混合流水车间调度 HFSP 优化 CPLEX 和 GA

算法 Matlab 实现视频教程配套资料

#### 作者：howarwang More on: [www.iescm.com/wor](http://www.iescm.com/wor)

**目录**

混合流水车间调度 HFSP 优化算法设计 1

1. [混合流水车间调度问题描述 1](#_TOC_250010)
2. [HFSP 问题数学模型 1](#_TOC_250009)
3. [HFSP 小规模问题 CPLEX 优化求解 3](#_TOC_250008)
   1. [小规模算例数据 3](#_TOC_250007)
   2. [CPLEX 模型设计 4](#_TOC_250006)
   3. [CPLEX 优化结果 6](#_TOC_250005)
4. [HFSP 求解 GA 算法 Matlab 实现 11](#_TOC_250004)
   1. [GA 算法框架 11](#_TOC_250003)
   2. [编码和解码规则 11](#_TOC_250002)
   3. [遗传操作选择 13](#_TOC_250001)
   4. [GA 算法Matlab 程序 13](#_TOC_250000)

# 混合流水车间调度问题描述

#### 混合流水车间调度问题（Hybrid Flow Shop Scheduling Problem，HFSP）也称为柔性流水车间调度问题（Flexible Flow Shop Scheduling Problem，FFSP），因为该问题是在 FSP 基础上增加了柔性，即在 m 道工序中至少有一道工序的机器数量为两台或多台。HFSP 的其他特征如下：

#### 车间要进行 n 个批次或 n 个种类产品的生产；

#### 这些批次或种类的产品的工艺路径都是相同的，都需要经过 m 道工序，不同的是不同产品在同一道工序上的加工时间可能不同；

#### 在m 到工序中存在至少一道工序的机器数量为两台或多台，且这些机器性能相同。

#### 优化目标设定为：最大完工时间最小。

#### 按照车间调度问题的三元组表示法，HFSP 规范标识为：

*F* *k*1, *k*2 ,Λ, *km*  || *C*max ，

# HFSP 问题数学模型

#### 数学模型的构建目标是将问题形成整数规划模型、混合整数规划模型或线性规划模型，以便利用一些运筹学软件对其进行小规模问题的优化求解，进而对比为了求解大规模应用问

#### 题的规则算法或启发式算法的性能。NP 难问题，令：

#### j,j’=1,…n 为作业的索引；job h=1,…, *ki* 为工序 i 中的机器索引， i=1,…,m 为工序索引

#### *ki* ：为工序 i 中的机器数量，

#### *Gi* ：为工序 i 中的机器集合，

#### *Cij* ：为工序 i 中开工顺序为 j 的作业的完工时间，n 个在第 i 个工序中，他们开工顺序如何？

*Mhi* ：为机器集合*Gi* 中的第 h 个机器。决策变量：

*x*  1

*jj* '*hi* 0



在机器*Mhr*上作业*j*为作业*j'*的紧前作业

#### j,j’=0,1..n,

*x*0 *j* '*hi*

*xj* 0*hi*

#### Z

###  1



0

###  1

0



作业*j'*为机器*Mhr*上第一个作业作业*j'*为机器*Mhr*上最后一个作业

#### HFSP 比较经典的混合整数规划模型（Mixed Integer Programming，MIP）如下：

#### Min Z （1）

#### Subject to

*n ki*

 *x jj* '*hi*  1

*j* 0 *h*1

*j* ', *i*

（2）

*n ki*

 *x jj* '*hi*  1

*j* '0 *h*1

*j*, *i*

（3）

*n n*

 *x jj* '*hi*   *x j* ' *jhi*  0

*j* ', *h*, *i*

*j* 0

*j* 0

*x jjhi*  0

*j*, *h*, *i*

#### （5）

*ki*  *ki* 

*Cij*   *x jj* '*hi pij* '    *x jj* '*hi* 1 *M*  *Cij* '

*j*, *j* ', *i*

*h*1

 *h*1 

*Cij*  *Ci*1, *j*  *pij*

*C*0 *j*  0 0  *Cij Cmj*  *Z*

*j*, *i*

*j*

*i*, *j*

*j*

#### （8）

#### （9）

#### （10）

*x jj* '*hi*  0,1

*j*, *j* ', *h*, *i*

#### 式（2）为任何作业在同一道工序中的所有机器上只能有一个紧前作业的约束；式（3）为任何作业在同一道工序的所有机器上只能作为一个作业的紧前作业的约束；式（4）为任何作业的加工不论是否出现在某台机器上，其紧前作业或紧后作业变量必然相等；式（5）同一个机器上同一个位置的作业之间不存在紧前紧后关系；式（6）当前机器各个顺序作业完工时间之间的约束；式（7）特定作业前后道工序之间完工时间的约束；式（8）为各项作业准备时间为 0 的约束；式（9）为各项作业在各道序上的完工时间不小于 0 约束；式（10）目标函数与各作业在最后一道序完工时间之间的约束；式（11）决策变量 0‐1 约束。

# HFSP 小规模问题 CPLEX 优化求解

#### CPLEX， IBM

#### 尝试利用 CPLEX 对 HFSP 进行小规模问题的优化求解。

## 小规模算例数据

#### 这里自行设置算例进行求解，如下两张表给出算例总的数据，工序最多 10 道，作业最

#### 多 50 个，后续实验中从中选取数据进行较小规模的实验。

#### 表 1 工序设备数量表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 工序 | Proc1 | Proc2 | Proc3 | Proc4 | Proc5 | Proc6 | Proc7 | Proc8 | Proc9 | Proc10 |
| 数量 | 2 | 3 | 2 | 1 | 2 | 4 | 4 | 3 | 3 | 4 |

#### 表 2 作业工序工时表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Proc1 | Proc2 | Proc3 | Proc4 | Proc5 | Proc6 | Proc7 | Proc8 | Proc9 | Proc10 |
| J1 | 61 | 77 | 82 | 17 | 104 | 96 | 33 | 102 | 89 | 79 |
| J2 | 89 | 28 | 29 | 52 | 103 | 98 | 64 | 60 | 35 | 52 |
| J3 | 55 | 103 | 58 | 89 | 5 | 10 | 78 | 86 | 8 | 64 |
| J4 | 66 | 31 | 61 | 60 | 35 | 83 | 103 | 32 | 57 | 57 |
| J5 | 66 | 84 | 62 | 77 | 59 | 60 | 96 | 79 | 7 | 86 |
| J6 | 39 | 51 | 83 | 75 | 95 | 101 | 40 | 80 | 96 | 38 |
| J7 | 80 | 80 | 36 | 77 | 53 | 97 | 54 | 74 | 103 | 87 |
| J8 | 72 | 96 | 92 | 96 | 95 | 99 | 34 | 12 | 68 | 26 |
| J9 | 81 | 18 | 18 | 81 | 24 | 30 | 82 | 33 | 44 | 87 |
| J10 | 27 | 27 | 13 | 52 | 88 | 54 | 65 | 47 | 35 | 62 |
| J11 | 93 | 52 | 96 | 63 | 9 | 34 | 19 | 61 | 75 | 45 |
| J12 | 6 | 102 | 53 | 70 | 73 | 43 | 83 | 35 | 47 | 67 |
| J13 | 56 | 7 | 74 | 90 | 20 | 48 | 28 | 70 | 91 | 52 |
| J14 | 90 | 30 | 10 | 83 | 56 | 5 | 69 | 87 | 22 | 67 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| J15 | 54 | 92 | 42 | 28 | 105 | 34 | 96 | 65 | 83 | 34 |
| J16 | 76 | 22 | 78 | 100 | 64 | 11 | 66 | 84 | 28 | 80 |
| J17 | 74 | 76 | 12 | 69 | 37 | 94 | 39 | 105 | 100 | 38 |
| J18 | 26 | 9 | 33 | 15 | 87 | 104 | 78 | 38 | 59 | 71 |
| J19 | 11 | 51 | 31 | 41 | 90 | 100 | 7 | 28 | 101 | 69 |
| J20 | 49 | 65 | 64 | 96 | 13 | 31 | 44 | 24 | 65 | 88 |
| J21 | 53 | 96 | 72 | 96 | 96 | 66 | 100 | 80 | 19 | 62 |
| J22 | 46 | 93 | 42 | 62 | 53 | 33 | 89 | 8 | 42 | 58 |
| J23 | 68 | 92 | 45 | 50 | 49 | 91 | 35 | 17 | 6 | 23 |
| J24 | 7 | 50 | 59 | 60 | 40 | 24 | 69 | 26 | 9 | 84 |
| J25 | 43 | 90 | 63 | 101 | 51 | 9 | 97 | 88 | 52 | 5 |
| J26 | 20 | 37 | 29 | 77 | 53 | 71 | 57 | 21 | 96 | 52 |
| J27 | 25 | 35 | 42 | 73 | 26 | 6 | 6 | 36 | 46 | 41 |
| J28 | 17 | 55 | 105 | 80 | 59 | 87 | 63 | 49 | 44 | 80 |
| J29 | 102 | 88 | 97 | 71 | 94 | 97 | 103 | 46 | 29 | 24 |
| J30 | 75 | 93 | 89 | 60 | 85 | 16 | 51 | 56 | 75 | 19 |
| J31 | 57 | 81 | 7 | 79 | 19 | 46 | 61 | 6 | 67 | 7 |
| J32 | 33 | 36 | 46 | 40 | 54 | 81 | 9 | 44 | 46 | 45 |
| J33 | 53 | 57 | 74 | 70 | 22 | 33 | 71 | 30 | 67 | 88 |
| J34 | 101 | 83 | 11 | 56 | 93 | 5 | 44 | 26 | 74 | 49 |
| J35 | 37 | 48 | 25 | 82 | 103 | 24 | 88 | 54 | 103 | 82 |
| J36 | 40 | 71 | 60 | 32 | 67 | 99 | 49 | 68 | 64 | 92 |
| J37 | 87 | 42 | 6 | 9 | 57 | 71 | 18 | 9 | 78 | 20 |
| J38 | 99 | 49 | 91 | 88 | 39 | 21 | 36 | 16 | 34 | 65 |
| J39 | 100 | 104 | 65 | 11 | 9 | 94 | 87 | 15 | 102 | 74 |
| J40 | 24 | 68 | 64 | 60 | 82 | 20 | 16 | 53 | 58 | 99 |
| J41 | 18 | 39 | 32 | 32 | 64 | 61 | 19 | 73 | 85 | 75 |
| J42 | 5 | 45 | 57 | 10 | 14 | 84 | 104 | 101 | 49 | 69 |
| J43 | 14 | 92 | 76 | 36 | 32 | 97 | 40 | 53 | 57 | 28 |
| J44 | 96 | 90 | 16 | 42 | 84 | 77 | 103 | 57 | 100 | 8 |
| J45 | 74 | 17 | 20 | 37 | 98 | 38 | 12 | 70 | 18 | 41 |
| J46 | 65 | 52 | 33 | 8 | 63 | 76 | 76 | 98 | 24 | 86 |
| J47 | 94 | 35 | 84 | 58 | 71 | 93 | 98 | 28 | 63 | 87 |
| J48 | 53 | 91 | 44 | 6 | 54 | 62 | 48 | 44 | 103 | 48 |
| J49 | 97 | 103 | 25 | 26 | 93 | 7 | 93 | 17 | 89 | 100 |
| J50 | 13 | 47 | 17 | 69 | 24 | 8 | 101 | 55 | 101 | 77 |

## CPLEX 模型设计

#### myHFSP.mod 模型文件‐数学模型文件

int jobQty=...; int procQty=...;

range jobs=1..jobQty;

range processes=1..procQty;

int machQty[processes]=...; //每个工序机器数量

int maxMach[processes]=...; //设置决策变量为0的起点 int ptimes[jobs][processes]=...;//作业工时数据

int maxMachQ=max(i in processes) machQty[i]; range eachMachQty=1..maxMachQ;//

int bigM=10 \* sum(j in jobs,i in processes) ptimes[j][i];//非常大的数字

dvar boolean x[0..jobQty][0..jobQty][eachMachQty][processes]; dvar int+ C[0..procQty][0..jobQty];

dvar int+ Z; minimize Z;

subject to {

hasPreConstraint://一个作业在某个阶段的全部机器中，必然有一个紧前工序 forall(j2 in jobs,i in processes)

sum(j1 in 0..jobQty, h in 1..machQty[i]) x[j1][j2][h][i] ==1;

notStartMoreJobs://同一台机器不能有两个处于第一位的作业 forall(i in processes) forall(h in 1..machQty[i])

sum(j1 in 0..jobQty) x[0][j1][h][i] <=1;

forall(j2 in jobs,i in processes)

sum(j1 in 0..jobQty, h in 1..machQty[i]) x[j2][j1][h][i] ==1; notFinishMoreJobs://同一台机器不能有两个处于最后一位的作业 forall(i in processes) forall(h in 1..machQty[i])

sum(j1 in 0..jobQty) x[j1][0][h][i] <=1;

forall(j2 in jobs,i in processes) forall(h in 1..machQty[i])

sum(j1 in 0..jobQty) x[j2][j1][h][i] ==sum(j1 in 0..jobQty) x[j1][j2][h][i]; forall(j2 in jobs,i in processes)

forall(h in 1..machQty[i]) x[j2][j2][h][i] ==0;

finishTimes:

forall(j1 in 0..jobQty,j2 in jobs,i in processes)

C[i][j1]+sum(h in 1..machQty[i]) x[j1][j2][h][i]\*ptimes[j2][i]

+bigM\*((sum(h in 1..machQty[i]) x[j1][j2][h][i])‐1)<=C[i][j2]; forall(j1 in jobs,i in processes)

C[i][j1]>=C[i‐1][j1]+ptimes[j1][i]; forall(j in jobs,i in processes)

C[i][j]<=Z;

// forall(j1 in 0..jobQty,j2 in jobs,i in processes:i!=2)

// forall(h in maxMach[i]..maxMachQ)

// x[j1][j2][h][i]==0;

}

execute DISPAY{

//对每个工序的每台设备上的作业前后安排显示出来，【工序+机器+各个作业】

var preMachQty=0;

for(var i=1;i<=procQty;i++){ if(i>1)

{

preMachQty+=machQty[i‐1];

}

//writeln("the sequence detail of process: "+ i); for(var h=1;h<=machQty[i];h++){

//writeln(" the sequence detail of machine: "+ h); for(var j1=0;j1<=jobQty;j1++){

for(var j2=1;j2<=jobQty;j2++){ if(x[j1][j2][h][i]==1){

writeln(j2 +" "+ (h+preMachQty) +" "+ i+" "+ (C[i][j2]‐ptimes[j2][i])+ " "+C[i][j2]+";");

}

}

}

}

}

}

## CPLEX 优化结果

#### （1）M=3，N=4

#### 数据文件 myHFSP.dat

jobQty=4; procQty=3; machQty=[2,3,2];

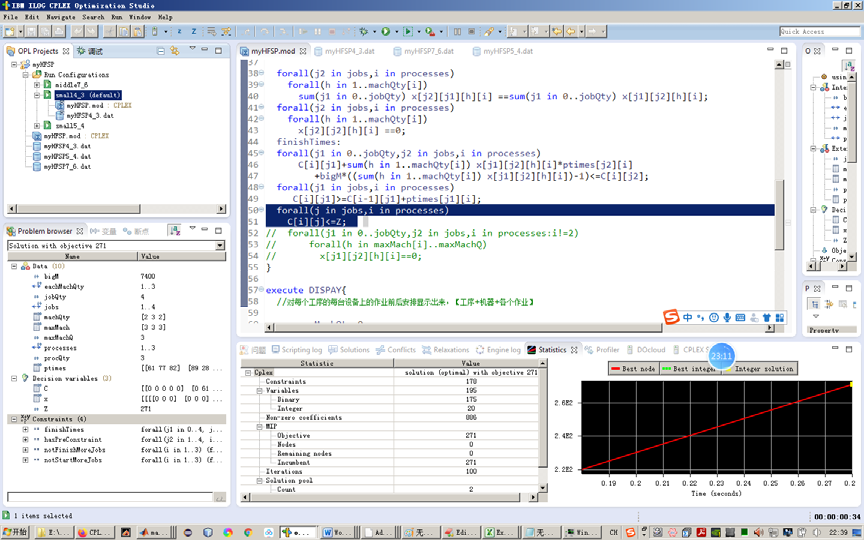
maxMach=[3,3,3];

ptimes=[[61,77,82],

[89,28,29],

[55,103,58],

[66,31,61]];



#### 图 1 n=4,m=3 算例模型界面和结果界面结果，目标函数：271，具体调度方案如表 3。

#### 表 3 n=4,m=3 算例优化详细调度方案

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| jobId | machId | processId | startTime | endTime |
| 1 | 1 | 1 | 0 | 61 |
| 3 | 2 | 1 | 0 | 55 |
| 4 | 2 | 1 | 55 | 121 |
| 2 | 2 | 1 | 121 | 210 |
| 3 | 3 | 2 | 55 | 158 |
| 2 | 3 | 2 | 210 | 238 |
| 1 | 4 | 2 | 61 | 138 |
| 4 | 5 | 2 | 121 | 152 |
| 4 | 6 | 3 | 152 | 213 |
| 3 | 6 | 3 | 213 | 271 |
| 1 | 7 | 3 | 138 | 220 |
| 2 | 7 | 3 | 238 | 267 |

#### 将上述调度方案绘制甘特图，用于检查结果也好，还是易于理解也好。绘图的 Matlab

#### 程序如下：

drawGant(sch);

%根据调度方案绘制甘特图程序

%schedule:1-jobId,2-machId,3-procId,4-startTime,5-endTime function drawGant(schedule)

rows=size(schedule,1); maxMachId=max(schedule(:,2)); jobQty=max(schedule(:,1));

mycolor=rand(jobQty,3); figure;

ylim([0 maxMachId+1]); for i=1:rows

x=schedule(i,4:5); y=[schedule(i,2) schedule(i,2)];

line(x,y,'lineWidth',16,'color',mycolor(schedule(i,1),:)); procId=schedule(i,3);

jobId=schedule(i,1);

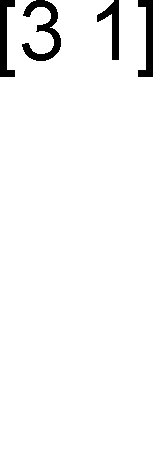
txt=['[' int2str(jobId) ' ' int2str(procId) ']']; text(mean(x)-1,y(1),txt);

end end

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| sch=[1 | | | 1 1 0 61; | |
| 3 2 1 | | | 0 55; | |
| 4 2 1 | | | 55 121; | |
| 2 2 1 | | | 121 210; | |
| 3 3 2 | | | 55 158; | |
| 2 3 2 | | | 210 238; | |
| 1 4 2 | | | 61 138; | |
| 4 | 5 | 2 | 121 | 152; |
| 4 | 6 | 3 | 152 | 213; |
| 3 | 6 | 3 | 213 | 271; |
| 1 | 7 | 3 | 138 | 220; |
| 2 | 7 | 3 | 238 | 267]; |

#### 绘图结果如下：





#### 图 2 n=4,m=3 算例调度方案甘特图

#### （2）M=4，N=5数据文件

jobQty=5; procQty=4; machQty=[2,3,2,1];

maxMach=[3,4,3,2];

ptimes=[[61,77,82,17],

[89,28,29,52],

[55,103,58,89],

[66,31,61,60],

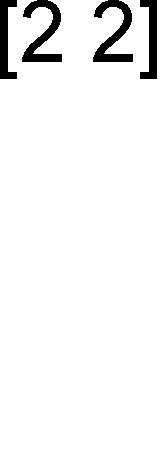
[66,84,62,77]

];

#### 表 4 M=4，N=5 算例详细调度数据

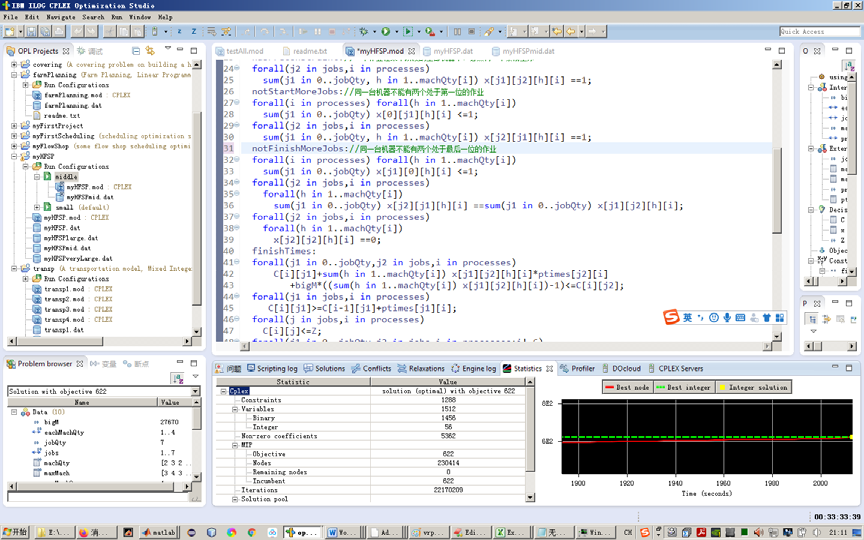
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| jobId | machId | processId | startTime | endTime |
| 4 | 1 | 1 | 0 | 66 |
| 2 | 1 | 1 | 66 | 155 |
| 5 | 2 | 1 | 0 | 66 |
| 3 | 2 | 1 | 131 | 186 |
| 1 | 2 | 1 | 66 | 127 |
| 5 | 3 | 2 | 66 | 150 |
| 2 | 3 | 2 | 232 | 260 |
| 3 | 4 | 2 | 186 | 289 |
| 4 | 5 | 2 | 66 | 97 |
| 1 | 5 | 2 | 277 | 354 |
| 4 | 6 | 3 | 97 | 158 |
| 1 | 6 | 3 | 354 | 436 |
| 5 | 7 | 3 | 150 | 212 |
| 3 | 7 | 3 | 289 | 347 |
| 2 | 7 | 3 | 260 | 289 |
| 4 | 8 | 4 | 158 | 218 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 8 | 4 | 347 | 436 |
| 1 | 8 | 4 | 436 | 453 |
| 5 | 8 | 4 | 218 | 295 |
| 2 | 8 | 4 | 295 | 347 |



#### 图 3 M=4，N=5 算例调度数据对应甘特图

#### （3）M=6，N=7



#### 运行 33 分钟，获得目标函数最小值：最早完工时间 Z=622，以及对应的调度方案如下表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| jobId | machId | processId | startTime | endTime |
| 6 | 1 | 1 | 0 | 39 |
| 7 | 1 | 1 | 171 | 251 |
| 1 | 1 | 1 | 110 | 171 |
| 5 | 1 | 1 | 39 | 105 |
| 2 | 2 | 1 | 0 | 89 |
| 3 | 2 | 1 | 167 | 222 |
| 4 | 2 | 1 | 286 | 352 |
| 1 | 3 | 2 | 171 | 248 |
| 7 | 3 | 2 | 251 | 331 |
| 6 | 4 | 2 | 64 | 115 |
| 4 | 4 | 2 | 352 | 383 |
| 2 | 5 | 2 | 89 | 117 |
| 5 | 5 | 2 | 117 | 201 |
| 3 | 5 | 2 | 222 | 325 |
| 2 | 6 | 3 | 117 | 146 |
| 5 | 6 | 3 | 211 | 273 |
| 4 | 6 | 3 | 383 | 444 |
| 3 | 6 | 3 | 325 | 383 |
| 6 | 7 | 3 | 115 | 198 |
| 7 | 7 | 3 | 331 | 367 |
| 1 | 7 | 3 | 248 | 330 |
| 2 | 8 | 4 | 146 | 198 |
| 7 | 8 | 4 | 367 | 444 |
| 6 | 8 | 4 | 198 | 273 |
| 3 | 8 | 4 | 518 | 607 |
| 1 | 8 | 4 | 350 | 367 |
| 5 | 8 | 4 | 273 | 350 |
| 4 | 8 | 4 | 444 | 504 |
| 2 | 9 | 5 | 198 | 301 |
| 7 | 9 | 5 | 444 | 497 |
| 3 | 9 | 5 | 607 | 612 |
| 5 | 9 | 5 | 503 | 562 |
| 6 | 10 | 5 | 273 | 368 |
| 4 | 10 | 5 | 504 | 539 |
| 1 | 10 | 5 | 400 | 504 |
| 7 | 11 | 6 | 515 | 612 |
| 3 | 11 | 6 | 612 | 622 |
| 4 | 12 | 6 | 539 | 622 |
| 6 | 13 | 6 | 461 | 562 |
| 5 | 13 | 6 | 562 | 622 |
| 2 | 14 | 6 | 301 | 399 |
| 1 | 14 | 6 | 526 | 622 |

15

[2 6]

[6 6]

[3 6]

[2 5]

[3 5]

[2 4]

[2 3]

[3 2]

[4 2]

[2 1]

[3 1]

[4 1]

[6 1]

[5 1]

[1 1]

[7 1]

[7 2]

[1 2]

[6 2]

[5 2]

[5 3]

[7 3]

[1 3]

[3 4]

[4 4]

[7 4]

4]

[1

[5 4]

[5 5]

[7 5]

[4 5]

[1 5]

[6 4]

[2 2]

[6 3]

[3 3]

[4 3]

[6 5]

|  |  |  |
| --- | --- | --- |
| [1 6] | | |
|  | | [5 6] |
|  | [4 6] | |
| [7 6] | | |

10

5

0

0 100 200 300 400 500 600 700

#### 图 4 调度方案甘特图

# HFSP 求解 GA 算法 Matlab 实现

## GA 算法框架

#### Step1： 问题参数初始化，作业数量 n，工序数量 m，每个工序机器数量 km，每个作业工序工时 ptimes[n][m]，等等；

#### Step2： 算法参数初始化，种群数量、变异概率、交叉概率、终止条件

#### Pop=40,60

#### Step3：种群初始化，如何设计编码规则和解码规则

#### Step4：进行选择、交叉、变异操作

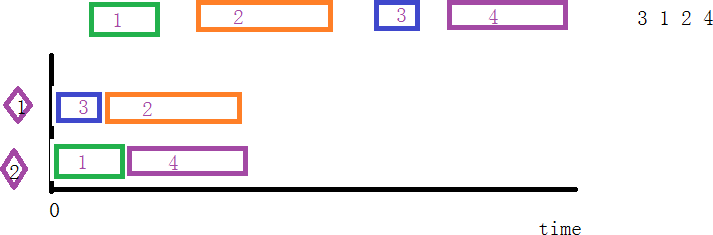
#### 选择：精英保留策略，先选最好的几个保留到下一代，然后运用轮盘赌策略从种群中选择剩下的染色体保留到下一代，直至下代种群达到 pop 数量。

#### Step5：输出最终的解

## 编码和解码规则

#### 求解的基本思路

#### 对每个工序的 n 个作业进行排序，利用优先规则选择最早可用的机器将作业排到对应的机器上。



#### 图 5 遗传算法求解 HFSP 基本思路示意图（详细解释不写文字，请看视频）

#### 编码规则

#### 采用基于操作的整数编码方式，具体规则为编码长度只同作业数量 n 和工序数量 m 相关，编码长度就是 n\*m，编码内容是：1,2,……,n\*m 之间的整数值。

#### 以 n=4，m=3 为例进行编码和操作*oji* 之间的对应关系说明。

#### 表？ 示例 GA 算法编码基础对应表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编码值 | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 解码 | 作业 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 工序 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 |

#### 解码规则

#### 解码规则必然同编码规则要相对应，同样以上述示例进行说明。某个染色体为： 8 3 11 2 6 12 5 1 10 9 4 7

#### 解码步骤：

#### Step1: 当前染色体 nowChrome,长度为 n\*m=L； Step2: 定义详细调度方案数组 schedule[L][5] Step2: 令当前工序 i=1

#### Step3: 从 nowChrome 中根据“基础对应表”找出工序 i 的全部作业的编码，保留其在

#### nowChrome 中顺序，假设为 procSeq，找出对应的作业编码 jobSeq nowChrome 中的数字小于等于：m\*i‐‐‐

#### 若 i=1， procSeq=[3 2 1 4], jobSeq=[3 2 1 4]

#### 若 i=2，procSeq=[8 6 5 7], jobSeq=[4 2 1 3]

#### nowChrome 中的数字小于等于 m\*i‐[8 3 2 6 5 1 4 7]

#### 再从上述数列中找出大于 m\*(i‐1)的数字：[8 6 5 7]

#### 将上述（数字‐1）对 m 取余并＋1，7%4+1

….

#### Step4: 根据当前工序 i 的作业编码排序 jobSeq，将作业安排到机器上以 i=1 的 jobSeq[3 2 1 4]为例说明如何具体调度方案表，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| machId | jobId | processId | startTime | endTime |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

#### Step4.1 令 j=1

#### Step4.2 获得 jobSeq[j]的作业编号 nowJ，当然可以获得 ptimes[nowJ][i]这个作业工时 Step4.3 将作业 nowJ 根据优先规则安排到最早可用机器，并完成 schedule 一行数据的

#### 设定

由工序 i 可以获得当前可用机器的编号 nowMachIds, 利用类似于下表的数组

### machProcArray；

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 机器 id | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 工序 | 1 | 1 | 2 | 2 | 2 | 3 | 3 |

### nowMachIds=machProcArray(1,machProcArray(2,:)==i);

#### 较好的一种方式：

#### 从 schedule 中查看可用机器 id 的全部排产数据，存入 nowSchedule，

#### 根据 nowSchedule 中每台机器的排产数据，生成 emptySlot 数据

#### Case1：某个机器 id 在 nowSchedule 中有数据，则将其按照 startTime 升序排序，然后逐条生成 emptySlot，

#### 例如现有排序如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| jobId | machId | processId | startTime | endTime |
| 2 | 3 | 2 | 144 | 172 |
| 1 | 4 | 2 | 61 | 138 |
| 3 | 4 | 2 | 138 | 241 |

#### 生成的 emptySlot 如下：

|  |  |  |
| --- | --- | --- |
| machId | startTime | endTime |
| 3 | 0 | 144 |
| 3 | 172 | M |
| 4 | 0 | 61 |
| 4 | 138 | 138 |
| 4 | 241 | M |
| 5 | 0 | M |

#### 对 emptySlot 利用 startTime 进行升序排列，逐条判断该 slot 是否能够安排拟排产的操作【操作所需的时间是否<=slot 的间距】

#### Step4.4 j=j+1，如果 j<=n，执行 Step4.2；否则执行 Step5； Step5：令 i=i+1，如果 i<=m,执行 Step3；否则解码结束。

## 遗传操作选择

#### 选择操作 Selection，采用精英保留策略，保留精英数量为 5；

#### 交叉操作 Crossover

#### 变异操作 mute

## GA 算法 Matlab 程序

%%

% GA算法求解混合流水车间调度问题

%%

function HFSPbyGA() [stageMachQty,ptimes]=initProblem20\_8(); machProcesses=getMachProc(stageMachQty); [jobQty,procQty]=size(ptimes); gaParas=initGA();

pop=gaParas(1); crossRate=gaParas(2); muteRate=gaParas(3); elitistQty=gaParas(4); pLength=jobQty\*procQty; chromes=initChromes(pop,pLength);

fit=getFitnesses(chromes,ptimes,machProcesses);

%初始化最优解 [sortFit,sortIdx]=sortrows(fit); optFit=sortFit(1); optChrome=chromes(sortIdx(1),:);

maxGeneration=500; nowGeneration=0;

while(maxGeneration>=nowGeneration)

%进行遗传的迭代操作

%交叉操作 crossedChromes=crossChromes(chromes,crossRate);

%变异操作 mutedChromes=muteChromesInsert(crossedChromes,muteRate);

%选择操作 fit=getFitnesses(mutedChromes,ptimes,machProcesses);

chromes=elitestSelection(fit,mutedChromes,elitistQty);

%记录最优的解 [sortFit,sortIdx]=sortrows(fit); if(sortFit(1)<=optFit)

optFit=sortFit(1) optChrome=mutedChromes(sortIdx(1),:);

end

nowGeneration=nowGeneration+1;

end

%最优结果显示 optFit

schedule=getSchedule2(optChrome,ptimes,machProcesses)

end

%变异操作--插入方法

function outChromes=muteChromesInsert(chromes,muteRate) [pop,len]=size(chromes);

for i=1:pop nowChrome=chromes(i,:); if rand()<muteRate

points=randperm(len,2); if points(1)<points(2)

insertChrome=nowChrome(points(1):points(2)-1); newChrome=circshift(insertChrome,-1); chromes(i,points(1):points(2)-1)=newChrome;

else

insertChrome=nowChrome(points(2)+1:points(1)); newChrome=circshift(insertChrome,1); chromes(i,points(2)+1:points(1))=newChrome;

end end

outChromes=chromes; end

end

%交叉操作--两点交叉

function outChromes=crossChromes(chromes,crossRate) pop=size(chromes,1);

cols=size(chromes,2); for i=1:2:pop

if rand()<crossRate %进行交叉操作 parent1=chromes(i,:); parent2=chromes(i+1,:); p=sortrows(randperm(cols,2)')'; crossP1=parent1(p(1):p(2)); crossP2=parent2(p(1):p(2)); son1=parent1; son1(p(1):p(2))=crossP2; son2=parent2; son2(p(1):p(2))=crossP1;

%子片段对比 crossLen=size(crossP1,2); for j=crossLen:-1:1

midCode=crossP1(j); for k=1:size(crossP2,2)

if crossP2(k)==midCode crossP1(j)=[];

crossP2(k)=[]; break;

end end

end

%染色体编码置换 repeatNum=size(crossP1,2); if repeatNum>0

%对子代1的有效性置换 for j=1:repeatNum

midCode=crossP2(j); if p(1)==1

for k=p(2)+1:cols

if son1(k)==midCode son1(k)=crossP1(j); break;

end end

else

if p(2)==cols for k=1:p(1)-1

if son1(k)==midCode son1(k)=crossP1(j); break;

end end

else

getIt=0;

for k=1:p(1)-1

if son1(k)==midCode son1(k)=crossP1(j); getIt=1;

break; end

end

if getIt==0

for k=p(2)+1:cols

if son1(k)==midCode son1(k)=crossP1(j); break;

end end

end end

end end

%对子代2的有效性置换 for j=1:repeatNum

midCode=crossP1(j); if p(1)==1

for k=p(2)+1:cols

if son2(k)==midCode son2(k)=crossP2(j); break;

end end

else

if p(2)==cols for k=1:p(1)-1

if son2(k)==midCode son2(k)=crossP2(j); break;

end end

else

getIt=0;

for k=1:p(1)-1

if son2(k)==midCode son2(k)=crossP2(j); getIt=1;

break; end

end

if getIt==0

for k=p(2)+1:cols

if son2(k)==midCode son2(k)=crossP2(j); break;

end end

end end

end end

end

chromes(i,:)=son1; chromes(i+1,:)=son2;

end end

outChromes=chromes; end

%根据输入的染色体及其适应度进行精英保留策略复制下代种群 function

selectedChromosomes=elitestSelection(fitnessValue,mutedChromosomes,elitistQty)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% selection by elisist retained strategy %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[population,geneLth]=size(mutedChromosomes); [~,queueIndex]=sort(fitnessValue);

%get the elitest chromosomes elitist=zeros(elitistQty,geneLth); for num=1:elitistQty

elitist(num,:)=mutedChromosomes(queueIndex(num),:);

minFit=min(fitnessValue); if minFit<0

fitnessValue=fitnessValue+abs(minFit);

end maxFit=max(fitnessValue)\*1.2; fitnessValue=maxFit-fitnessValue; sumFitness=sum(fitnessValue); accumFitness=zeros(1,population); accumFitness(1)=fitnessValue(1);

%select chromosomes by roullette method for i=2:population

accumFitness(i)=accumFitness(i-1)+fitnessValue(i);

end

selectRate=unifrnd(0,sumFitness,[1 population]); selectedChromosomes=zeros(population,geneLth);

for i=1:population selectNumber=find(selectRate(i)<=accumFitness); selectedChromosomes(i,:)=mutedChromosomes(selectNumber(1),:);

end

%integrate the elistists into son chromosomes selectedChromosomes(1:elitistQty,:)=elitist; randIdx=randperm(population)'; selectedChromosomes=selectedChromosomes(randIdx,:);

end

%根据当前种群获取全部染色体的适应度

function fitnesses=getFitnesses(chromes,ptimes,machProcesses) [pop,~]=size(chromes);

fitnesses=zeros(pop,1); for i=1:pop

schedule=getSchedule2(chromes(i,:),ptimes,machProcesses); cmax=max(schedule(:,5));

fitnesses(i)=cmax;

end end

%Step4的小程序：根据stageMachQty数组生成工序和机器编码之间的二维数组 function machProcesses=getMachProc(stageMachQty)

procQty=max(size(stageMachQty)); cols=sum(stageMachQty); machProcesses=zeros(2,cols); midIdx=1;

for i=1:procQty myMachs=stageMachQty(i); for j=1:myMachs

machProcesses(2,midIdx)=i; machProcesses(1,midIdx)=midIdx; midIdx=midIdx+1;

end end

%Step1 问题相关数据初始化

function [stageMachQty,ptimes]=initProblem4\_3() stageMachQty=[2,3,2];

ptimes=[61 77 82;89 28 29;55 103 58;66 31 61];

end

%Step1 问题相关数据初始化

function [stageMachQty,ptimes]=initProblem5\_4() stageMachQty=[2,3,2,1];

ptimes=[61 77 82 17

|  |  |  |  |
| --- | --- | --- | --- |
| 89 | 28 | 29 | 52 |
| 55 | 103 | 58 | 89 |
| 66 | 31 | 61 | 60 |
| 66 | 84 | 62 | 77 |
| ]; |  |  |  |
| end |  |  |  |

%Step1 问题相关数据初始化

function [stageMachQty,ptimes]=initProblem7\_6()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| stageMachQty=[2 | | | | | 3 2 1 2 4]; |
| ptimes=[61 77 | | | | | 82 17 104 96 |
| 89 | 28 | 29 | 52 | 103 | 98 |
| 55 | 103 | 58 | 89 | 5 | 10 |
| 66 | 31 | 61 | 60 | 35 | 83 |
| 66 | 84 | 62 | 77 | 59 | 60 |
| 39 | 51 | 83 | 75 | 95 | 101 |
| 80 | 80 | 36 | 77 | 53 | 97 |
| ]; |  |  |  |  |  |
| end |  |  |  |  |  |

%Step1 问题相关数据初始化

function [stageMachQty,ptimes]=initProblem10\_8()

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| stageMachQty=[2 | | | | | 3 | 2 | 1 2 4 4 3]; |
| ptimes=[61 77 | | | | | 82 | 17 | 104 96 33 102 |
| 89 | 28 | 29 | 52 | 103 | 98 | 64 | 60 |
| 55 | 103 | 58 | 89 | 5 | 10 | 78 | 86 |
| 66 | 31 | 61 | 60 | 35 | 83 | 103 | 32 |
| 66 | 84 | 62 | 77 | 59 | 60 | 96 | 79 |
| 39 | 51 | 83 | 75 | 95 | 101 | 40 | 80 |
| 80 | 80 | 36 | 77 | 53 | 97 | 54 | 74 |
| 72 | 96 | 92 | 96 | 95 | 99 | 34 | 12 |
| 81 | 18 | 18 | 81 | 24 | 30 | 82 | 33 |
| 27  end | 27 | 13 | 52 | 88 | 54 | 65 | 47]; |
| %Step1 问题相关数据初始化  function [stageMachQty,ptimes]=initProblem20\_8() | | | | | | | |
| stageMachQty=[2 | | | | | 3 | 2 | 1 2 4 4 3]; |
| ptimes=[61 77 | | | | | 82 | 17 | 104 96 33 102 |
| 89 28 29 52 103 | | | | | 98 | 64 | 60 |
| 55 103 58 89 5 | | | | | 10 | 78 | 86 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 66 | 31 | 61 | 60 | 35 | 83 | 103 | 32 |
| 66 | 84 | 62 | 77 | 59 | 60 | 96 | 79 |
| 39 | 51 | 83 | 75 | 95 | 101 | 40 | 80 |
| 80 | 80 | 36 | 77 | 53 | 97 | 54 | 74 |
| 72 | 96 | 92 | 96 | 95 | 99 | 34 | 12 |
| 81 | 18 | 18 | 81 | 24 | 30 | 82 | 33 |
| 27 | 27 | 13 | 52 | 88 | 54 | 65 | 47 |
| 93 | 52 | 96 | 63 | 9 | 34 | 19 | 61 |
| 6 | 102 | 53 | 70 | 73 | 43 | 83 | 35 |
| 56 | 7 | 74 | 90 | 20 | 48 | 28 | 70 |
| 90 | 30 | 10 | 83 | 56 | 5 | 69 | 87 |
| 54 | 92 | 42 | 28 | 105 | 34 | 96 | 65 |
| 76 | 22 | 78 | 100 | 64 | 11 | 66 | 84 |
| 74 | 76 | 12 | 69 | 37 | 94 | 39 | 105 |
| 26 | 9 | 33 | 15 | 87 | 104 | 78 | 38 |
| 11 | 51 | 31 | 41 | 90 | 100 | 7 | 28 |
| 49 | 65 | 64 | 96 | 13 | 31 | 44 | 24 |
| ]; |  |  |  |  |  |  |  |
| end |  |  |  |  |  |  |  |

%Step2:算法参数初始化 function gaParas=initGA()

pop=10; crossRate=0.6; muteRate=0.4; elistQty=5;

gaParas=[pop,crossRate,muteRate,elistQty]; end

%Step3： 种群初始化

function chromes=initChromes(pop,pLength) chromes=zeros(pop,pLength);

for i=1:pop

chromes(i,:)=randperm(pLength);

end

end

%根据单一染色体获取其详细的调度方案

% jobId machId processId startTime endTime

function schedule=getSchedule(singleChrome,ptimes,machProcArray) [jobQty,procQty]=size(ptimes);

jobPreEndTime=zeros(1,jobQty); machQty=size(machProcArray,2); machPreEndTime=zeros(1,machQty); chromeLength=max(size(singleChrome)); schedule=zeros(chromeLength,5); schIdx=1;

for i=1:procQty

%首先获取当前工序的可用机器编码 nowMachIds=machProcArray(1,machProcArray(2,:)==i);

midIdx=singleChrome<=jobQty\*i; midChrome=singleChrome(midIdx);

midIdx=midChrome>jobQty\*(i-1); midChrome=midChrome(midIdx); jobSeq=mod((midChrome+1),jobQty)+1;

%依次为jobSeq中的作业安排当前的机器及其开工、完工时间 for j=1:jobQty

%获取这些可用机器中最早可用的机器编码 nowMachEndTime=machPreEndTime(1,nowMachIds); [~,idx]=sortrows(nowMachEndTime'); earlyMachId=nowMachIds(idx(1));

%根据可用机器的最早可用时间和当前作业的最早可用时间确定改作业

%在该到工序上的开工时间，完工时间等信息 myStartTime=max(machPreEndTime(earlyMachId),jobPreEndTime(jobSeq(j))); myEndTime=myStartTime+ptimes(jobSeq(j),i); schedule(schIdx,:)=[jobSeq(j),earlyMachId,i,myStartTime,myEndTime];

%更新作业和机器的可用时间 machPreEndTime(earlyMachId)=myEndTime; jobPreEndTime(jobSeq(j))=myEndTime; schIdx=schIdx+1;

end

end end

%根据单一染色体获取其详细的调度方案

% jobId machId processId startTime endTime

function schedule=getSchedule2(singleChrome,ptimes,machProcArray) bigM=1000000;

[jobQty,procQty]=size(ptimes); jobPreEndTime=zeros(1,jobQty); machQty=size(machProcArray,2);

chromeLength=max(size(singleChrome)); schedule=zeros(0,5);

for i=1:procQty

%首先获取当前工序的可用机器编码 nowMachIds=machProcArray(1,machProcArray(2,:)==i); nowMachQty=size(nowMachIds,2); midIdx=singleChrome<=jobQty\*i; midChrome=singleChrome(midIdx); midIdx=midChrome>jobQty\*(i-1); midChrome=midChrome(midIdx); jobSeq=mod((midChrome+1),jobQty)+1;

%依次为jobSeq中的作业安排当前的机器及其开工、完工时间 for j=1:jobQty

nowJobId=jobSeq(j);

%获取这些可用机器中最早可用的机器编码 nowSchedule=zeros(0,5);

%获取当前工序各台机器已经安排的调度方案

for k=1:nowMachQty midMachId=nowMachIds(k);

midSch=schedule(schedule(:,2)==midMachId,:); nowSchedule=[nowSchedule;midSch];

end

%获取当前工序各台机器可用时段 emptySlot=zeros(0,3);

nowSchedule=sortrows(nowSchedule,[2,4]);%先按照机器编码和开工时间升序排序 for k=1:nowMachQty

midMachId=nowMachIds(k); midSch=nowSchedule(nowSchedule(:,2)==midMachId,:); myQty=size(midSch,1);

if(myQty==0)

midSlot=[midMachId 0 bigM]; emptySlot=[emptySlot;midSlot];

else

preEmptyTime=0; for kk=1:myQty

midSlot=[midMachId preEmptyTime midSch(kk,4)]; emptySlot=[emptySlot;midSlot]; preEmptyTime=midSch(kk,5);

end

end

end

midSlot=[midMachId preEmptyTime bigM]; emptySlot=[emptySlot;midSlot];

%对emptySlot进行排序，按照slot的开始时间升序排序 emptySlot=sortrows(emptySlot,2); emptyQty=size(emptySlot,1);

%依次判断emptySlot中的空闲间距是否能够放下当前操作 jCanStartTime=jobPreEndTime(nowJobId); jobPTime=ptimes(nowJobId,i);

for kk=1:emptyQty midCanStartTime=max(jCanStartTime,emptySlot(kk,2)); if (emptySlot(kk,3)-midCanStartTime>=jobPTime)

myEndTime=midCanStartTime+jobPTime;

midSch=[nowJobId emptySlot(kk,1) i midCanStartTime myEndTime]; schedule=[schedule;midSch]; jobPreEndTime(nowJobId)=myEndTime;

break;

end

end

end

end end

%根据调度方案绘制甘特图程序

%schedule:1-jobId,2-machId,3-procId,4-startTime,5-endTime function drawGant(schedule)

rows=size(schedule,1);

maxMachId=max(schedule(:,2)); jobQty=max(schedule(:,1));

mycolor=rand(jobQty,3); figure;

ylim([0 maxMachId+1]); for i=1:rows

x=schedule(i,4:5); y=[schedule(i,2) schedule(i,2)];

line(x,y,'lineWidth',16,'color',mycolor(schedule(i,1),:)); procId=schedule(i,3);

jobId=schedule(i,1);

txt=['[' int2str(jobId) ' ' int2str(procId) ']']; text(mean(x)-1,y(1),txt);

end end