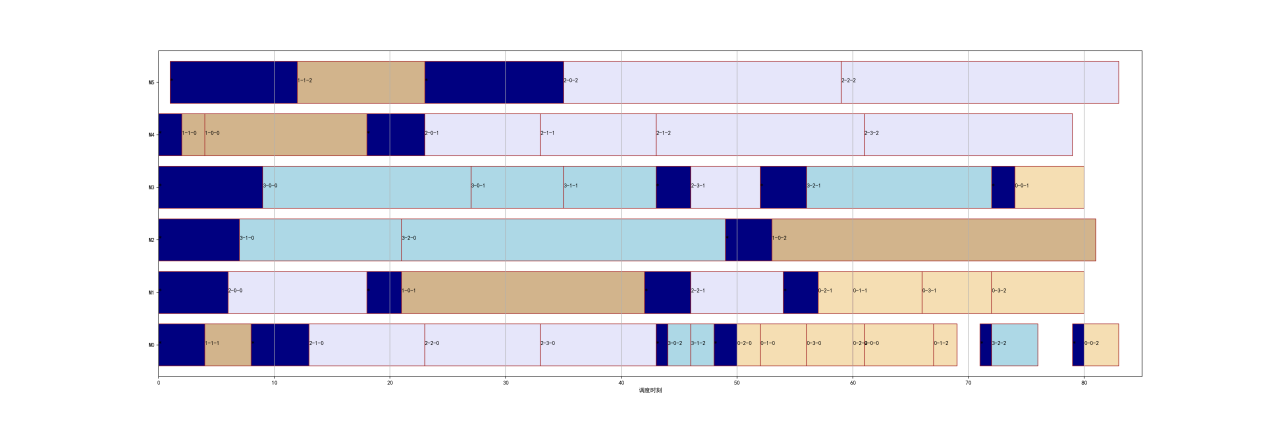
# 编码方式

## 实验结果：

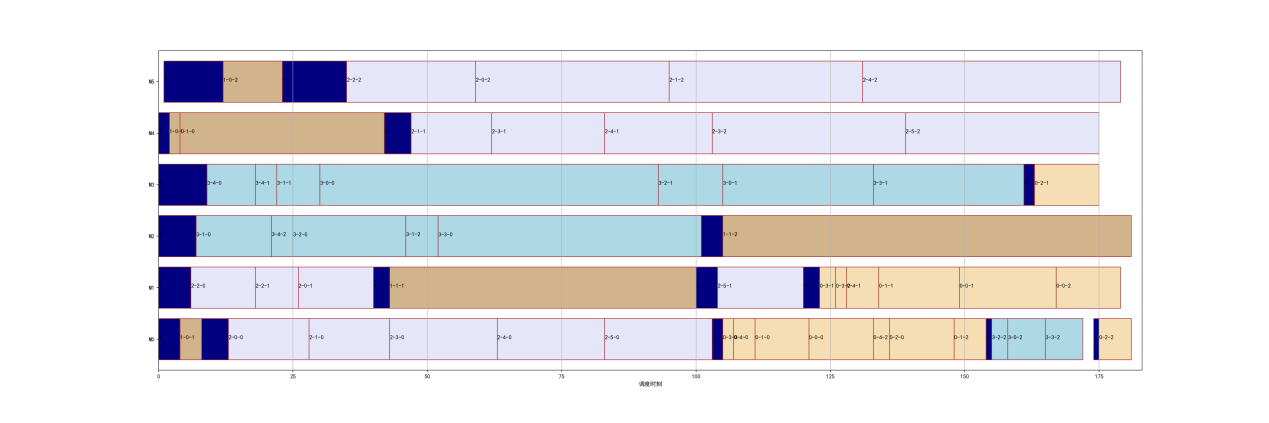
我的编码方式能找到的最好的解

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 问题 | 别人的最小完工时间 | 我的最小完工时间 | gant数据 | 编码 |
| P1 | 85 | 83 | bestgant-P1.csv | [4, 2, 4, 3], [[3, 2, 1, 2], [7, 1], [2, 2, 2, 2], [2, 2, 4]], [[1, 2, 3, 0], [1, 2, 3, 0], [3, 1, 0, 2], [3, 1, 0, 2], [2, 0, 3, 1], [1, 2, 3, 0]] |
| P2 | 183 | 181 | bestgant-P2.csv | [5, 2, 6, 5], [[6, 5, 6, 1, 2], [1, 19], [3, 3, 2, 4, 4, 4], [7, 2, 3, 7, 1]], [[1, 2, 0, 3], [1, 2, 0, 3], [3, 1, 0, 2], [3, 0, 2, 1], [3, 2, 0, 1], [2, 1, 3, 0]] |
| P3 | 213 | 192 | bestgant-P3.csv | [5, 3, 4, 3, 6, 6], [[1, 2, 1, 3, 3], [4, 5, 1], [2, 2, 2, 4], [4, 4, 2], [1, 1, 2, 2, 2, 2], [2, 2, 2, 1, 1, 2]], [[2, 3, 5, 0, 4, 1], [2, 1, 5, 0, 4, 3], [1, 2, 3, 4, 5, 0], [0, 1, 3, 5, 2, 4], [5, 4, 0, 2, 3, 1], [3, 5, 2, 4, 0, 1]] |
| P4 | 415 | 370 | bestgant-P4.csv | [7, 7, 8, 6, 8, 8], [[5, 2, 4, 1, 2, 1, 5], [3, 2, 3, 3, 4, 1, 4], [4, 4, 1, 2, 3, 4, 1, 1], [4, 4, 4, 2, 2, 4], [1, 2, 3, 5, 2, 5, 1, 1], [2, 2, 3, 3, 3, 1, 3, 3]], [[4, 0, 3, 2, 1, 5], [0, 1, 2, 3, 4, 5], [1, 3, 4, 5, 0, 2], [2, 0, 1, 5, 3, 4], [4, 5, 0, 3, 2, 1], [1, 3, 4, 5, 0, 2]] |

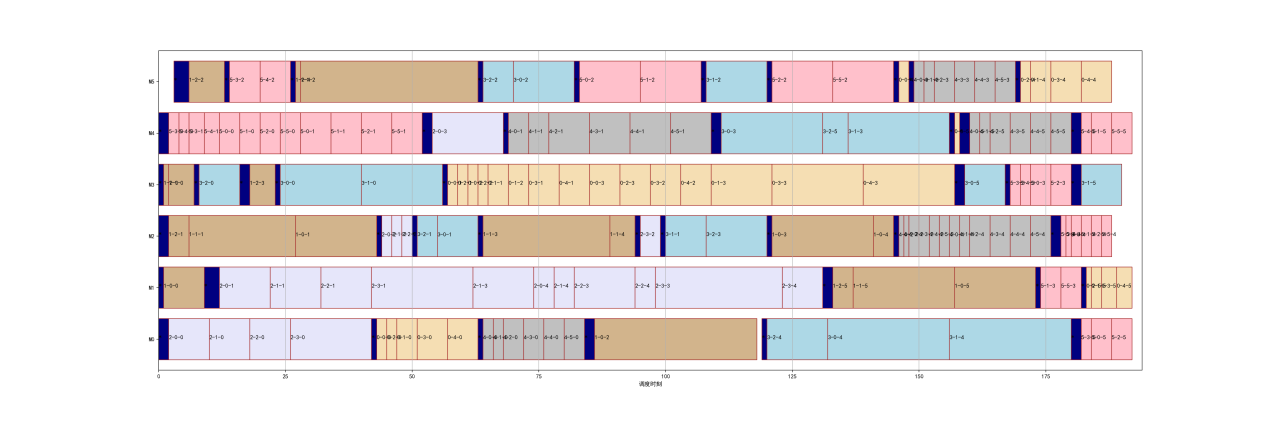
P1：



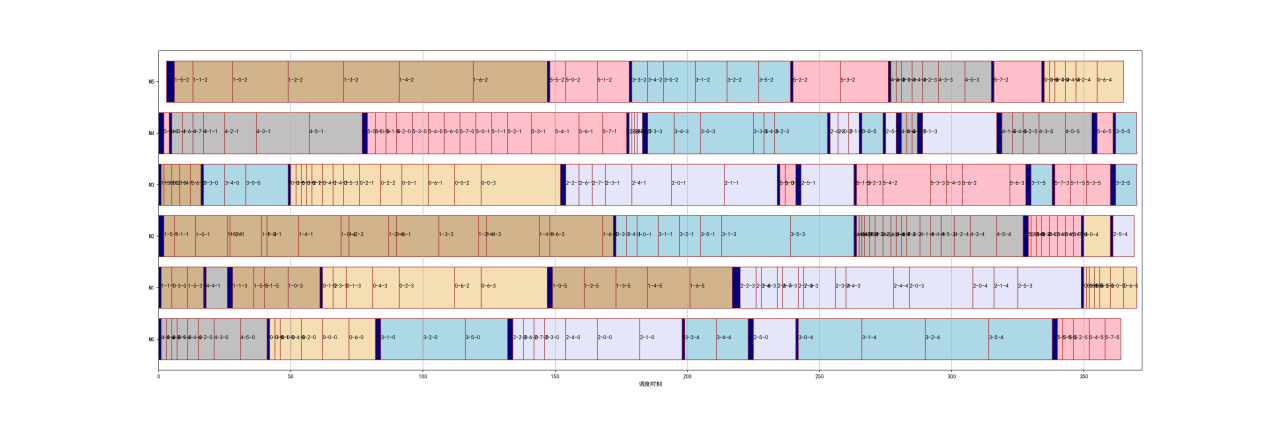
P2：



P3:



P4：



# 其他算法

## 数据结果

### IMGA

原始的岛群GA

参数：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| outerIterNum | innerIterNum | p1 | p2 | p3 | ps1 |
| 100 | 10 | 0.8 | 0.3 | 0.3 | 0.4 |
| ps2 | ps3 | ps4 | ps5 | mode | migratePercentage |
| 0.4 | 0.3 | 0.3 | 0.3 | ‘exchange’ | 10 |

原始数据：

|  |  |
| --- | --- |
| P1 | 85, 85, 85, 83, 85, 85, 85, 89, 85, 86, 85, 86, 85, 86, 85, 85, 85, 85, 85, 85 |
| P2 | 185, 183, 187, 188, 187, 185, 187, 185, 185, 186, 185, 185, 187, 188, 189, 189, 189, 189, 190, 185 |
| P3 | 195, 198, 197, 199, 197, 197, 194, 195, 195, 196, 194, 193, 194, 199, 195, 196, 197, 198, 194, 195 |
| P4 |  |

### NIMGA

来自An effective new island model genetic algorithm for job shop scheduling problem

P2~p3都改为了0.5，要重新实验。改的目的是，细粒度的改变只能完全依靠mutation。而不能依赖交叉，所以mutation的概率要适当提高

参数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iterNum | islandSize | p1 | p2 | p3 |
| 1000 | 18 | 0.8 | 0.3 | 0.3 |

原始数据：

|  |  |
| --- | --- |
| P1 | 85, 89, 85, 89, 83, 88, 86, 87, 85, 88, 92, 88, 85, 87, 83, 90, 89, 86, 89, 85 |
| P2 | 196, 192, 193, 195, 190, 191, 184, 185, 197, 185, 189, 199, 183, 193, 185, 193, 203, 189, 189, 188 |
| P3 | 202, 204, 209, 207, 203, 200, 207, 200, 206, 206, 201, 202, 199, 201, 208, 206, 199, 202, 202, 198 |
| P4 | 387, 383, 385, 391, 384, 391, 383, 384, 384, 380, 379, 392, 392, 386, 394, 381, 394, 396, 396, 386 |

# 原始MBO

## 参数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| iteration | popsize | K | S | M |
| 1000 | 51 | 3 | 1 | 10 |

## 特性：

1. 只有细粒度邻域算子，没有粗粒度的

2. 没有启发式的邻域算子

## 感性分析：

1. 很容易陷入局部最优，例如p=2时，整个种群就陷入了194，p=3时，整个种群就陷入了209。

2. 进化非常缓慢，可能是因为只用了细粒度邻域搜索

## 数据结果：

原始数据

|  |  |
| --- | --- |
| P1 | 86, 87, 84, 83, 88, 86, 89, 85, 85, 86, 83, 85, 87, 83, 84, 85, 85, 83, 84, 83 |
| P2 | 188, 190, 186, 194, 185, 192, 185, 189, 189, 186, 187, 185, 187, 188, 194, 195, 200, 194, 184, 185 |
| P3 | 197, 200, 201, 199, 198, 197, 196, 198, 204, 202, 199, 207, 198, 201, 199, 199, 200, 207, 201, 202, |
| P4 | 389, 385, 392, 378, 385, 381, 391, 385, 382, 389, 392, 384, 378, 390, 383, 385, 385, 378, 390, 388 |

统计数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| problem | Running times | mean | std | best | worse | Times of BK | BK |
| 1 | 20 | 85.5 | 1.7168284 | 83 | 89 | 5 | 83 |
| 2 | 20 | 189.15 | 4.234087 | 184 | 200 | 0 | 181 |
| 3 | 20 | 200.25 | 2.9474565 | 196 | 207 | 0 | 192 |
| 4 | 20 | 385.5 | 4.4553338 | 378 | 392 | 0 | 370 |

# 原始MBO+新邻域算子（启发式&粗粒度）

## 参数：

（除了额外的邻域算子以外，其他与原始MBO相同）

|  |  |
| --- | --- |
| S1coarse range | S2coarse range |
| 0.13 | 0.13 |

## 各个邻域算子起作用的次数：

单次run结果：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1n1 | S1n2 | S1n3 | S2n1 | S2n2 | S2n3 | S2n4 | S1coarse | S2coarse |
| P1 | 28 | 24 | 30 | 18 | 15 | 30 | 24 | 41 | 24 |
| P2 | 65 | 69 | 49 | 18 | 21 | 17 | 25 | 67 | 21 |
| P3 | 58 | 51 | 66 | 72 | 47 | 30 | 40 | 74 | 31 |
| P4 | 102 | 77 | 76 | 58 | 48 | 58 | 45 | 50 | 42 |

20run结果：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1n1 | S1n2 | S1n3 | S2n1 | S2n2 | S2n3 | S2n4 | S1coarse | S2coarse |
| P1 | 609 | 605 | 621 | 339 | 355 | 382 | 399 | 800 | 383 |
| P2 | 1492 | 1145 | 1010 | 426 | 443 | 499 | 564 | 896 | 553 |
| P3 | 924 | 951 | 1053 | 920 | 800 | 836 | 752 | 1132 | 790 |
| P4 | 1777 | 1541 | 1359 | 1059 | 924 | 887 | 886 | 1328 | 811 |
| 总 | 4802 | 4242 | 4043 | 2744 | 2522 | 2604 | 2601 | 4156 | 2537 |

额外加的实验（10run）：

P2：{'s1n1': 712, 's1n2': 542, 's1n3': 444, 's2n1': 192, 's2n2': 227, 's2n3': 231, 's2n4': 223, 's1coarse': 420, 's2coarse': 208}

P3: {'s1n1': 487, 's1n2': 459, 's1n3': 501, 's2n1': 464, 's2n2': 411, 's2n3': 378, 's2n4': 378, 's1coarse': 533, 's2coarse': 368}

## 分析：

1. 在各个算子机会均等的情况下，S1的算子效果好于S2的算子，说明邻域搜索的重点在于S1，应该减少S2算子的数量。其中的原因是，S1的解空间比S2的大一些，可搜索范围大一些。

下面是平均单个算子在20run中成功的次数：

|  |  |  |
| --- | --- | --- |
|  | S1 | S2 |
| P1 | 2635/4=658.75 | 1858/5=371.6 |
| P2 | 4543/4=1135.75 | 2485/5=497 |
| P3 | 4060/4=1015 | 4096/5=819.6 |
| P4 | 6005/4=1501.25 | 4567/5=913.4 |
| 总 | 17243/16=1077.6875 | 13006/20=650.3 |

2. S1n1效果不错，在S1算子里排第1，说明加入启发式有效果。S1coarse效果也不错，排第3，比S1n3好，说明粗粒度有效。

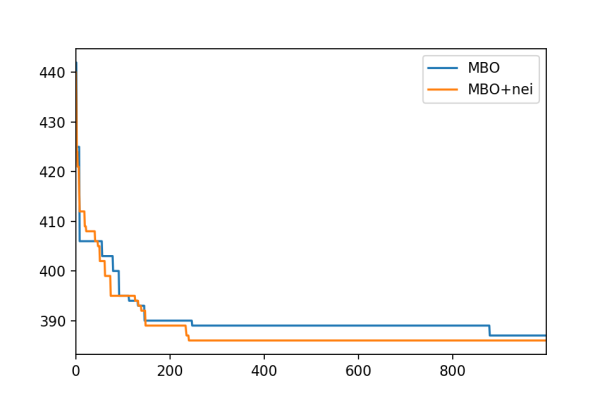
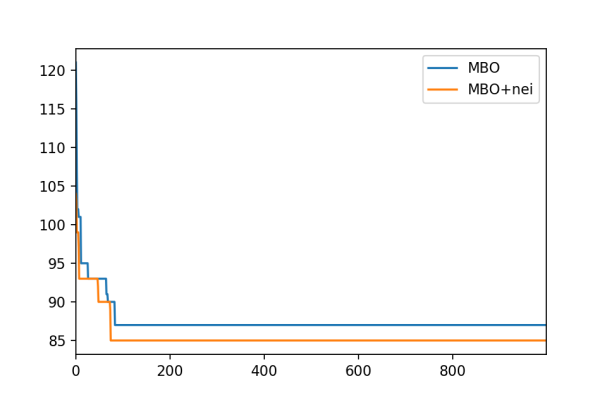
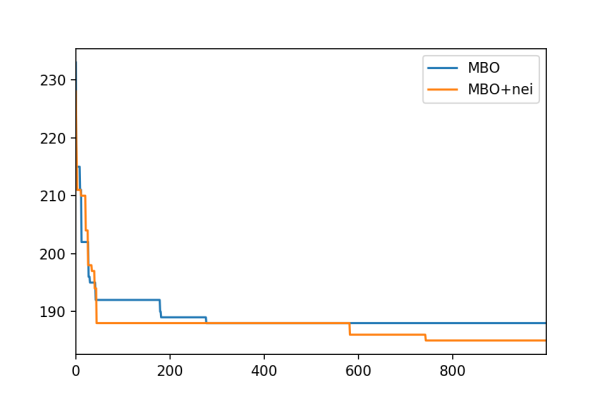
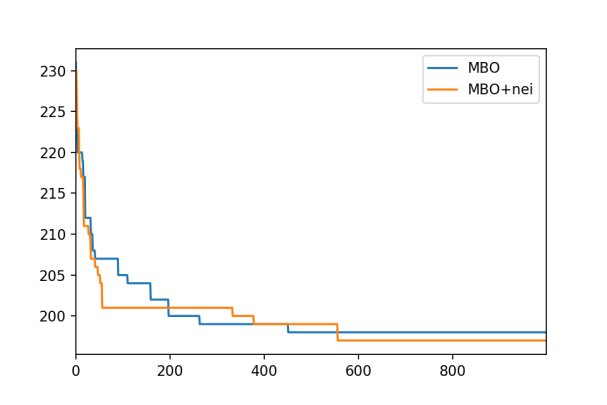
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | S1n1 | S1n2 | S1n3 | S1coarse |
| 次数 | 4802 | 4242 | 4043 | 4156 |
| 排名 | 1 | 2 | 4 | 3 |

3. S2n1效果不错，在S2算子里排第1，说明加入启发式有效果。S2coarse效果也不错，排第4，比S2n2，说明粗粒度有效。

粗粒度在一定程度上能代替S2n2中insert的效果，而且S2n2效果是最差的，因此考虑不使用S2n2，留下4个S2算子。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | S2n1 | S2n2 | S2n3 | S2n4 | S2coarse |
| 次数 | 2744 | 2522 | 2604 | 2601 | 2537 |
| 排名 | 1 | 5 | 2 | 3 | 4 |

## 收敛曲线：



## 数据结果：

还在不断补充数据，后期要更新统计数据

原始数据

|  |  |
| --- | --- |
| P1 | 85, 85, 85, 83, 89, 88, 86, 89, 85, 87, 85, 87, 85, 85, 83, 83, 85,85, 85, 87 |
| P2 | 189, 184, 189, 184, 192, 191, 185, 185, 185, 191, 185, 193, 195,185, 196, 199, 193, 187, 189, 187, 183 |
| P3 | 195,194, 203, 196, 202, 203, 200, 204, 201, 198, 201, 200, 201, 199, 196,202, 204, 202, 202, 200, 201, 198, 202, 203, 196, 200, 197, 200, 202, 200 |
| P4 | 386, 401, 389, 389, 381, 387, 381, 377, 382, 384, 386, 388, 387,387, 379, 386, 383, 374, 383, 376 |

统计数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| problem | Running times | mean | std | best | worse | Times of BK | BK |
| 1 | 20 | 85.6 | 1.7146428 | 83 | 89 | 3 | 83 |
| 2 | 20 | 189.2 | 4.3081318 | 183 | 199 | 0 | 181 |
| 3 | 20 | 200.15 | 2.8857408 | 194 | 204 | 0 | 192 |
| 4 | 20 | 384.5 | 5.7192656 | 37 | 401 | 0 | 370 |

# myMBO1

## 参数：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| iteration | popsize | K | S | M | A |
| 1000 | 51 | 3 | 1 | 8 | 2 |

A是队形调整阶段的迭代次数

## 特性：

1. 邻域共享改为交叉

2. 加入模糊顺序V字型

3. 加入队形调整阶段

## 数据结果：

原始数据：

|  |  |
| --- | --- |
| P1 | 83, 85, 85, 83, 85, 85, 85, 85, 85, 85, 85, 85, 86, 86, 85, 85, 88, 85, 87, 88 |
| P2 | 189, 187, 187, 194, 183, 195, 188, 183, 192, 189, 189, 187, 189, 191, 184, 190, 183, 189, 187, 189 |
| P3 | 198, 200, 196, 199, 197, 196, 197, 197, 198, 196, 199, 199, 200, 198, 196, 197, 199, 199, 198, 196 |
| P4 | 380, 378, 374, 383, 380, 385, 381, 378, 378, 375, 382, 382, 386, 384, 385, 377, 377, 382, 384, 376 |

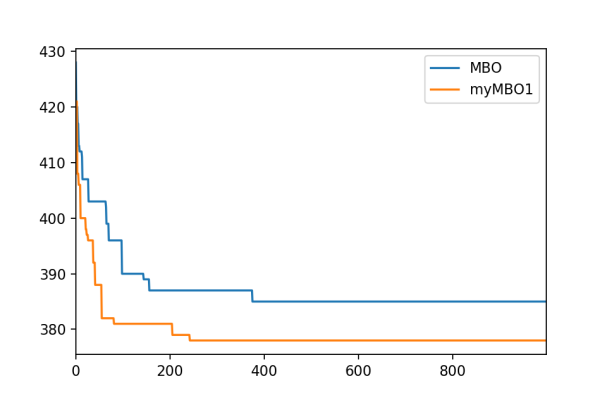
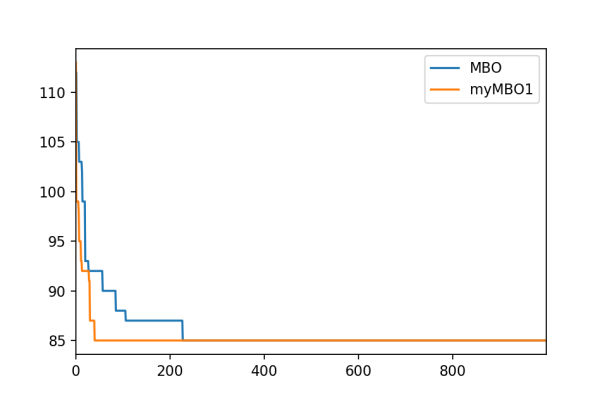
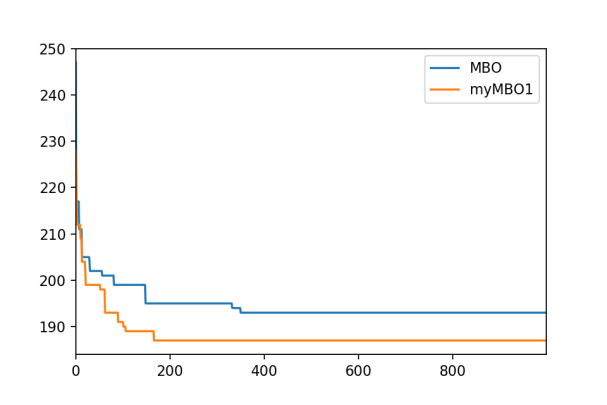
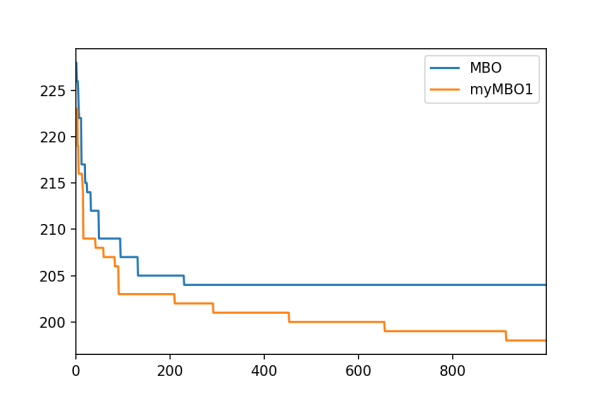
统计数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| problem | Running times | mean | std | best | worse | Times of BK | BK |
| 1 | 20 | 85.3 | 1.2288205 | 83 | 88 | 2 | 83 |
| 2 | 20 | 188.25 | 3.2691742 | 183 | 195 | 0 | 181 |
| 3 | 20 | 197.75 | 1.3369741 | 196 | 200 | 0 | 192 |
| 4 | 20 | 380.35 | 3.4967842 | 374 | 386 | 0 | 370 |

各个邻域成功次数（20run）：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | S1n1 | S1n2 | S1n3 | S2n1 | S2n2 | S2n3 | S2n4 | S1coarse | S2coarse |
| P1 | 472 | 520 | 558 | 283 | 299 | 341 | 339 | 764 | 293 |
| P2 | 1096 | 988 | 851 | 346 | 409 | 412 | 407 | 873 | 412 |
| P3 | 797 | 1016 | 1321 | 920 | 1046 | 968 | 929 | 1358 | 914 |
| P4 | 1469 | 1633 | 1524 | 1040 | 1077 | 1063 | 997 | 1399 | 841 |
| 总 |  |  |  |  |  |  |  |  |  |

## 收敛曲线：



# myMBO1+aging

## 参数：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| iteration | popsize | K | S | M | agingThreshold |
| 500 | 51 | 3 | 1 | 10 | [10, 30, 40] |

## 特性：

在myMBO1的基础上，加上aging策略：

age< agingThreshold[0]，用一次一步邻域算子

agingThreshold[0]<=age< agingThreshold[1]，用两次一步邻域算子

agingThreshold[1]<=age< agingThreshold[2]，用两次两步邻域算子

agingThreshold[2]<=age，重新初始化

## 数据结果：

从收敛图上看，500代还是少了点，后面要重新做实验

原始数据：

|  |  |
| --- | --- |
| P1 | 83, 83, 85, 85, 85, 85, 85, 83, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 85, 83 |
| P2 | 183, 187, 183, 187, 185, 186, 183, 185, 185, 185, 183, 185, 189, 184, 185, 189, 184, 183, 183, 181 |
| P3 | 194, 195, 196, 193, 194, 193, 194, 194, 195, 194, 195, 195, 192, 195, 193, 194, 195, 194, 195, 194 |
| P4 | 371, 375, 370, 374, 373, 371, 375, 376, 375, 373, 374, 370, 374, 371, 373, 377, 376, 376, 374, 374 |

统计数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| problem | Running times | mean | std | best | worse | Times of BK | BK |
| 1 | 20 | 84.6 | 0.7999999 | 83 | 85 | 4 | 83 |
| 2 | 20 | 184.75 | 2.0217566 | 181 | 189 | 1 | 181 |
| 3 | 20 | 194.2 | 0.9273618 | 192 | 196 | 192 | 192 |
| 4 | 20 | 373.6 | 2.0346989 | 370 | 377 | 2 | 370 |

## Aging成功次数：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 一次一步 | 两次一步 | 两次两步 | 重新初始化 |
| P1 | 12179 | 643 | 51 | 8241 |
| P2 | 15663 | 1257 | 113 | 7060 |
| P3 | 18846 | 3948 | 469 | 5507 |
| P4 | 21526 | 4694 | 506 | 5141 |

感觉还是改为成功百分比比较好，即成功次数/试探次数，这样才公平

## 收敛曲线：

