Algorithm1:

Step 1: Pair the constrain as nx2 matrix where number in first column of matrix is in same group as number in second column

Step 2: Enter the number of gene (k)

Step 3: Make randomly k gene. Gene length is number of instances. So each number in gene represents in which group particular instance is located.

Step 4: Find objective function value which is needed to be maximize and sort each row in ascending order

Step 5: i= 1

Step 6: Do mutation for instances/10 times

Step 7: Check if constraints are satisfied otherwise change the value accordingly

Step 8: Do as Step 4

Step 9: i=i+1, Go to step6 till i<50. Now we will have relatively good answer.

Step10: Set j=1.

Step11: Make 6 genes out of consecutive 2 genes. Select best 2 out of 6. For next 2 genes keep the same value of both genes as it is and change whichever has different corresponding value. For next 2 genes keep the different value of both genes as it is and change whichever has same corresponding value.

Step12: Do as step7.

Step13: Do as step4.

Step 14: j=j+1, Go to step11 till j<50.

Step15: Find total distance of instances from its respective mean.

Step16: Give clusters and total distance

Algorithm2:

Step 1: Pair the constrain as nx2 matrix where number in first column of matrix is in same group as number in second column

Step 2: Make k centroid for each parameter varying from mean-2\*std to mean+2\*std

Step 3: Group element to nearest centroid

Step 4: Find number of element

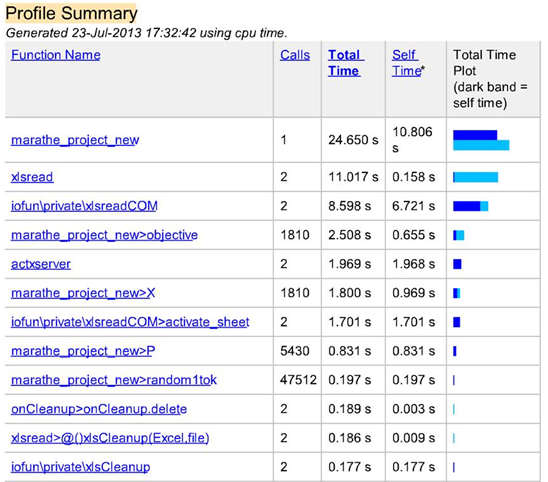
Step 5: Find new centroid for each group with new element

Step 6: Check terminating condition if terminated leave otherwise go to step 3

Step 7: Find total distance of instances from its respective mean.

Step 8: Give clusters and total distance

Run1: algorithm1



Group 1: 4 5 7 8 12 29 34 39 40 43 44 58 60 61 63 66 67 75 76 77 85 88 109 110 111 117 118 140 141

Group 2: 1 16 19 26 28 34 45 48 51 53 54 66 74 78 92 95 111 113 114 119 124 130 135 136

Group 3: 4 5 6 12 13 14 33 43 47 55 59 71 81 110 118 122 123 137 138

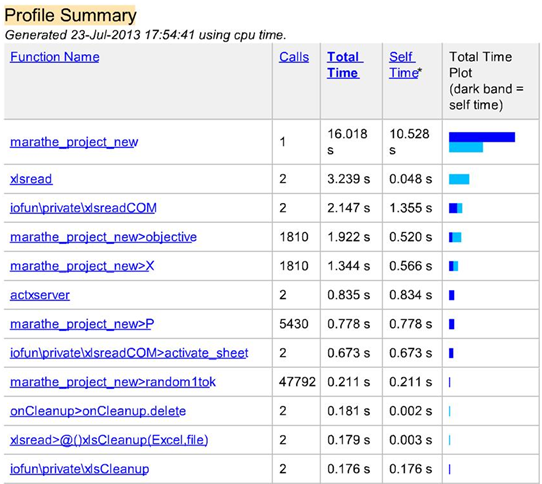
Group 4: 9 20 35 36 38 44 46 50 52 62 75 80 82 85 88 90 99 103 104 106 107 108 115 126 129 131

Group 5: 21 22 24 25 30 40 56 57 58 60 61 63 70 73 76 84 93 97 100 101 102 105 109 117 134

Group 6: 10 11 27 31 39 42 49 65 67 68 72 79 91 94 98 112 116 120 121 125 128 139 140 141

Result: 3.0167e+03

Run2: algorithm1



Group 1: 2 3 7 8 15 17 18 23 29 32 37 41 64 69 77 83 86 87 89 96 127 132 133

Group 2: 1 2 3 14 17 20 23 26 32 35 83 94 96 104 113 124 127 132

Group 3: 9 15 18 22 28 30 33 46 47 50 55 78 86 89 95 101 103 114 115 121 131

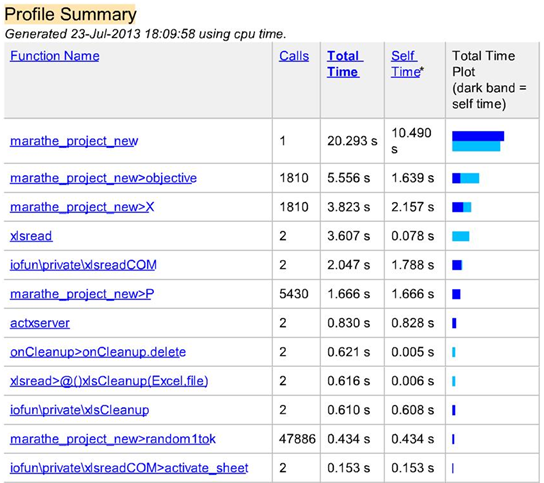
Group 4: 6 10 21 36 41 52 53 56 62 64 68 70 79 87 90 91 99 112 120 122 133 135 136 138

Group 5: 13 19 25 31 38 42 48 51 54 57 59 65 71 72 73 74 80 92 93 105 106 123 125 126 130 134 139

Group 6: 11 16 24 27 37 45 49 69 81 82 84 97 98 100 102 107 108 116 119 128 129 137

Result: 3.0369e+03

Run3: algorithm1



Group 1: 4 6 9 12 16 38 43 47 51 56 59 61 71 76 77 78 97 98 100 103 104 108 111 118 122 128 130 131 134

Group 2: 1 2 3 17 23 31 32 40 41 50 64 75 83 112 113 116 127

Group 3: 13 15 19 25 27 30 45 48 65 67 70 85 93 94 105 106 117 121 125 126 132 133 135 136 137 139

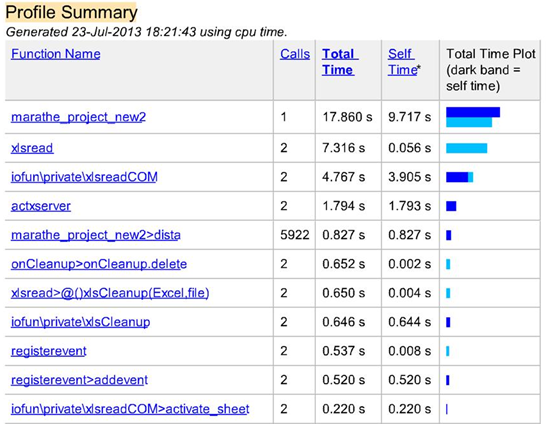
Group 4: 6 11 24 26 28 29 35 36 39 46 52 53 60 68 72 73 82 95 101 102 110 115 119

Group 5: 5 7 14 18 20 21 33 44 49 54 69 74 80 88 89 90 91 96 99 114 124 141

Group 6: 8 10 22 34 37 42 55 57 58 62 63 66 79 81 84 86 87 92 107 109 120 123 129 138 140

Result: 3.0696e+03

Run1: algorithm2



Group 1: 1 113

Group 2: 8 9 10 12 13 21 22 24 25 30 34 38 44 48 51 54 55 57 59 60 63 65 66 67 71 77 78 84 85 93 95 99 105 106 112 117 118 121 122 123 125 126 128 131 135 136 140

Group 3: 18 32 41 89 94

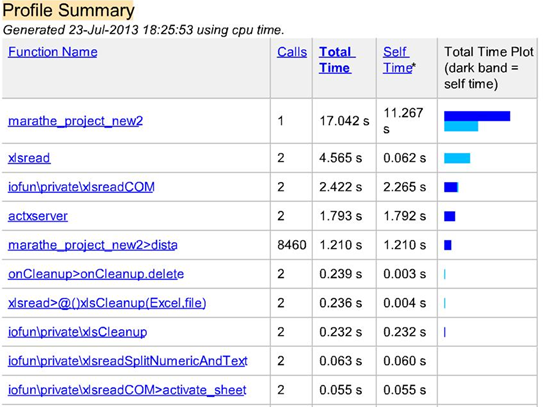
Group 4: 4 5 6 11 16 19 20 27 39 40 45 47 56 58 61 70 73 76 80 81 92 97 98 100 102 103 107 109 110 111 120 130 134 137

Group 5: 2 3 17 83

Group 6: 7 14 15 23 26 28 29 31 33 35 36 37 42 43 46 49 50 52 53 62 64 68 69 72 74 75 79 82 86 87 88 90 91 96 101 104 108 114 115 116 119 124 129 133 139

Result: 2.2003e+03

Run2: algorithm2



Group 1: 2 3 17 32 83 127

Group 2: 7 15 18 23 28 35 36 37 41 42 50 52 64 69 72 75 86 88 89 90 91 94 96 104 114 124 132 133 141

Group 3: 8 9 10 12 13 21 22 30 34 38 44 48 51 54 55 57 59 60 63 65 66 67 71 77 78 84 85 93 95 99 106 112 117 118 120 121 122 123 125 126 128 131 136 140

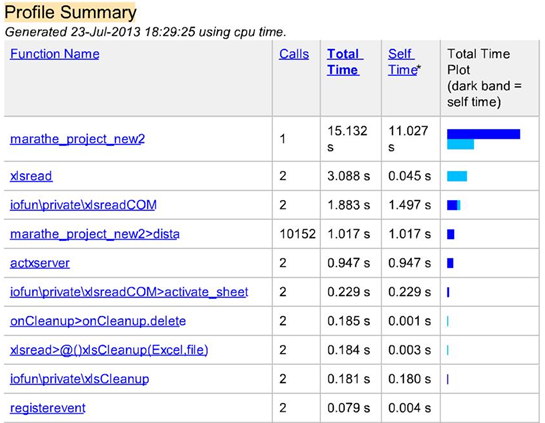
Group 4: 5 11 14 20 26 27 29 31 33 39 43 45 46 47 49 53 56 62 68 70 74 79 80 82 87 98 100 101 102 107 108 109 110 115 116 119 129 135 137 139

Group 5: 4 6 16 19 24 25 40 58 61 73 76 81 92 97 103 105 111 130 134 138

Group 6: 11 1 113

Result : 2.0701e+03

Run3: algorithm2



Group 1: 21 24 25 38 44 48 51 54 55 59 60 65 66 67 71 77 78 84 85 95 99 105 106 112 117 118 122 123 125 128 131 136 140

Group 2: 8 9 10 12 13 22 30 34 57 63 93 109 110 115 120 121 126 135

Group 3: 5 7 11 14 20 26 27 29 31 33 39 43 45 46 47 49 53 56 62 68 70 74 79 82 87 98 100 101 102 107 108 116 119 129 137 139

Group 4: 15 18 23 28 35 36 37 41 42 50 52 64 69 72 75 86 88 89 90 91 94 96 104 114 124 132 133 141

Group 5: 1 2 3 17 32 83 113 127

Group 6: 4 6 16 19 40 58 61 73 76 80 81 92 97 103 111 130 134 138

Result: 2.0858e+03

Result:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Total time (run1) | Self time (run 1) | Total time (run2) | Self time (run 2) | Total time (run3) | Self time (run 3) | Result  (run 1) | Result  (run 2) | Result  (run 3) |
| 1 | 24.650 s | 10.806 s | 16.018 s | 10.528 s | 20.293 s | 10.490 s | 3.0167e+03 | 3.0369e+03 | 3.0696e+03 |
| 2 | 17.860 s | 9.717 s | 17.042 s | 11.267 s | 15.132s | 11.027s | 2.2003e+03 | 2.0701e+03 | 2.0858e+03 |

Here result is distance of point from mean of its cluster. For better clustering result should be lower

Conclusion:

* From above table it can be deduce that time taken for algorithm 2 is lower than algorithm 1. So algorithm 2 is time efficient
* From above table it can be seen that result of algorithm 2 is much lower than algorithm1. So algorithm 2 gives better clustering

We can say that algorithm 2 is better clustering algorithm.