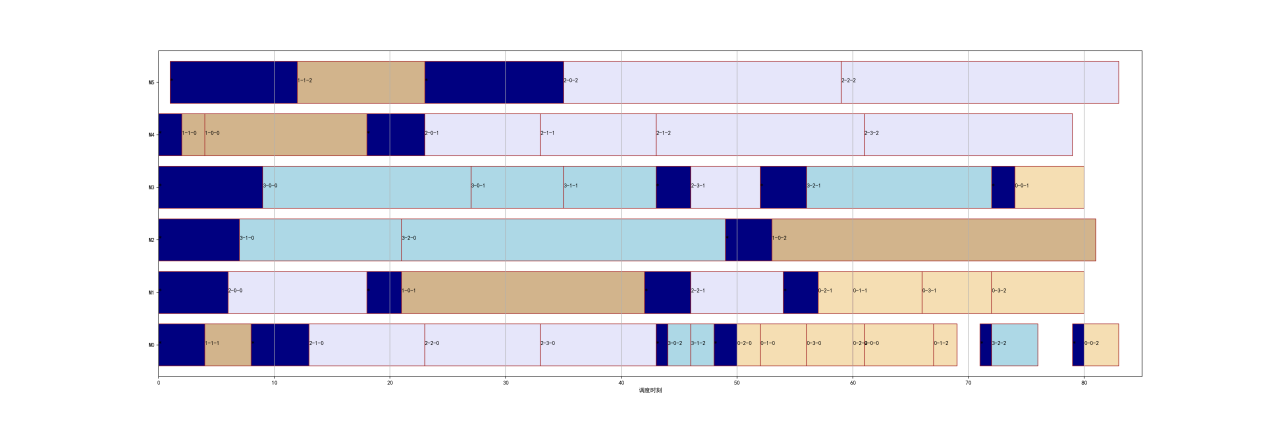
# 编码方式

## 实验结果

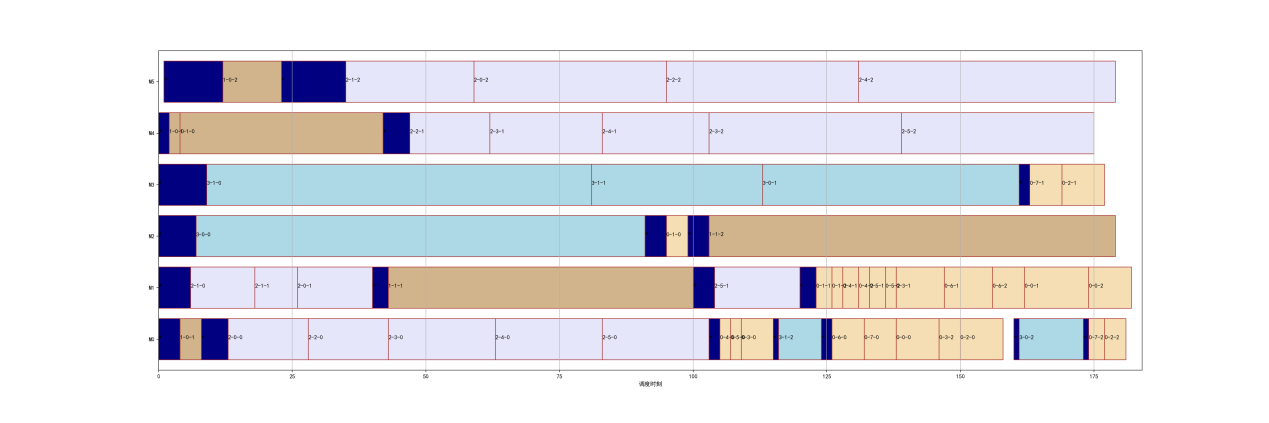
我的编码方式能找到的最好的解

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 问题 | 别人的最小完工时间 | 我的最小完工时间 | gantchart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 | 182 | bestgant-P2.csv | [8, 2, 6, 2],[[4, 1, 4, 3, 1, 1, 3, 3], [1, 19], [3, 2, 3, 4, 4, 4], [12, 8]],[[1, 2, 3, 0], [1, 2, 0, 3], [3, 1, 0, 2], [3, 0, 2, 1], [2, 3, 1, 0], [2, 0, 1, 3]] |
| P3 | 213 | 194 | bestgant-P3.csv | [5, 6, 5, 2, 5, 6], [[2, 2, 2, 2, 2], [1, 1, 1, 1, 3, 3], [3, 1, 2, 2, 2], [5, 5], [2, 2, 2, 2, 2], [3, 2, 1, 2, 1, 1]], [[5, 2, 3, 0, 1, 4], [1, 2, 0, 5, 4, 3], [1, 3, 4, 5, 2, 0], [2, 1, 3, 0, 4, 5], [5, 4, 3, 0, 2, 1], [1, 5, 4, 0, 2, 3]]  Or  [6, 6, 6, 2, 2, 5], [[2, 2, 1, 2, 2, 1], [2, 1, 1, 2, 1, 3], [1, 1, 2, 1, 2, 3], [4, 6], [6, 4], [2, 2, 3, 2, 1]], [[2, 4, 5, 3, 0, 1], [2, 1, 0, 4, 5, 3], [1, 2, 3, 4, 0, 5], [0, 1, 3, 5, 4, 2], [3, 4, 5, 2, 0, 1], [5, 3, 4, 1, 2, 0]] |
| P4 | 415 | 374 | bestgant-P4.csv | [3, 7, 6, 8, 7, 3], [[8, 10, 2], [1, 2, 4, 5, 4, 2, 2], [1, 3, 2, 3, 6, 5], [4, 3, 3, 2, 2, 1, 1, 4], [1, 2, 2, 1, 5, 7, 2], [9, 3, 8]], [[0, 5, 4, 2, 3, 1], [0, 1, 2, 4, 5, 3], [1, 3, 5, 4, 0, 2], [2, 1, 0, 5, 3, 4], [4, 5, 0, 3, 2, 1], [3, 1, 5, 4, 2, 0]] |

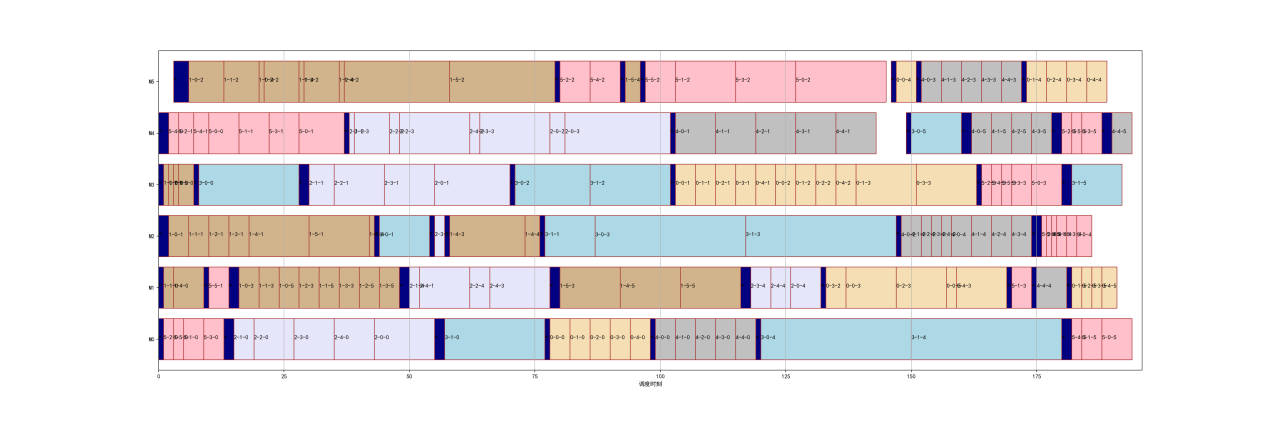
P1：



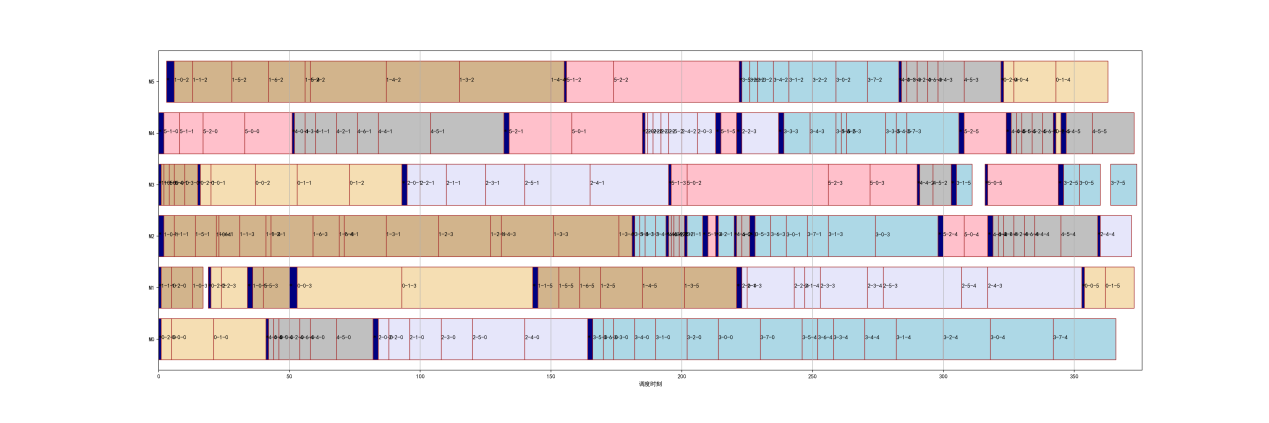
P2：



P3:



P4：



# 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 | 182 |
| 3 | 20 | 200.25 | 2.9474565 | 196 | 207 | 0 | 194 |
| 4 | 20 | 385.5 | 4.4553338 | 378 | 392 | 0 | 374 |

# 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 |

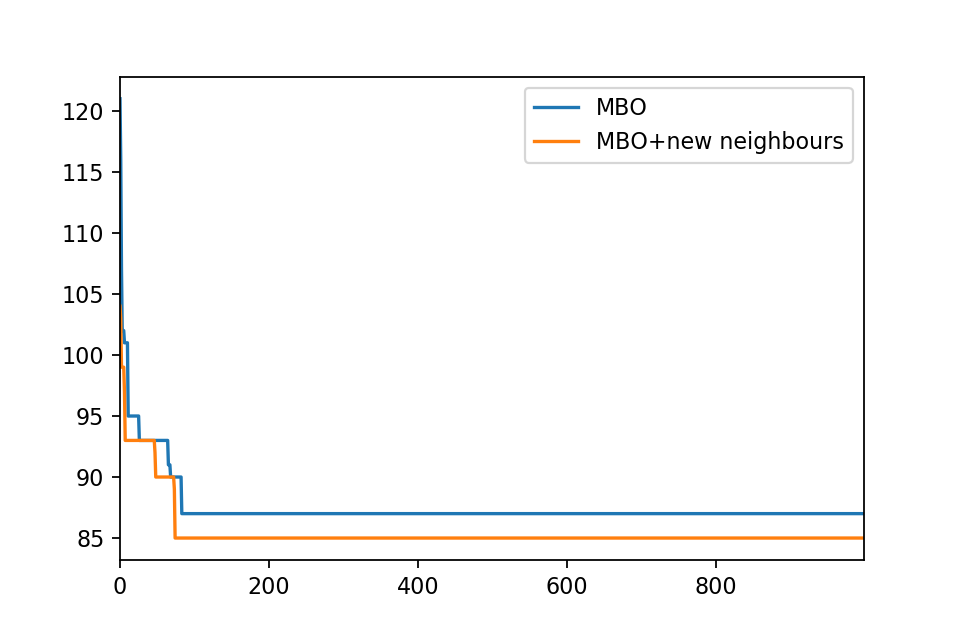
3. S2n1效果不错，在S2算子里排第1，说明加入启发式有效果。S2coarse效果也不错，排第4，比S2n2，说明粗粒度有效。

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。

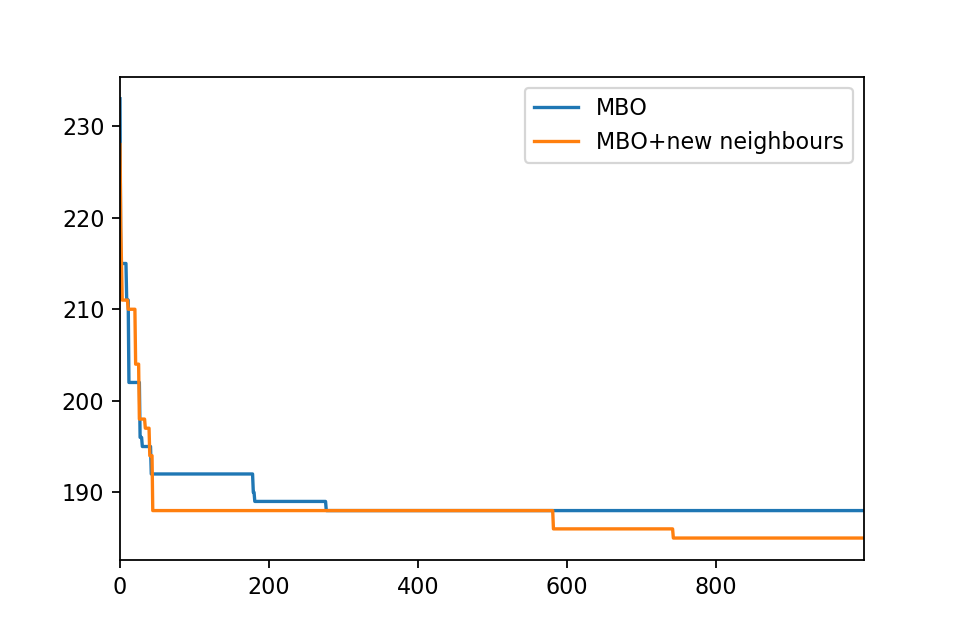
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | S2n1 | S2n2 | S2n3 | S2n4 | S2coarse |
| 次数 | 2744 | 2522 | 2604 | 2601 | 2537 |
| 排名 | 1 | 5 | 2 | 3 | 4 |

## 对比收敛曲线：

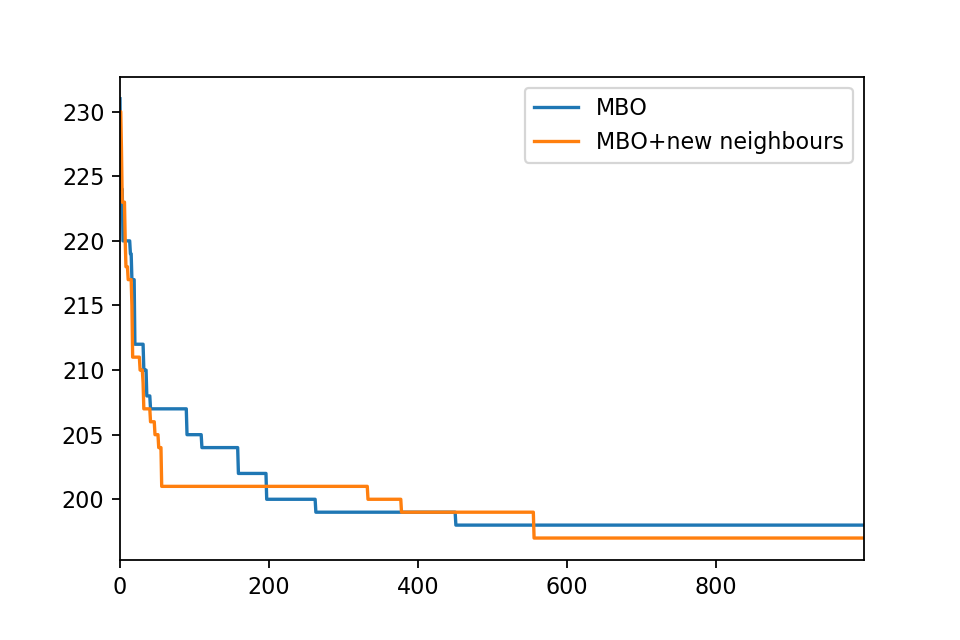
P1



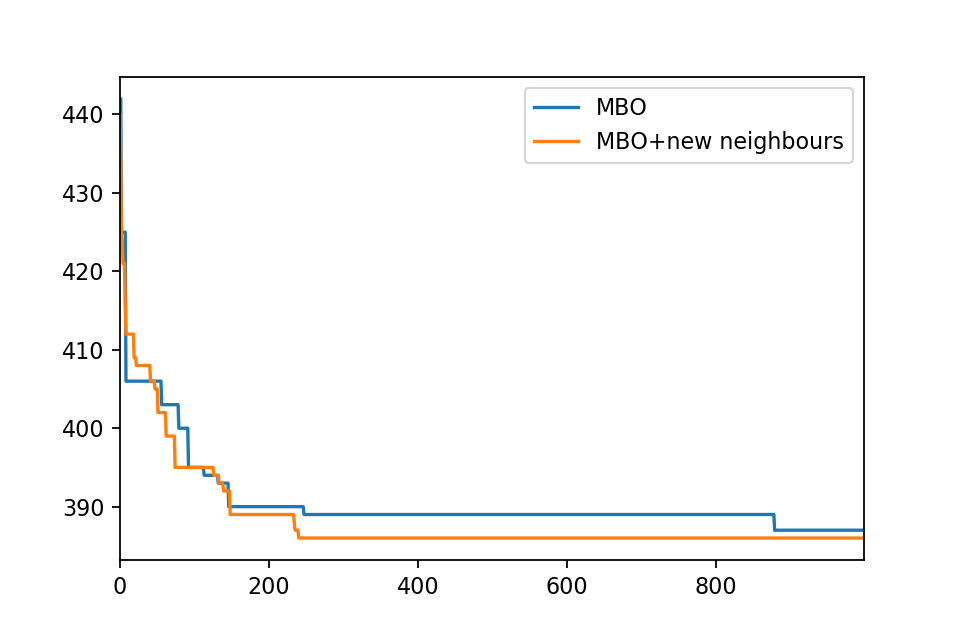
P2



P3



P4



## 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, 193, 198, 194, 190, 189, 183, 189, 190, 198, 191 |
| 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 | 184 | 199 | 0 | 182 |
| 3 | 20 | 200.15 | 2.8857408 | 194 | 204 | 1 | 194 |
| 4 | 20 | 384.5 | 5.7192656 | 37 | 401 | 1 | 374 |