直接将其算法添加至 LLH 算法库中，使得改进的算法用于提高整个该类问题解的质量。

2） 缩小了问题的解空间。在 n 个工件 m 个机器，p 个 LLH 算法（启发式规则）的情况下，超启发式算法的解空间为 p^m，元启发式算法解空间为 $\frac{(n\times m)!}{(m!)^n}$ 故，相较于元启发式算法，在较大规模(n 较大，即工件较多)的情况下，超启发式算法耗时显著低于元启发式算法。

3） 解的质量具有较好的稳定性，不会如元启发式算法一般过度依赖于随机数以及随机初始化时的初始状态。

测试结果：（测试用例）

表中数值为测试用例运行 20 次取平均值得到

	测试一	测试二	测试三	测试四
启发式规则	79	137	280	512
元启发式算法	79	135.1	270	421.68
超启发式算法	79	135	270	425.26

表中用例为上表中测试四用例

	最优解	最差解	均值	方差
启发式规则	512	512	512	0
元启发式算法	407	431	421.68	45.85
超启发式算法	418	428	425.26	8.02

小结与展望：

在本次实践中，发现 PSO 算法仍有许多可以改进的地方，如：

1） 在种群最优记录(gB)影响粒子时，可以通过轮盘赌，依概率从全部个体最优记录(pB)中选出 gB’代替 gB 对该粒子施加种群的影响。

2） 在种群每轮位置更新结束后取当前个体最优记录(pB)中最小值作为 gB’，通过轮盘赌依概率更新种群最优记录(gB)，以 PSO 避免早熟

3） 在每个粒子位置更新后，用当前粒子位置作为 pB’，通过轮盘赌依概率更新个体最优记录(pB)，增强了搜索能力

而在结合了 GA(Genetic algorithm)中操作的超启发算法中可以考虑引入变异算子进行辅助。另，由于许多 NP-H 问题的解均具有比较明显的解空间特征，如“大坑”结构，即全局最优解被一系列的优质解包围。一个典型的例子是 TSP 问题中，优质解与优质解之间，以及优质解与全局最优解之间，有约 80%的重合边，故实际构造解的过程中可以通过搜索记录将多次被重复选中的边确定下来，显著地减小搜索空间。本此实践中未使用这种特性来构造解，但这方面的文章并不少见。进一步的，也有人提出超启发式算法中的解空间结构也具有类似特征，但如何具体的利用解空间特征缩小搜索空间，还有待进一步研究。

参考文献：

[1] Pisut Pongchairerks Particle swarm optimization algorithm applied to scheduling problems, ScienceAsia 35 (2009):89–94 |doi: 10.2306/scienceasia1513-1874.2009.35.089

[2] Jin Yan, Xiuli Wu A genetic based hyper-heuristic algorithm for the job shop scheduling problem, 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics

[3] 刘志雄，杨光祥 基于演化策略算法的作业车间调度优化：

1000—3428(2010)19—0008—03

[4] 江贺 超启发式算法：跨领域的问题求解模式，中国计算机协会通讯，第 7 卷，第 3 期，

2011年 3 月

[5] 基于粒子群算法的车间作业调度研究，李小华，武汉科技大学硕士学位论文，2009 年 5

月 4 日:10~28, 36~42